UNIVERSIDAD DE MURCIA

FACULTAD DE INFORMÁTICA

Metodologías basadas en Minería de Datos para el diseño y optimización de técnicas de Clasificación Automática

Dña. Raquel Martínez España

2014

UNIVERSIDAD DE MURCIA

FACULTAD DE INFORMÁTICA

METODOLOGÍAS BASADAS EN MINERÍA DE DATOS PARA EL DISEÑO Y OPTIMIZACIÓN DE TÉCNICAS DE CLASIFICACIÓN AUTOMÁTICA

Tesis doctoral presentada por Raquel Martínez España

dentro del Programa de Doctorado en Informática

Dirigida por Dr. José Manuel Cadenas Figueredo

y Dra. María del Carmen Garrido Carrera

UNIVERSIDAD DE MURCIA

FACULTAD DE INFORMÁTICA

METODOLOGÍAS BASADAS EN MINERÍA DE DATOS PARA EL DISEÑO Y OPTIMIZACIÓN DE TÉCNICAS DE CLASIFICACIÓN AUTOMÁTICA

Tesis doctoral presentada por Raquel Martínez España

dentro del Programa de Doctorado en Informática

Dirigida por Dr. José Manuel Cadenas Figueredo

y Dra. María del Carmen Garrido Carrera

Raquel Martínez José Manuel Cadenas María del Carmen Garrido

Departamento de Ingeniería de la Información y las Comunicaciones

Murcia, Octubre de 2014

A los pilares fundamentales que componen mi vida,
mis padres y mi marido Paco.

Índice General

Agradecimientos

Llegó el día de tener que escribir los agradecimientos de esta tesis. Siempre había pensado que sería la parte más sencilla de la tesis, pero ahora que estoy escribiéndolos no estoy tan segura que sea una tarea tan fácil.

Deseo comenzar dando las gracias a la Fundación Séneca - Agencia de Ciencia y Tecnología de la Región Murcia por haberme concedido la beca FPI. Dicha financiación también ha hecho posible mi estancia en el grupo de investigación ASAP de la universidad de Nottingham, que también debo de agradecer a los profesores Bob John y Dario Landa Silva. Además gracias al Ministerio de Ciencia e Innovación de España y al Fondo Europeo de Desarrollo Regional (FEDER), por los proyectos de investigación TIN2008-06872-C04-03 y TIN2011-27696-C02-02, en los cuales he tenido la oportunidad de participar, los cuales pertenecen a proyectos coordinados dirigidos por los doctores D. José Andrés Moreno Pérez y D. Jose Luis Verdegay Galdeano, respectivamente. También, gracias de nuevo a la Fundación Séneca por el apoyo dado por medio del proyecto 04552/GERM/06 de grupos de excelencia concedido al grupo de investigación Sistemas Inteligentes y Telemática al cual pertenezco. Y por supuesto, gracias a mi departamento de Ingeniería de la Información y las Comunicaciones por todo el apoyo en infraestructura y gestión.

También deseo dar las gracias a todas las personas que han compartido sala conmigo, pero en especial quisiera dar las gracias a Ramón Andrés Díaz por los los ánimos y las palabras de apoyo que siempre ha tenido para mí, a Enrique Muñoz por la tranquilidad que transmite y sus buenos consejos y a Alejandro Martínez por sus ánimos, sus palabras y su colaboración en algunos trabajos. No podría olvidarme de dos profesores que comenzaron formándome en el instituto y a los que les tengo mucho cariño, Ana María Valencia y Manuel Suárez. Muchas gracias por preocuparos siempre por mí y por las magníficas palabras que tenéis siempre para animarme.

Seguidamente, quiero dar mucho más que las gracias a mis directores, D. José Manuel Cadenas y Dña. María del Carmen Garrido. Os agradezco de todo corazón, no sólo la estupenda dirección de esta tesis, sino el trato tan humano y tan cercano que habéis tenido conmigo, ya no sólo durante esta tesis, sino también en todos los años anteriores que hemos trabajado juntos. Porque de una cosa estoy segura y aunque suene un poco a "peloteo" os lo digo de verdad, jamás en la vida tendré "jefes" como vosotros, como mucho igual, pero nunca mejor, porque el

trato, el ambiente de trabajo día a día, la confianza y los consejos ya no sólo profesionales sino también a nivel personal jamás se podrán ver superados. Muchas gracias por todo.

Pero qué sería una tesis sin amigos, aquellos que te hacen olvidar por momentos las preocupaciones del trabajo. Gracias a todos mis amigos y amigas, no os pongo a todos que no quiero olvidarme de nadie. Sin embargo, sí deseo hacer especial mención a cuatro amigos con los que he pasado momentos muy especiales y divertidos, que seguro repetiremos en el futuro. Muchas gracias Francis, Noelia, Ospy y Ester por ser como sois.

Además, quiero dar las gracias a mi abuela y toda mi familia en general, tanto la que me toca de cerca, como a mi familia política, gracias por estar siempre ahí. Pero quiero hacer mención especial en estos agradecimientos a mis hermanas, Miriam y Lorena, sin las cuales yo sé que mi mundo no sería lo que es hoy. Muchas gracias por ser como sois, por vuestras palabras del día a día y por la ya famosa frase "Nosotras sabemos que tú puedes con todo y más, seguro que vas a aprobar", palabras que escuchaba después de cada examen o reto al que me enfrentaba. Muchas gracias de verdad. Pero cómo olvidarme de las personas que me dieron la vida, mis padres, Antonio y Manoli. Gracias a ellos estoy donde estoy y he conseguido todo lo que me he propuesto, porque desde siempre me han dicho que "lo difícil se consigue y lo imposible se intenta" y que "la suerte en la vida hay que buscarla, sola no viene". Pero la suerte la he tenido yo por tener unos padres como ellos, y quisiera aprovechar estos agradecimientos para que quede constancia de lo importantes que son en mi vida y que me faltan palabras para agradecerles todo lo que han hecho por mí.

Por último, aunque no menos importante, no podía terminar estos agradecimientos sin darle mucho más que las gracias a la persona que me hace la mujer más feliz del mundo día a día, mi marido Paco. Muchas gracias por todos los años que llevas a mi lado soportando las horas y horas que paso al ordenador, por sacarme una sonrisa cuando más la he necesitado, por tu apoyo incondicional, por tus palabras, por tus consejos, por las horas y minutos escuchándome sin saber muy bien de qué hablaba y sobre todo gracias por tu comprensión, te quiero.

Muchas gracias,
Raquel Martínez España
diciembre 2014

Abstract

The advance of new technologies has allowed the storage of large volumes of information, composed of different types of data, that are not always as precise and perfect as one would desire. It is increasingly common to find imperfect data due to the facts that we usually use vague or imprecise quantifiers (pretty young, too old, too small, too large, etc.), the mistakes often made by the measuring instruments used to obtain such information, there are missing data, etc.. However, despite the fact that imperfection is present directly in data, the number of techniques within the discipline of Intelligent Data Analysis explicitly enabling the treatment of this type of information are few to date. In this way, if the techniques are not able to deal with imperfect data, these data are transformed into precise data and during this transformation process, it is possible to lose relevant information. This loss can affect directly the quality of the results expected.

Due to the problem of shortage of techniques that work directly with imperfect data, in this work we propose to develop techniques within the Intelligent Data Analysis, that allow the treatment of datasets that may explicitly contain imperfect data. To perform this proposal, the methodologies that Softcomputing provides will be combined with the techniques of Intelligent Data Analysis. On the one hand, to extend some of these techniques in order to provide them with the ability to deal explicitly with imperfect data. On the other hand, to design new techniques that are able to work directly with such data.

As we have said, in a wide variety of areas and fields, data have been collected and accumulated at a dizzying pace, hence the growing need for a new computational generation of theories, tools and techniques that help to extract useful knowledge quickly from large volumes of information. This new computational generation is located in the field of Intelligent Data Analysis. At a certain level of abstraction, the area of Intelligent Data Analysis is concerned about the development of techniques that give meaning to the data collected. However, the basic problem addressed by the process of Intelligent Data Analysis is the mapping of lower level data in other ways that can be more compact, more abstract or more useful. Thus, Intelligent Data Analysis is defined as a complex process of information discovering or useful knowledge from the data, where data play the most important role in the discipline, which is divided into several phases:

- Integration and data collection

- Data preprocessing
- Data Mining
- Evaluation and interpretation
- Dissemination and use of models

Because data are the raw material for the process of Intelligent Data Analysis and these could be imperfect, the process must incorporate various methodologies that are tolerant to imprecision, uncertainty and partial truth. These methodologies have been coined with the term Softcomputing. The term Softcomputing can be defined as a set of methodologies developed to deal with the real practical situations in the same way that humans tend to do, that is, based on intelligence, common sense, consideration of analogies, approaches, etc.. Softcomputing is positioned as the theoretical basis of the area of Intelligent Systems, and makes clear that the difference between the area of classical Artificial Intelligence, and the area of Intelligent Systems is that the first is based on the so-called Hardcomputing, while the second is based on Softcomputing.

In this work we focused on the phases of data preprocessing and data mining of Intelligent Data Analysis, as we have said, to extend existing techniques and to design new techniques within these two phases in order that techniques can work with imperfect data (low quality data) directly without the need of a prior transformation. Figure 1 shows the general scheme of this work, done in the field of Intelligent Data Analysis of low quality.

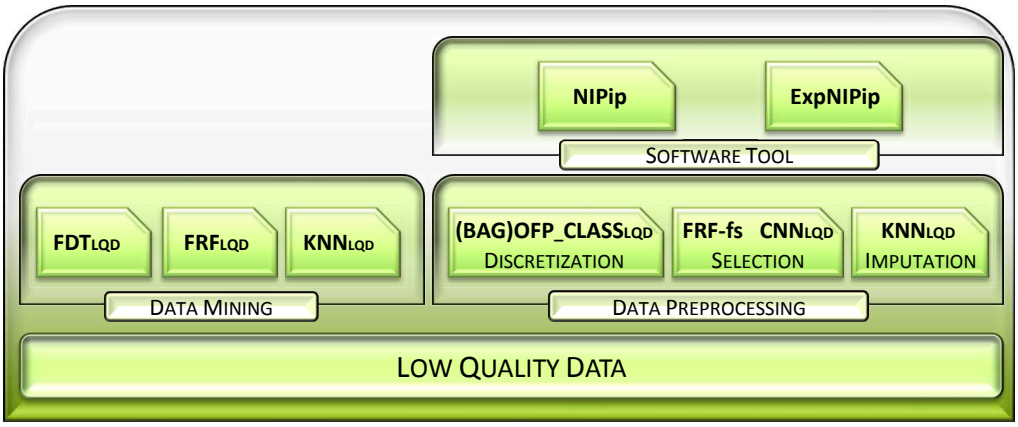


Figure 1: Scheme of treated tasks at work

Although the natural order would be to begin by detailing our proposals for the data preprocessing phase, we are going to begin by describing the proposed techniques for the data mining phase. Since these techniques will be needed later to evaluate the performance of the

preprocessing techniques and besides, some of the data preprocessing techniques use some of the components of the data mining techniques.

In the data mining phase, our proposals are focused on the extensions of three existing techniques, in order to provide them with the ability to deal with low quality data directly. The first technique extended is a fuzzy decision tree (FDT). The FDT used as a basis, needs a fuzzy partition of numerical attributes and is capable of handling missing values both in nominal attributes and in numerical attributes. In addition, it can also handle fuzzy attributes into numerical values as long as they are members of a partition of that attribute. Our proposal extends the functionality of this FDT to give it the ability to deal with imprecise classes, with fuzzy and interval values different from the initial fuzzy partition, with fuzzy subsets in numeric attributes and with fuzzy and crisp subsets in nominal attributes. The new extension of the decision tree have been called FDT_{LQD} and the main key of this extension is the definition of a similarity measure to calculate the membership degree of a low quality value to each of the descendants of a given node N.

Another technique, that has been extended, is the ensemble Fuzzy Random Forest (FRF). The ensemble FRF is composed of fuzzy decision trees (FDT's). To perform the extension of FRF (FRF_{LQD}) the fuzzy decision tree FDT_{LQD} has been used, so that FRF can work explicitly with low quality data, specifically the same types of data with which FDT_{LQD} works. Both FRF_{LQD} and FDT_{LQD} are evaluated by means of a series of experiments. In one of the experiments we have checked that a loss of information occurs when low quality data are transformed into crisp data, directly affecting the performance/accuracy of the results. In another experiment, data from real problems are used, that contain low quality data. The results obtained are compared with another classifier of literature that also supports low quality data. The results obtained are satisfactory and competitive. As with the experiments performed, we can conclude that the fuzzy decision tree FDT_{LQD} and the ensemble FRF_{LQD} are robust techniques and their behaviors are stable and competitive against other techniques of literature.

The last extended technique is the k-nearest neighbor technique. Specifically, we have extended the rule of k-nearest neighbors (KNN_{LQD}) so this technique can work with low quality data. As a main element of this extension, a measure to calculate the distance between examples with imperfect values is defined. More specifically, KNN_{LQD} can work with fuzzy and interval values,fuzzy subsets in numerical attributes and fuzzy and crisp subsets in nominal attributes.

Now that detailed proposals and advances in the data mining phase have been made, we are going to focus on the preprocessing data phase. In this phase a set of techniques have been proposed. This techniques include processes of discretization of numerical attributes, of attribute selection, of examples selection and of imputation of missing values.

With regard to the process of numerical attributes discretization, we have proposed an initial discretization technique and several improvements which perform the partitioning of numerical attributes by means of fuzzy partitions. This technique, called OFP_CLASS, is supervised, local, top-down and incremental. Futhermore OFP_CLASS uses entropy as a measure to obtain and evaluate the intervals. OFP_CLASS is a hybrid technique composed of two stages: in the first stage a fuzzy decision tree is used to obtain the possible cut points to divide the domain of an attribute; and in the second stage a genetic algorithm is used to optimize the number of cut points and to construct the final fuzzy partitions. The technique OFP_CLASS can work with nominal and numerical attributes and both attributes can contain missing values. To extend the types of data with which this technique can work, an improvement of this technique (OFP_-CLASS$_{\text{LQD}}$) has been proposed. OFP_CLASS$_{\text{LQD}}$ can work directly with fuzzy and interval values in numerical attributes and also allows an imprecise value class of the examples. Another improvement of the technique (BAGOFP_CLASS) is performed in order to deal with datasets where the number of examples is proportionately less than the number of attributes and classes. The essential part of this improvement is the introduction of bagging, both in the first stage of the technique and in the second stage. With the introduction of bagging, the quality of results are improved and a greater wealth of information in the two stages is obtained. Both the initial technique and the other improvements of the technique are evaluated through a series of experiments showing the robustness, the balance and the quality of the results when comparing with other techniques in literature and when comparing the improvements of this technique together.

Focusing on the process of attribute selection, a hybrid technique to select attributes from low quality data, called FRF-fs, is proposed. This technique is classified as a hybrid Filter-Wrapper technique with sequential forward search on the subset of attributes obtained by the technique of filtering and using the ranking obtained of such attributes. It is important to note that during the attribute selection process, not only does the technique provide the user with an optimal set of attributes, but each step also provides the user, with the information of the attributes that may be relevant according to the end user goal. In the first stage of the technique, an ensemble FRF$_{\text{LQD}}$ is used to obtain the information about the importance of the attributes. All this information collected after using the ensemble is then combined using an OWA operator in order to create an attribute importance ranking (a ranking which the user can obtain without having to proceed to the next stage of the technique FRF-fs). From the ranking obtained, in the second stage, a classifier is used to obtain the optimum set of attributes. It is important to take into account that both the ensemble FRF$_{\text{LQD}}$ used in the first stage, and the final classifier for the second stage, can deal with low quality data directly. The low quality data with which FRF-fs can work are: fuzzy and interval values in numerical attributes; missing

values both in numeric attributes and in nominal attributes; fuzzy subsets in numerical attributes and fuzzy and crisp subsets in nominal attributes. The FRF-fs technique has been evaluated using datasets of low quality and this technique has been compared with other techniques of literature, obtaining very competitive and satisfactory results that ensure the robustness and competitiveness of the technique both when working with low quality data and when working with crisp data.

To finish with the data preprocessing phase, a technique of predictive imputation of missing values and a technique to select examples CNN_{LQD} have been proposed to work explicitly with low quality data. The imputation technique is based on the technique of k-nearest neighbors KNN_{LQD}. The technique of imputation proposed can work with fuzzy and interval values and with fuzzy subsets in numerical attributes and with fuzzy and crisp subsets in nominal attributes. Furthermore, the technique includes an external parameter. This parameter expresses the degree of confidence with which the user wants to perform that imputation. Depending on the degree of trust imposed by the user, the imputation of a missing value can be performed or not. When the imputation of a missing value is performed, the imputed value does not have to be a crisp value, the imputed value may be of low quality, where the imperfection of it will depend on the rest of the low quality data containing in its nearest neighbors. Because of the fact that the technique of k-nearest neighbors has a high computational cost when working with datasets with many examples, a technique of selecting examples has been developed, specifically a technique of condensation, called CNN_{LQD}. CNN_{LQD} works with the same types of data that the imputation technique proposed, but CNN_{LQD} has the disadvantage that the results depend on the order of the examples in the dataset. As a measure of initial order, the fuzzy entropy of the class has been used to sort the examples of the dataset. This measure is based on the imperfection of the attribute class. It is important to note that although these techniques have been validated by means of a series of experiments, they are in their initial phase of development and they are subject to future improvements.

As a final objective of this work (see Figure 2), a software tool has been developed. This tool includes part of the data preprocessing techniques that we have proposed and developed throughout the work, . The objective of this software tool, called “NIP imperfection processor” (NIPip), is to provide a common framework where researchers can perform preprocessing on datasets either to add low quality data or to transform low quality data in other data types. The NIPip tool consists of two environments, the first being more interactive environment and the second being an environment that is more oriented to the experimentation. Among the main characteristics of the tool includ, we must highlight its flexibility to define the input and output data formats, where users can select a standard format or a custom format defined by themselves. The tool allows you to add low quality data of different types and in different

amounts to each of the attributes that make up an example. Among the low quality data that can be added, we can find fuzzy and interval values, fuzzy and crisp subsets, missing values and noise. The tool also contains an internal library of discretization techniques and imputation techniques for missing values. Imputation techniques allow the user to eliminate low quality data by means of replacing by crisp data, or transforming one type of low quality data into another type of low quality data.

To conclude, we must mention that all proposals and extended techniques have shown very satisfactory and optimistic behavior, both when working with low quality data and when working with crisp data. Although the results obtained can be classified as good results, there are still possible improvements to be made in each. The same goes for the software tool developed, which opens a very interesting line in the available tools in the field of Intelligent Data Analysis.

Resumen

El avance de las nuevas tecnologías ha permitido el almacenamiento de grandes volúmenes de información compuestos de diferentes tipos de datos, los cuales no siempre son tan precisos y perfectos como sería deseable. Cada vez es más frecuente encontrar datos imperfectos ya que habitualmente utilizamos cuantificadores vagos o imprecisos (bastante joven, muy mayor, muy pequeño, demasiado grande, etc.), frecuentemente se cometen errores en los instrumentos de medida utilizados al obtener dicha información, hay datos que faltan, etc.. Sin embargo, a pesar que la imperfección está presente de forma directa en los datos, todavía el número de técnicas dentro de la disciplina del Análisis Inteligente de Datos que permitan el tratamiento de forma explícita de este tipo de información es bastante escaso. De esta forma, si las técnicas no son capaces de tratar con datos imperfectos, estos datos son transformados en precisos y durante este proceso de transformación es posible que se produzca una pérdida de información relevante que afecte de forma directa a la calidad de los resultados esperados.

Ante la problemática de la escasez de técnicas que trabajen de forma directa con datos imperfectos, nos planteamos desarrollar en este trabajo técnicas dentro del Análisis Inteligente de Datos que permitan el tratamiento de conjuntos de datos que puedan contener de forma explícita datos imperfectos. Para llevar a cabo esta propuesta vamos a combinar las metodologías que nos ofrece el Softcomputing con las técnicas del Análisis Inteligente de Datos con el fin de, por un lado, extender algunas de estas técnicas para añadirles la capacidad de trabajar de forma directa con datos imperfectos y, por otro lado, diseñar nuevas técnicas que sean capaces de trabajar explícitamente con este tipo de datos.

Como ya hemos comentado, en una amplia variedad de áreas y campos se han ido coleccionando y acumulando datos a un ritmo vertiginoso, de ahí la creciente necesidad de una nueva generación computacional de teorías, herramientas y técnicas que ayuden a extraer conocimiento útil de forma rápida a partir de los grandes volúmenes de información. Esta nueva generación computacional se encuentra bajo el campo del Análisis Inteligente de Datos. En un cierto nivel de abstracción, el área del Análisis Inteligente de Datos manifiesta su preocupación por el desarrollo de técnicas que proporcionan un sentido a los datos coleccionados. Sin embargo, el problema básico que aborda el proceso del Análisis Inteligente de Datos es el mapeo de datos de más bajo nivel en otras formas que pueden ser más compactas, más abstractas o

más útiles. Así, el Análisis Inteligente de Datos se define como un proceso complejo de descubrimiento de información o conocimiento útil a partir de los datos, donde éstos componen la parte más importante de la disciplina, la cual se divide en varias fases:

- Integración y recopilación de datos
- Preprocesamiento de datos
- Minería de datos
- Evaluación e interpretación
- Difusión y uso de modelos

Dado que los datos constituyen la materia prima para el proceso del Análisis Inteligente de Datos y éstos pueden ser imperfectos, debemos incorporar al proceso diversas metodologías que sean tolerantes a la imprecisión, a la incertidumbre y a la verdad parcial. Estas metodologías han sido acuñadas con el término de Softcomputing. El término Softcomputing puede definirse como una serie de metodologías que permiten tratar las situaciones prácticas reales de la misma forma que suelen hacerlo los seres humanos, es decir, en base a inteligencia, sentido común, consideración de analogías, aproximaciones, etc. El Softcomputing queda situado como la base teórica del área de los Sistemas Inteligentes, y hace patente que la diferencia entre el área de la Inteligencia Artificial clásica, y la de los Sistemas Inteligentes, es que la primera se apoya en la denominada Hardcomputing, mientras que la segunda lo hace en el Softcomputing.

En este trabajo nos hemos centrado en las fases de preprocesamiento y de minería de datos del Análisis Inteligente de Datos para, tal y como hemos comentado, extender técnicas ya existentes y diseñar nuevas técnicas dentro de estas dos fases con el objetivo de que ellas puedan trabajar con datos imperfectos (datos de baja calidad) de forma directa, sin necesidad de una transformación previa. En la Figura 2 mostramos el esquema general del trabajo realizado en el ámbito del Análisis Inteligente de Datos sobre datos de baja calidad.

Aunque el orden natural sería comenzar detallando nuestras propuestas para la fase de preprocesamiento de datos, vamos a comenzar describiendo las técnicas propuestas para la fase de minería de datos, puesto que estas técnicas después serán necesarias para poder evaluar el comportamiento de las técnicas de preprocesamiento y, además, algunas de las técnicas de preprocesamiento de datos utilizan parte de las componentes de las técnicas de minería de datos.

En la fase de minería de datos, nuestras propuestas se centran en la extensión de tres técnicas ya existentes, con el fin de añadirles la capacidad de tratar datos de baja calidad de forma directa. La primera de las técnicas que extendemos es un árbol de decisión fuzzy (FDT). El

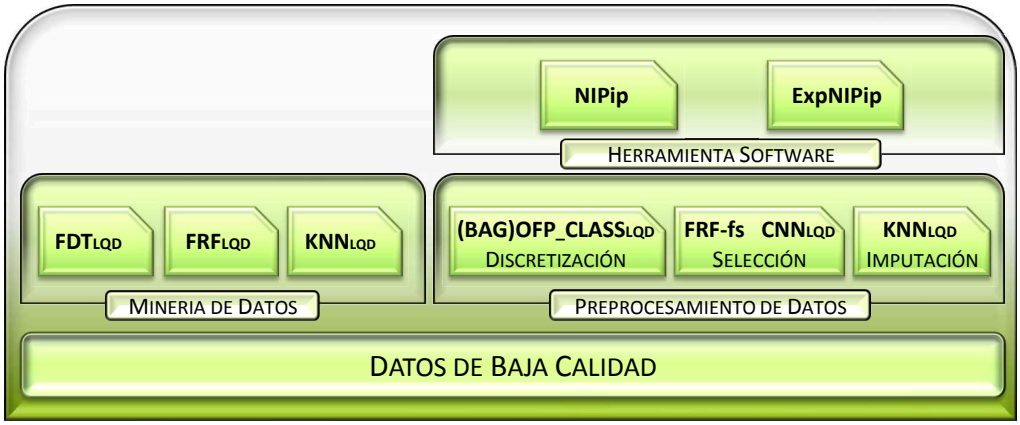


Figura 2: Esquema de las tareas tratadas en el trabajo

FDT que tomamos como base necesita una partición fuzzy de los atributos numéricos y es capaz de trabajar con valores missing tanto en atributos nominales como en atributos numéricos. Además, también puede manejar valores fuzzy en atributos numéricos siempre que éstos sean pertenecientes a una partición/discretización de dicho atributo. Nuestra propuesta extiende la funcionalidad de este FDT para darle la capacidad de poder tratar con clases imprecisas, con valores fuzzy e intervalares diferentes a los de la partición fuzzy inicial, con subconjuntos de valores fuzzy en atributos numéricos y con subconjuntos de valores fuzzy y crisp en atributos nominales. La nueva extensión del árbol la hemos denominado FDT_{LQD} y la clave principal de esta extensión consiste en la definición de una medida de similitud para calcular el grado de pertenencia de un valor de baja calidad a cada uno de los descendientes de un nodo N dado.

Otra de las técnicas que hemos extendido es el ensamble Fuzzy Random Forest (FRF). El ensamble FRF se compone de árboles de decisión fuzzy (FDT's). Para llevar a cabo la extensión de FRF (FRF_{LQD}) hemos utilizado el árbol de decisión fuzzy FDT_{LQD}, para que así pueda trabajar de forma explícita con datos de baja calidad, específicamente los mismos tipos de datos con los cuales trabaja FDT_{LQD}. Tanto FRF_{LQD} como FDT_{LQD} son evaluados mediante una serie de experimentos. En uno de los experimentos hemos comprobado que se produce una pérdida de información cuando los datos de baja calidad son transformados a datos crisp afectando directamente al rendimiento/precisión de los resultados. En otro de los experimentos se utilizan datos de problemas reales, los cuales contienen datos de baja calidad, y se comparan los resultados obtenidos con otro clasificador de la literatura que también soporta datos de baja calidad, obteniéndose unos resultados satisfactorios y competitivos. Con los experimentos realizados, podemos concluir que el árbol FDT_{LQD} y el ensamble FRF_{LQD} son robustos y tienen un comportamiento estable y competitivo frente a otras técnicas presentes en la literatura.

La última técnica extendida es la técnica k-vecinos más cercanos. En concreto, hemos extendido la regla de k-vecinos más cercanos (KNN_{LQD}) para que pueda trabajar con datos de baja calidad. Como principal elemento de esta extensión se define una medida para calcular la distancia entre ejemplos con valores imperfectos. Más concretamente, KNN_{LQD} permite trabajar con valores fuzzy, valores intervalares, subconjuntos de valores fuzzy en atributos numéricos y con subconjuntos de valores fuzzy y crisp en atributos nominales.

Una vez detalladas las propuestas y avances en la fase de minería de datos, nos centramos en la fase de preprocesamiento. En esta fase hemos propuesto un conjunto de técnicas que abarcan los procesos de discretización de atributos numéricos, la selección de atributos, la selección de ejemplos y la imputación de valores de missing.

Respecto al proceso de discretización de atributos numéricos, hemos propuesto una técnica inicial de discretización y varias mejoras las cuales llevan a cabo el particionamiento de los atributos numéricos mediante particiones fuzzy. Esta técnica, denominada OFP_CLASS, es supervisada, local, top down e incremental, además de usar la entropía como medida para obtener y evaluar los intervalos. OFP_CLASS es una técnica híbrida compuesta de dos etapas: En la primera etapa se hace uso de un árbol de decisión fuzzy que obtiene los posibles puntos de corte para dividir el dominio de un atributo; y en la segunda etapa se utiliza un algoritmo genético para optimizar el número de puntos de corte y construir las particiones fuzzy finales. La técnica OFP_CLASS puede trabajar con atributos nominales y numéricos y ambos atributos pueden contener valores missing. Para ampliar los tipos de datos con los cuales esta técnica puede trabajar se ha propuesto una mejora de la técnica (OFP_CLASS_{LQD}) que permite trabajar de forma directa con valores intervalares y valores fuzzy en los atributos numéricos y permite además que el valor de clase de los ejemplos sea impreciso. Otra mejora de la técnica (BAGOFP_CLASS) se lleva a cabo con el objetivo de tratar con conjuntos de datos donde el número de ejemplos es proporcionalmente inferior al número de atributos y de clases. La parte esencial de esta mejora es la introducción de bagging tanto en la primera etapa de la técnica como en la segunda etapa. Con la introducción de bagging mejoramos la calidad de los resultados ya que obtenemos una mayor riqueza en información en las dos etapas. Tanto la técnica inicial como el resto de mejoras son evaluadas mediante una serie de experimentos que muestran la robustez, el equilibrio y la calidad de los resultados comparando con otras técnicas de la literatura y comparando las mejoras de la técnica entre sí.

Centrándonos en el proceso de selección de atributos, hemos propuesto una técnica híbrida para seleccionar atributos desde datos de baja calidad llamada FRF-fs. Esta se clasifica como técnica híbrida Filtro-Wrapper con búsqueda secuencial hacia delante sobre el subconjunto obtenido por la técnica de filtrado y utilizando el ranking obtenido de estos atributos. Es importante tener en cuenta que durante todo el proceso de selección de atributos, la técnica no solo

proporciona al usuario un conjunto óptimo de atributos, sino que en cada etapa proporciona al usuario información de los atributos que puede ser relevante según el objetivo final del usuario. En la primera etapa de la técnica, se utiliza un ensamble FRF_{LQD} para obtener información acerca de la importancia de los atributos. Toda la información recogida tras utilizar el ensamble se combina haciendo uso de un operador OWA con el fin de crear un ranking de importancia de los atributos (ranking que el usuario puede obtener sin necesidad de seguir con la siguiente etapa de la técnica FRF-fs). A partir del ranking obtenido, en la segunda etapa se utiliza un clasificador para obtener el conjunto óptimo de atributos. Es importante resaltar que tanto el ensamble FRF_{LQD} utilizado en la primera etapa como el clasificador final para la segunda pueden tratar con datos de baja calidad de forma directa. Los datos de baja calidad con los que FRF-fs puede trabajar son valores fuzzy y valores intervalares en atributos numéricos, valores missing tanto en atributos numéricos como en atributos nominales, subconjuntos de valores fuzzy en atributos numéricos y subconjuntos de valores fuzzy y crisp en atributos nominales. La técnica FRF-fs ha sido evaluada utilizando conjuntos de datos de baja calidad y comparada con otras técnicas de la literatura, obteniendo resultados bastante competitivos y satisfactorios que aseguran la robustez y competitividad de la técnica tanto cuando trabaja con datos de baja calidad como cuando trabaja con datos crisp.

Para terminar con la fase de preprocesamiento de los datos, hemos propuesto una técnica de imputación predictiva de valores missing basada en la técnica de k-vecinos más cercanos KNN_{LQD} y una técnica de selección de ejemplos, CNN_{LQD}, que son capaces de trabajar de forma explícita con datos de baja calidad. La técnica de imputación propuesta es capaz de trabajar con valores intervalares, con valores fuzzy, con subconjuntos fuzzy en atributos numéricos y con subconjuntos fuzzy y crisp en atributos nominales. Además, la técnica incluye un parámetro externo donde el usuario expresa el grado de confianza con el que quiere que se lleva a cabo la imputación. Según el grado de confianza impuesto por el usuario la imputación de un valor missing podrá llevarse a cabo o no. Cuando se lleva a cabo la imputación de un valor missing, el valor imputado no tiene porqué ser un valor crisp, sino que podrá ser un valor de baja calidad, donde la imperfección del mismo dependerá del resto de datos de baja calidad que contengan sus vecinos más cercanos. Debido a que la técnica de k-vecinos más cercanos tiene un coste elevado de procesamiento cuando trabaja con conjuntos de datos con muchos ejemplos, hemos desarrollado una técnica de selección de ejemplos, concretamente una técnica de condensación, que hemos denominado CNN_{LQD}. CNN_{LQD} trabaja con los mismos tipos de datos que la técnica de imputación propuesta, pero tiene el inconveniente que los resultados dependen del orden de los ejemplos en el conjunto de datos. Como una medida de orden inicial, hemos utilizado la entropía fuzzy de la clase para ordenar los ejemplos de un conjunto de datos en función de la imperfección de su atributo clase. Es importante tener en cuenta que a pesar

de que estas técnicas han sido validadas mediante una serie de experimentos, se encuentran en su fase inicial de desarrollo y sujetas a diversas mejoras futuras.

Como último objetivo de este trabajo (véase la Figura 2), hemos desarrollado una herramienta software que recoge parte de las técnicas de preprocesamiento de datos que hemos propuesto y desarrollado a lo largo del trabajo. El objetivo de esta herramienta software, llamada "NIP imperfection processor" (NIPip), es proporcionar un marco de trabajo común donde los investigadores puedan llevar a cabo un preprocesamiento sobre conjuntos de datos bien para añadirles datos de baja calidad o bien para transformar dicho datos de baja calidad en otros tipos de datos. La herramienta NIPip se compone de dos entornos, un primer entorno más interactivo y un segundo entorno más orientado a la experimentación. Entre las características principales de la herramienta destacamos su flexibilidad a la hora de definir los formatos de entrada y salida de los conjuntos de datos pudiendo seleccionar el usuario un formato estándar o un formato propio definido por él mismo. La herramienta permite añadir datos de baja calidad de diferente tipo y en cantidades diferentes a cada uno de los atributos que componen un ejemplo. Entre los datos de baja calidad a añadir nos encontramos valores fuzzy, intervalos, subconjuntos fuzzy y crisp, valores missing y ruido. Además, la herramienta contiene una librería interna de técnicas de discretización y de técnicas de imputación de valores missing. Las técnicas de imputación permiten eliminar los datos de baja calidad sustituyéndolos por datos crisp o transformar un tipo de dato de baja calidad en otro tipo de dato de baja calidad.

Para concluir, debemos de comentar que todas las técnicas propuestas y extendidas han mostrado un comportamiento muy satisfactorio y optimista tanto cuando trabajan con datos de baja calidad como cuando trabajan con datos crisp. A pesar de que los resultados obtenidos los podemos catalogar de buenos resultados, todavía quedan posibles mejoras por realizar en cada una de ellas. Lo mismo ocurre con la herramienta software desarrollada con la cual se abre una línea muy interesante en las herramientas disponibles en el campo del Análisis Inteligente de Datos.

Índice de Tablas

Índice de Figuras

Índice de Algoritmos

CAPÍTULO

1

Introducción

1.1 Motivación

Las nuevas tecnologías de ordenadores, redes y sensores han hecho de la recolección y organización de datos una tarea casi sin esfuerzo. Sin embargo, los datos capturados necesitan ser transformados en información y conocimiento útil. El valor de los datos se basa en la habilidad para extraer información útil durante la toma de decisiones o durante la exploración, y en la comprensión del fenómeno gobernante en la fuente de datos.

En muchos dominios, el análisis de datos fue tradicionalmente un proceso manual. Uno o más analistas familiarizados con los datos, con la ayuda de técnicas estadísticas, proporcionaban resúmenes y generaban informes. En efecto, el analista hacía de procesador de preguntas sofisticado. Sin embargo, tal enfoque cambió como consecuencia del crecimiento del volumen de datos en una multitud de dominios tales como: librerías digitales, archivos de imágenes, bioinformática, cuidados médicos, finanzas e inversión, fabricación y producción, negocios y marketing, redes de telecomunicación, dominios científicos, la biométrica, etc.. Cuando la escala de manipulación de datos, exploración e inferencia va más allá de la capacidad humana, se necesita la ayuda de las tecnologías computacionales para automatizar el proceso. De ahí, que todos los dominios involucrados necesitan de las metodologías que ofrece el análisis inteligente de datos.

Durante el proceso de recopilación de datos y antes de pasar a su análisis existen muchas posibles razones por las cuales las fuentes de adquisición de datos pueden no ser fiables debido por ejemplo a sensores defectuosos, errores en la recopilación de datos, falta de normas de representación en los datos, etc.. Por lo tanto, esto puede provocar la obtención de datos que

no son todo lo perfecto que sería deseable. Aunque generalmente estos factores y limitaciones son ampliamente aceptados por los investigadores, la mayoría de las aplicaciones han ignorado tradicionalmente la necesidad de desarrollar enfoques apropiados para la representación y el razonamiento con tales imperfecciones en los datos. A veces, estas imperfecciones en los datos han sido ignoradas. Sin embargo, cada vez aparecen con más frecuencia y las técnicas del análisis inteligente de datos deben de adaptarse para poder trabajar con tales datos. Para desarrollar tal adaptación de las técnicas, podemos hacer uso de las metodologías que nos ofrece el Softcomputing, puesto que la computación tradicional supone que los datos son precisos pero, atendiendo a la forma de razonar humana y a los problemas presentes hoy en día, es más adecuado considerar que hay vaguedad e imprecisión en los datos. Por esta razón y como vamos a exponer en la siguiente sección, nuestro objetivo es adaptar y proponer nuevas técnicas dentro del análisis inteligente de datos para que sean capaces de trabajar eficaz y eficientemente con datos imperfectos.

1.2 Definición problema

La revolución en el crecimiento de las tecnologías de la información ha dado como resultado el uso de los ordenadores en un amplio rango de aplicaciones tales como aplicaciones científicas, de negocios, geográficas, de ingeniería de diseño, etc.. La variedad y continuado uso de los sistemas de información ha proporcionado una enorme cantidad de datos a disposición de los usuarios. Los datos almacenados pueden provenir de diferentes fuentes de información y cada vez es más frecuente que dichos datos contengan información no siempre con tanta precisión como el usuario desearía. Cuando la imperfección contenida en los datos es insignificante, normalmente es eliminada, ya que es ampliamente aceptado por los investigadores que la cantidad de información que se pierde al eliminarla es muy pequeña. Sin embargo, la imperfección en los datos es cada día más frecuente y por consecuencia el eliminarla supondría la perdida de información relevante. De ahí que las diferentes formas de imperfección inherentes en los problemas del mundo real necesiten ser manejadas adecuadamente.

Para manejar esta información, por un lado, disponemos de un conjunto de metolodogías ofrecidas por el Softcomputing para tratar la imprecisión e incertidumbre de manera adecuada, con el objetivo de alcanzar soluciones razonables a bajo coste. Por otro lado, dentro del análisis inteligente de datos tenemos la fase de minería de datos, la cual nos facilita una gran variedad de técnicas para estudiar, analizar y obtener resultados de los diferentes problemas que se presentan en la vida real.

El problema surge cuando, aunque en la literatura existen muchas técnicas de minería de datos, la mayoría de estas técnicas no pueden tratar con información imperfecta explícitamente

en los datos. Por lo tanto, cuando estas técnicas se encuentran con este tipo de información en los datos, la tendencia es transformar los datos para eliminar la imperfección. Sin embargo, esta transformación lleva asociada en la mayoría de los casos la perdida de información relevante, perdida de información que afecta posteriormente a los resultados obtenidos. Debido a esto, nuestro objetivo es poder trabajar directamente con datos imperfectos y para ello incorporaremos las metodologías del Softcomputing en el Análisis Inteligente de Datos. De esta forma, a lo largo de este trabajo vamos a proponer un conjunto de metodologías, técnicas y funciones que aplicadas a técnicas ya existentes en la literatura o bien aplicadas a nuevas propuestas nos van a permitir trabajar con la información imperfecta que aparece de forma explícita en los datos.

El conjunto de metodologías, técnicas y funciones que proponemos no solo las vamos a englobar en la fase de minería de datos dentro del Análisis Inteligente de Datos, sino también en la fase previa a la minería de datos, la fase de preprocesamiento de los datos, puesto que esta fase tiene como principal objetivo adecuar los datos para que las técnicas de minería de datos aplicadas posteriormente puedan extraer la máxima información y puedan obtener unos resultados lo más competitivos posibles. Así, el trabajo que vamos a desarrollar queda centrado en el área del Análisis Inteligente de Datos y el Softcomputing.

En la siguiente sección vamos a describir un poco más en detalle los objetivos que nos hemos planteado desarrollar en este trabajo.

1.3 Objetivos propuestos

Los objetivos que nos planteamos siempre se encuentran centrados en el desarrollo de técnicas del Análisis Inteligente de datos que hagan uso de metodologías del Softcomputing para trabajar con datos imperfectos. El objetivo global de la tesis, es proponer un conjunto de técnicas del análisis inteligente de datos, que tienen la capacidad de tratar datos imperfectos, datos de baja calidad. Para cumplir tal objetivo, este trabajo lo hemos divido en tres partes, las cuales se pueden observar gráficamente en la Figura 1.1.

Como se aprecia en la Figura 1.1, tomando siempre como base los datos de baja calidad, centramos el trabajo en las fases de minería de datos y de preprocesamiento de datos del Análisis Inteligente de Datos. Así, aplicando las metodologías que nos ofrece el Softcomputing, vamos a realizar la propuesta de nuevas técnicas y la extensión de otras ya existentes de tales fases. Además, cada una de las técnicas que proponemos las vamos a evaluar y validar con el fin de analizar la robustez y el comportamiento de las mismas con conjuntos de datos con y sin datos de baja calidad. A continuación vamos a describir más en detalle los objetivos es-

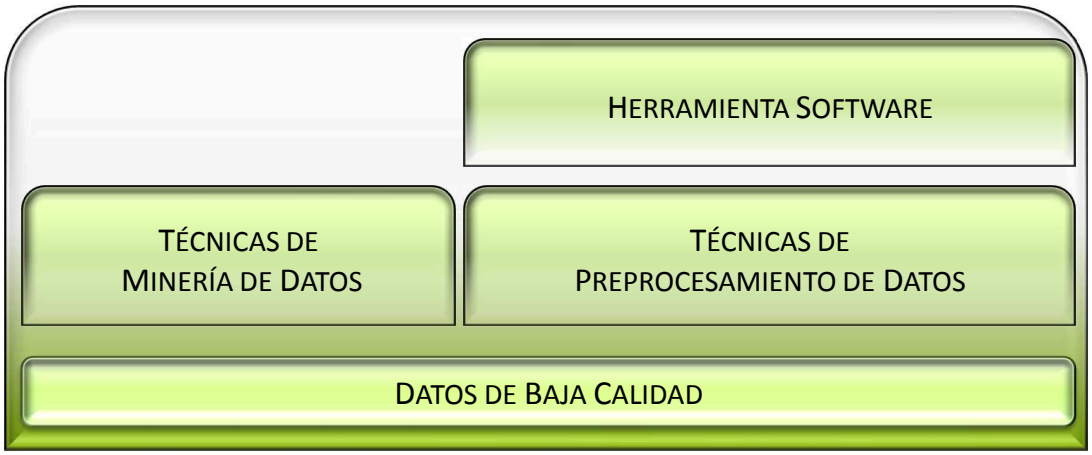


Figura 1.1: Esquema general de la Tesis

pecíficos que nos hemos propuesto y que vamos a desarrollar en los siguientes capítulos de este documento:

- Tras centrar el problema en el Análisis Inteligente de Datos y el Softcomputing, primero vamos a llevar a cabo una descripción de los datos que pueden considerarse de baja calidad y cómo las técnicas suelen trabajar con este tipo de datos. Después nos proponemos realizar un estudio de las fases de preprocesamiento y de minería de datos del Análisis Inteligente de Datos. Durante el estudio de estas dos fases, vamos a detallar sin ser exhaustivos, algunas de las técnicas que abordan cada una de estas dos fases, centrándonos en aquellas que trabajen en algún grado con datos de baja calidad.

- Comenzando por la fase de minería de datos, nuestro objetivo es extender un conjunto de técnicas de minería de datos a fin de que estas puedan trabajar de forma directa con datos de baja calidad. Las tres técnicas que extenderemos son un árbol de decisión fuzzy, un ensamble fuzzy random forest y la regla de k-vecinos más cercanos. Para cada una de estas técnicas proponemos las modificaciones necesarias para que puedan trabajar de forma explícita con datos de baja calidad. Para evaluar y valorar el comportamiento de dichas técnicas vamos a llevar a cabo ciertos experimentos con diferentes conjuntos de datos.

- Otro de nuestros objetivos es diseñar nuevas técnicas dentro de la fase de preprocesamiento de datos para que puedan trabajar con información de baja calidad. La fase de preprocesamiento de datos abarca varios procesos que tienen como objetivo mejorar la calidad de los datos. En nuestro caso nos vamos a centrar en los procesos de discretización de atributos numéricos, la selección de atributos y ejemplos y la imputación de

valores missing. Para cada uno de estos procesos proponemos una serie de algoritmos con el fin de poder desarrollar técnicas que llevan a cabo tales procesos de mejora de los datos. Para evaluar la calidad, la robustez y el comportamiento de dichas técnicas vamos a hacer uso de las técnicas de minería de datos propuestas.

- El último objetivo de este trabajo es crear una herramienta software, centrada en el preprocesamiento de datos de baja calidad. Las funciones principales de esta herramienta de preprocesamiento de datos serán las de tratar con ellos, bien transformándolos cuando ciertas técnicas no pueden trabajar con ellos o bien añadiéndolos con el fin de poder evaluar técnicas que tratan con tales datos dada la escasez de conjuntos de datos con imperfección explícita que existen en la actualidad. Esta herramienta como veremos está compuesta de dos entornos, uno más interactivo y otro más orientado a la experimentación. Así, con esta herramienta pretendemos complementar el número de herramientas existentes en la literatura centradas en la fase de preprocesamiento de datos.

1.4 Organización de la memoria

El presente documento está organizado en cuatro partes y cada parte de encuentra dividida en capítulos. Antes de comenzar cada una de las partes se realiza una pequeña descripción acerca del contenido de cada parte. A continuación vamos a describir de forma breve el contenido de cada una de las partes.

La Parte I, titulada “Estado del Conocimiento”, está formada por cuatro capítulos. En el Capítulo 2, realizamos una descripción de los conceptos de Análisis Inteligente de Datos y de Softcomputing, ya que este trabajo toma como base esencial tales conceptos. En el siguiente capítulo, Capítulo 3, presentamos los tipos de datos de baja calidad que nos podemos encontrar en los conjuntos de datos y cómo las técnicas manejan de forma general tales datos. Como últimos capítulos de esta parte, los Capítulos 4 y 5 presentan dos fases del Análisis Inteligente de Datos, el preprocesamiento de datos y la minería de datos respectivamente. Como ya hemos comentado, nuestra propuesta es presentar nuevas técnicas o extensiones de técnicas ya existentes en la literatura con el fin de que puedan tratar de forma directa con los datos de baja calidad.

La Parte II se titula “Minería de datos de baja calidad”. Esta parte está formada por cuatro capítulos donde en los tres primeros capítulos describimos una técnica de minería de datos por capítulo. En el Capítulo 6, presentamos un árbol de árbol de decisión fuzzy y su correspondiente extensión para trabajar con datos de baja calidad. En el Capítulo 7 presentamos el ensamble Fuzzy Random Forest, junto con las modificaciones necesarias en tal ensamble para añadirle la capacidad de tratar datos de baja calidad. En el último capítulo de esta Parte II, Capítulo 8,

presentamos la técnica K-vecinos más cercanos, junto con las modificaciones necesarias para que pueda trabajar con conjuntos de datos de baja calidad. Para finalizar esta parte, se presenta en el Capítulo 9, las conclusiones parciales de esta parte, destacando los principales resultados alcanzados y las aportaciones más relevantes relacionadas con esta parte del trabajo.

La tercera parte de este documento, Parte III titulada "Preprocesamiento de datos de baja calidad", también se encuentra dividida en cuatro capítulos. En los tres primeros capítulos presentamos un conjunto de técnicas centradas en la fase de preprocesamiento de los datos, concretamente nos centramos en el proceso de discretización de atributos numéricos, de selección de atributos, de selección de ejemplos y de imputación de valores missing. En el primer capítulo de esta parte, Capítulo 10, proponemos una técnica de discretización de atributos numéricos, con sus correspondientes extensiones y mejoras para tratar datos de baja calidad. Después en el Capítulo 11, presentamos una técnica de selección de atributos que trabaja con información de baja calidad. Por último, en el Capítulo 12, describimos una técnica de imputación predictiva de valores missing, basada en la técnica k-vecinos más cercanos y una técnica de selección de ejemplos, ambas trabajando con datos de baja calidad. Es importante destacar que las técnicas presentadas en esta Parte III, son evaluadas y analizadas utilizando las técnicas de minería de datos propuestas en la Parte II. Debido a esto, hemos alterado el orden natural de presentación de las técnicas, presentando la fase de minería de datos antes que la fase de preprocesamiento de datos. En el último capítulo de esta parte, Capítulo 13, se presentan las concusiones parciales de esta parte, junto con las publicaciones más relevantes relacionadas con la misma.

La Parte IV, titulada "Una herramienta software para el trabajo con datos de baja calidad", se compone de dos capítulos. En el Capítulo 14 se describe una herramienta software centrada en la fase de preprocesamiento de datos de baja calidad, que está compuesta de dos entornos, uno más interactivo y visual y otro entorno más centrado en el ámbito de la investigación y de la experimentación. Durante este capítulo de descripción de la herramienta llevamos a cabo una comparativa con otras herramientas y destacamos la falta de capacidad de estas herramientas para tratar de forma directa datos de baja calidad. En el último capítulo de esta parte, Capítulo 15, se detallan las conclusiones parciales de esta parte y las aportaciones más relevantes relacionadas con la herramienta software presentada.

Por último y para finalizar el documento, presentamos las conclusiones generales del trabajo y los trabajos futuros a desarrollar.

Parte I

ESTADO DEL CONOCIMIENTO

CAPÍTULO

2

Análisis Inteligente de Datos y Softcomputing

2.1 Introducción

La revolución digital ha conseguido que la información digitalizada sea más fácil de capturar y bastante más barata de almacenar, [73, 150]. El aumento del volumen y de la variedad de información ha hecho surgir la necesidad de una nueva generación de herramientas y técnicas para soportar la extracción de conocimiento a partir de la información disponible. Actualmente, la disciplina que se encarga de la obtención de este conocimiento útil a partir de los datos es "El Análisis Inteligente de Datos". En [73] se define el Análisis Inteligente de Datos, denotado a partir de ahora como AID, como "el proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de los datos". Como se deduce de esta definición anterior, el AID es un proceso complejo que incluye no sólo la obtención de los modelos que capturan el conocimiento, sino también la evaluación y posible interpretación de los mismos. En esta disciplina, los datos son un conjunto de hechos (por ejemplo, los casos en una base de datos), y el patrón es una expresión en algún lenguaje que describe un subconjunto de los datos o un modelo aplicable al subconjunto. Los patrones descubiertos deben de ser válidos para los nuevos datos con algún grado de certeza y además deben de ser potencialmente útiles para conseguir obtener un beneficio para el usuario o para la tarea que desempeñan. Por último, los patrones deben de ser comprensibles si no inmediatamente, en algún tipo de postprocesamiento posterior.

En el AID, nos encontramos que los datos componen la parte más importante de la disciplina, la cual se divide en varias etapas que se van a describir más adelante. Centrándonos en los datos y antes de aplicar cualquier etapa de la disciplina del AID debemos de tener en cuenta la naturaleza de éstos para asegurar el éxito al finalizar el proceso del AID. Esto significa que, dependiendo de la naturaleza y de la precisión con la que se hayan obtenido esos datos, las técnicas a aplicar serán diferentes dependiendo por ejemplo del grado de tolerancia de las técnicas a la imprecisión e incertidumbre. Un ejemplo claro para ilustrar el problema de la diferente naturaleza de los datos es el problema de aparcar un coche [217, 218]. La mayoría de la población es capaz de aparcar un coche fácilmente ya que la posición y la orientación del vehículo nunca es especificada de forma precisa. Sin embargo, si la posición y la orientación del coche nos fuera especificada de forma precisa, la dificultad de aparcar crecería geométricamente con el incremento en precisión y eventualmente llegaría a ser inmanejable para los humanos. De aquí es importante observar que el problema del aparcamiento es fácil para los humanos cuando es formulado de forma imprecisa, pero sería difícil de resolver mediante una técnica tradicional ya que tales técnicas no explotan la tolerancia a la imprecisión.

Para tolerar la imperfección de los datos y describir el uso simbiótico de muchas disciplinas computacionales emergentes, en los años 90 se acuñó el término Softcomputing, [20]. Las tecnologías bajo Softcomputing son tolerantes a la imprecisión, a la incertidumbre y a la verdad parcial. Bajo la explotación de dicha tolerancia subyace la remarcable habilidad de los humanos para entender los discursos distorsionados, para descifrar la letra descuidada, para comprender los matices del lenguaje natural, etc.. El Softcomputing utiliza la mente humana como un modelo a seguir y al mismo tiempo tiene como objetivo la formalización de los procesos cognitivos humanos que empleamos tan eficazmente para llevar a cabo las tareas diarias, [217].

Tras un primera visión sobre los conceptos del AID y del Softcomputing, en las siguientes secciones vamos a profundizar en tales conceptos, detallando términos como Fuzzy Set o variables lingüísticas ligados al Softcomputing, las fases que componen el AID y las funciones que cumplen dichas fases en tal análisis. Por último, vamos a definir el motivo y objetivo de esta tesis que es mezclar los procesos definidos en el AID con las metodologías que nos ofrece el Softcomputing con el fin de flexibilizar las técnicas del AID para que puedan tolerar de forma directa en los conjuntos de datos la imperfección de los mismos, puesto que el razonamiento humano y los problemas del mundo real tratan de forma natural con este tipo de datos.

2.2 Softcomputing

La primera definición que se aportó del término Softcomputing fue la siguiente:

"Básicamente, Softcomputing no es un cuerpo homogéneo de conceptos y técnicas. Más bien es una mezcla de distintas técnicas que de una forma u otra cooperan desde sus fundamentos. En este sentido, el principal objetivo del Softcomputing es aprovechar la tolerancia que conllevan la imprecisión y la incertidumbre, para conseguir manejabilidad, robustez y soluciones de bajo costo. Los principales ingredientes del Softcomputing son la Lógica Fuzzy, la Neurocomputación y el Razonamiento Probabilístico, incluyendo este último a los Algoritmos Genéticos, las Redes de Creencia, los Sistemas Caóticos y algunas partes de la Teoría de Aprendizaje. En esa asociación de Lógica Fuzzy, Neurocomputación y Razonamiento Probabilístico, la Lógica Fuzzy se ocupa principalmente de la imprecisión y el Razonamiento Aproximado; la Neurocomputación del aprendizaje, y el Razonamiento Probabilístico de la incertidumbre y la propagación de las creencias", [218]. En otras palabras el término Softcomputing hace referencia a una colección de metodologías que tiene por objetivo explotar la tolerancia a la imprecisión e incertidumbre para lograr la manejabilidad, la robustez y las soluciones a bajo costo. El modelo a seguir por el Softcomputing es la mente humana.

Tras esta definición, está claro que la primera definición de Softcomputing en lugar de ser una definición precisa, es una definición por extensión por medio de diferentes conceptos y técnicas intentando sobreponerse a las dificultades que se presentan en los problemas reales de un mundo impreciso, incierto y difícil de categorizar, [192]. Al no ser una definición precisa, ha habido varios intentos para ajustar esta definición con diferentes resultados. Entre las definiciones más adecuadas nos encontramos la presentada en [127], donde cada proceso computacional que incluye a propósito imprecisión dentro de algún cálculo en uno o más niveles y que permite a esta imprecisión o cambiar (decrecer) la granularidad del problema o "suavizar" la meta de optimización en algunas etapas, es considerado como perteneciente al campo del Softcomputing.

Otra definición y punto de vista respecto al término Softcomputing es considerado en [192], donde definen Softcomputing como una antítesis de lo que se llama hard computing. Este punto de vista es consistente con el propuesto por Zadeh en [218]. Por tanto, el Softcomputing puede ser visto como una serie de técnicas y métodos que permiten tratar las situaciones prácticas reales de la misma forma que suelen hacerlo los seres humanos, es decir, en base a inteligencia, sentido común, consideración de analogías, aproximaciones, etc.. En este sentido el Softcomputing es una familia de técnicas de resolución de problemas cuyos primeros miembros serían el Razonamiento Aproximado y los Métodos de Aproximación Funcional y de Optimización, incluyendo los de búsqueda. Por lo tanto, el Softcomputing queda situado como la base teórica del área de los Sistemas Inteligentes, y se hace patente que la diferencia entre el área de la Inteligencia Artificial clásica, y la de los Sistemas Inteligentes, es que la primera se apoya en la denominada Hard Computing, mientras que la segunda lo hace en el Softcomputing.

Desde este último punto de vista, en un segundo nivel, el Softcomputing se puede desgranar entonces en otros componentes que ayudan a una definición por extensión, como la que se dio en primer lugar. Desde el principio, se han considerado que en ese segundo nivel los componentes más importantes son el Razonamiento Probabilístico, la Lógica y los Conjuntos Fuzzy, las Redes Neuronales y los Algoritmos Genéticos, [20], que debido a su alta interdisciplinariedad, la importancia y relevancia de sus aplicaciones y el volumen de resultados logrados, inmediatamente se destacaron de otras metodologías como, la Teoría del Caos, la Teoría de la Evidencia, etc..

Un esquema general que representa y define los elementos del Softcomputing se muestra en la Figura 2.1.

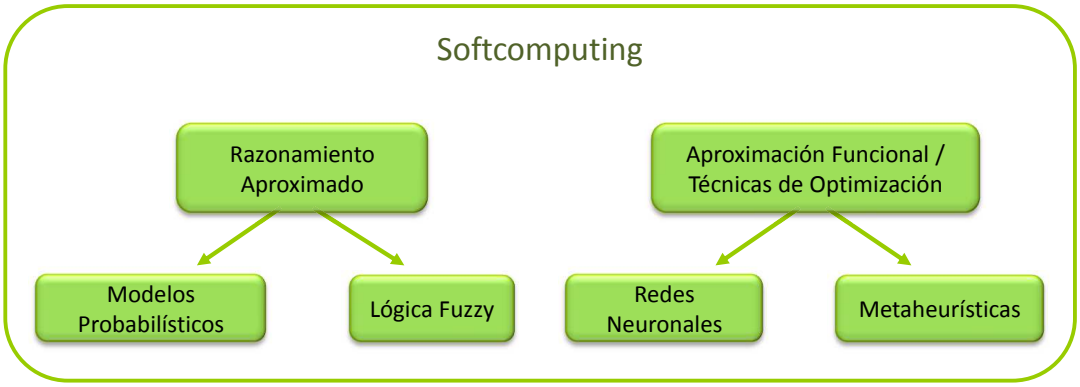


Figura 2.1: Esquema de los elementos que componen el Softcomputing

Los conjuntos fuzzy constituyen el paradigma más antiguo y más estudiado del Softcomputing, [20, 192, 218]. Esto es por dos razones, por un lado porque los conjuntos fuzzy son muy adecuados para modelar diferentes formas de incertidumbre y ambigüedades, que encontramos frecuentemente en la vida real. Además la integración de los conjuntos fuzzy con otras herramientas del Softcomputing han dado lugar a una generación de sistemas más poderosos, inteligentes y eficientes, [149]. Por otro lado porque, hasta que en 1994 en [218] se presentara la primera definición del término Softcomputing, todas las referencias a los conceptos que actualmente rodean a este término solía hacerse de forma atómica, es decir, se hablaba de manera aislada de cada uno de ellos con indicación del empleo de metodologías fuzzy. Todo lo fuzzy quedaba centrado en los conjuntos fuzzy.

Desde que en 1965 fuera introducido el concepto de conjunto fuzzy, [215], permitiendo la pertenencia de un elemento a un conjunto de forma gradual y no de manera absoluta como establece la teoría conjuntista clásica, las aplicaciones y desarrollos basados en este sencillo

concepto han evolucionado de tal modo que, hoy en día, es prácticamente imposible calcular el volumen de negocio que generan en todo el mundo, [191].

Los conjuntos fuzzy fueron introducidos como una nueva forma de representar la vaguedad en la vida diaria. La teoría de conjuntos fuzzy nos proporciona una forma aproximada pero efectiva para describir las características de un sistema que es demasiado complejo y/o impreciso para ser definido por un análisis matemático preciso. Un enfoque fuzzy está basado en la premisa de que los elementos claves en el pensamiento humano no son solamente números, sino que también puede ser aproximaciones a tablas de conjuntos fuzzy, es decir, a clases de objetos cuya transición de pertenecer a una clase y otra es una transición gradual en lugar de abrupta, [149]. Mucha de la lógica que subyace bajo el razonamiento humano no es la lógica tradicional bivaluada, sino que es lógica multivaluada, la lógica fuzzy. La lógica fuzzy ha sido considerada en dos sentidos diferentes.

Por un lado se ha considerado la lógica fuzzy en un sentido estricto. Esta lógica consiste en un sistema lógico dirigido a una formalización del razonamiento aproximado. Como tal, tiene sus raíces en la lógica multivaluada, pero desde un punto de vista diferente del sistema tradicional lógico multivaluado. Así que debe de tenerse en cuenta que muchos de los conceptos que explican la eficacia de la lógica fuzzy como una lógica de razonamiento aproximado no son parte de un sistema tradicional lógico de valores multivaluados.

Por otro lado, desde un sentido más amplio, la lógica fuzzy es casi un sinónimo de la teoría de los conjuntos fuzzy. La teoría de los conjuntos fuzzy, como su nombre sugiere, es básicamente una teoría de clases con límites poco definidos, [217]. La teoría de los conjuntos fuzzy es mucho más amplia que la lógica fuzzy en su sentido estricto y contiene a esta última como una de sus ramas. Otras ramas de la teoría de los conjuntos fuzzy son por ejemplo, la aritmética fuzzy, la programación matemática fuzzy, la topología fuzzy, la teoría de grafos fuzzy o el análisis de datos fuzzy.

Lo que es importante reconocer es que cualquier teoría crisp puede ser fusificada mediante la generalización del concepto de conjunto fuzzy. De hecho, es muy probable que eventualmente muchas teorías puedan ser fusificadas en este sentido, puesto que el ímpetu por la transición desde la teoría crisp a la fuzzy deriva en el hecho de que tanto la generalidad de una teoría como su aplicabilidad a los problemas del mundo real, son mejorados sustancialmente reemplazando el concepto de un conjunto por el de un conjunto fuzzy. La tendencia se inclina por utilizar el concepto de lógica fuzzy en su más amplio sentido.

Un concepto importante para terminar de comprender a grandes rasgos la teoría de los conjuntos fuzzy es el término de variable lingüística. Este término, como su nombre sugiere, es una variable cuyo valor son palabras o frases en un lenguaje natural o sintético, [218]. Por ejemplo, la edad es una variable lingüística si sus valores lingüísticos son joven, adulto, de

mediana edad, de avanzada edad, no muy joven, etc.. Una variable lingüística es interpretada como una etiqueta de un conjunto fuzzy que está caracterizada por una función de pertenencia. Un aspecto importante es que una variable lingüística puede ser vista como una forma de comprensión de los datos, que puede ser denominada granulación. Los gránulos de información son colecciones de entidades dibujadas entre sí por su similitud, su proximidad funcional, espacial o temporal. Un efecto similar se puede conseguir también por la cuantificación convencional. Sin embargo, en el caso de la cuantificación los valores son intervalos mientras que en el caso de la granulación, los valores son conjuntos fuzzy superpuestos, [149, 217]. Algunas ventajas de la granulación frente a la cuantificación son que la granulación es más general, tiende a imitar la forma en que los seres humanos interpretan los valores lingüísticos o que la transición de un valor lingüístico a otro valor lingüístico contiguo es gradual y no abrupto, dando como resultado continuidad y robustez.

2.3 El análisis inteligente de datos

En una amplia variedad de áreas y campos se han ido coleccionado y acumulado datos a un ritmo dramático, de ahí la creciente necesidad de una nueva generación computacional de teorías, herramientas y técnicas que ayuden a extraer conocimiento útil de forma rápida a partir de los grandes volúmenes de información. Esta nueva generación computacional se encuentra bajo el campo del AID. En un cierto nivel de abstracción, el área del AID manifiesta su preocupación con el desarrollo de métodos y técnicas que proporcionan un sentido en los datos coleccionados. Sin embargo, el problema básico que aborda el proceso del AID es el mapeo de datos de más bajo nivel en otras formas que pueden ser más compactas, más abstractas o más útiles. Así, el AID se define como un proceso complejo de descubrimiento de información o conocimiento útil a partir de los datos. En otras palabras, el AID es un proceso no trivial de identificación válida, novedosa y potencialmente útil de los patrones comprensibles en los datos. Este proceso complejo, se compone de una serie de fases interactivas e iterativas donde el usuario debe de tomar bastantes decisiones a lo largo del proceso. Concretamente, el proceso del AID se organiza en torno a las cinco fases siguientes, (Figura 2.2) [92, 97, 136, 173, 201]:

- Integración y recopilación de datos
- Preprocesamiento de datos
- Minería de datos
- Evaluación e interpretación
- Difusión y uso de modelos

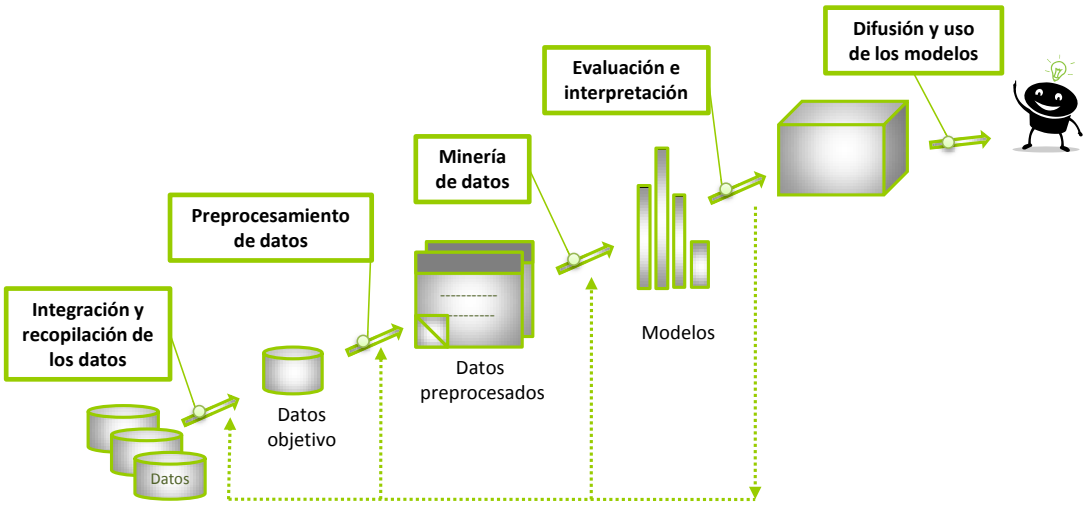


Figura 2.2: Fases del proceso del Análisis Inteligente de Datos

A continuación, vamos a detallar cada una de las fases que componen el AID. En la primera fase "integración y recopilación de datos" el usuario debe de determinar el dominio de aplicación del problema que se presenta, para identificar el objetivo a cumplir y determinar las posibles fuentes de información que podrían ser útiles y dónde obtener tales fuentes. Por lo tanto, la salida de esta fase es un conjunto de datos descritos mediante un conjunto de atributos que identifican las variables del problema a resolver.

La segunda fase, la fase de preprocesamiento de datos, es una fase determinante en el proceso del AID, ya que el principal objetivo de esta fase es la de llevar a cabo un preproceso en los datos con el fin de mejorar la calidad de los mismos. En esta fase los datos son modificados. Primero se transforman todos los datos a un formato común, frecuentemente mediante un almacén de datos para conseguir unificar de manera operativa toda la información recogida. Después de la transformación inicial de los datos, se lleva a cabo un preprocesamiento en los mismos para por ejemplo, eliminar o corregir algunos datos incorrectos, eliminar datos irrelevantes o incoherentes, etc.. Un aspecto importante en esta fase es que el usuario ya debe de haber decidido la técnica de minería de datos a aplicar en la siguiente fase del AID, ya que la salida de esta fase consiste en un conjunto de datos preprocesados con una mayor calidad en los mismos y adaptados a la técnica de minería de datos que vaya a ser aplicada con posterioridad.

La siguiente fase es la minería de datos. Esta fase también es crucial en el AID, ya que los resultados de esta fase serán claves en la solución final al problema propuesto. En esta fase, el usuario debe de decidir la tarea a realizar y debe elegir la técnica que se va a utilizar. El objetivo fundamental de la minería de datos es encontrar modelos inteligibles a partir de los datos. Para que este proceso fuera efectivo debería ser automático o semiautomático y los modelos

descubiertos deberían ayudar a tomar decisiones más seguras que aporten algún beneficio al sistema. La salida de esta fase es un modelo o varios modelos construidos a partir de los datos de entrada provistos por la fase anterior de preprocesamiento. El modelo o los modelos obtenidos dependerán del algoritmo de minería de datos seleccionado como más adecuado para los datos a analizar.

La penúltima fase consiste en evaluar el modelo o los modelos obtenidos en la fase de minería de datos. En esta fase de evaluación e interpretación, el objetivo es analizar y comprender la robustez y la adecuación del modelo obtenido como solución al problema propuesto. Esta evaluación se suele llevar a cabo por expertos en el problema a tratar y es decisión de dichos expertos indicar si el modelo o los modelos provistos son lo suficientemente válidos y/o comprensibles. En caso afirmativo el proceso continuaría con la última fase, pero en caso negativo habría que volver a fases anteriores para realizar una nueva iteración del proceso y conseguir una nueva solución.

Como última fase tenemos la fase de difusión y uso del modelo. En esta fase, una vez que el modelo o los modelos propuestos han sido validados, solamente queda el uso del mismo para obtener la información y conocimiento útil que dicho modelo nos proporciona para solucionar el problema en cuestión.

Aunque todas las fases del AID son fundamentales, en este trabajo nos vamos centrar en las fases de preprocesamiento de los datos y de minería de datos que son fases claves para determinar una buena solución a un problema propuesto. Por este motivo dedicaremos más adelante dos capítulos a estas dos fases, centrándonos en el tratamiento de datos que realizan.

2.4 Softcomputing en el análisis inteligente de datos

Las metodologías que integran el Softcomputing, más que poder ser entendidas de forma aislada, hay que comprenderlas como el resultado de la cooperación, la asociación, la complementariedad o la hibridación de sus componentes de segundo nivel, [192]. Los componentes que integran el segundo nivel del Softcomputing como la lógica fuzzy, los modelos probabilísticos, las redes neuronales, los algoritmos genéticos y las metaheurísticas, son ampliamente aplicados en el proceso del AID. Más específicamente, estos componentes son utilizados principalmente en los modelos y en el diseño de técnicas y algoritmos en las fases del AID de preprocesamiento de datos y de minería de datos.

La implantación del Softcomputing dentro del marco del AID, ofrece la posibilidad de crear modelos tolerantes a la imprecisión e incertidumbre. En nuestro caso estamos interesados en aplicar el Softcomputing dentro del marco de AID con el objetivo de flexibilizar el tratamiento del tipo de datos con los cuales trabajar. Más concretamente, nuestra meta es permitir que las

técnicas del AID puedan manejar de forma directa datos vagos, imprecisos e inciertos, es decir, de baja calidad. En otra palabras, nuestro objetivo es proponer técnicas dentro de las fases del AID de preprocesamiento y minería de datos que sean flexibles y tolerantes a la imperfección que aparece en los datos.

Desde la primera definición del Softcomputing se ha tendido a introducir la tolerancia a la imperfección directamente en el diseño de las técnicas y algoritmos, evitando, ignorando o transformando la imperfección cuando ésta aparecía de forma explícita en los datos. Sin embargo, en los últimos años hay una tendencia a desarrollar y a adaptar técnicas ya existentes para que sean capaces de manejar de forma directa la imperfección en los datos, puesto que cada vez con más frecuencia debido a una gran cantidad de factores los datos no son tan precisos como sería deseable. La motivación de dirigir este trabajo hacía el diseño de nuevas técnicas y la adaptación de técnicas ya existentes, dentro de las fases del preprocesamiento de datos y de la minería de datos, está marcada por el hecho de que los datos que se toman como entrada en la fase de minería de datos deben de ser de cierta calidad para asegurar el éxito de los resultados. Si los datos originales recopilados ya contienen información de baja calidad y estos datos deben de pasar por la fase de preprocesamiento de datos previamente, es lógico que tanto las técnicas aplicadas durante la fase de preprocesamiento como durante la fase de minería de datos trabajen con datos de baja calidad. El problema es que como vamos comprobar en los siguientes capítulos de este documento, el número de técnicas dentro de la fase de preprocesamiento que trabajan de forma directa con datos de baja calidad es bastante escaso y raramente son estudiadas. Pero no sólo en la fase de preprocesamiento de datos las técnicas para trabajar con este tipo de datos son escasas, sino que también en la fase de minería de datos hasta hace muy poco tiempo las técnicas propuestas eran bastante pocas.

Antes de realizar un recorrido por la literatura para presentar de forma más detallada las fases de preprocesamiento de datos y de minería de datos, en el Capítulo 3, vamos a presentar qué se entiende por datos de baja calidad, cuáles son estos tipos de datos y como se suele trabajar con cada uno de ellos. Tras presentar estos tipos de datos en el Capítulo 4 vamos a detallar la fase de preprocesamiento de datos del proceso del AID, centrándonos en las técnicas de discretización, selección de atributos e imputación de valores missing tomando especial atención en aquellas técnicas que puedan trabajar con datos de baja calidad. La misma atención tendremos en el Capítulo 5, donde detallaremos la fase de minería de datos dentro del proceso del AID, realizando una taxonomía de sus técnicas y poniendo especial énfasis en aquellas que manejen datos de baja calidad de forma explícita en los conjuntos de datos.

CAPÍTULO

3

Datos de Baja Calidad

3.1 Introducción

En la literatura podemos encontrar una gran variedad de técnicas del AID basadas en diferentes propuestas teóricas. Desafortunadamente, muchas de estas técnicas convencionales no consideran posibles fuentes de información imperfecta. Como resultado, los datos incompletos, imprecisos e inciertos suelen ser descartados e ignorados en el aprendizaje de las técnicas y subsecuentemente en el proceso de inferencia. Sin embargo, estos datos aparecen inevitablemente en aplicaciones del mundo real. Los errores en los instrumentos y/o la corrupción debido al ruido durante los experimentos pueden provocar la obtención de información con datos incompletos cuando estamos obteniendo el valor de un atributo específico. En otros casos, la extracción de información exacta puede ser excesivamente costosa o inviable. Por otra parte, podría ser de utilidad completar los datos disponibles con información adicional de un experto que normalmente es obtenida mediante valores imprecisos como datos intervalares, conceptos fuzzy, etc..

Antes de continuar, vamos a definir el concepto de información imperfecta o datos imperfectos, para así después poder etiquetar a las técnicas de acuerdo con el tipo de información imperfecta que permitan en sus datos de entrada.

La imprecisión y la incertidumbre pueden ser considerados como dos aspectos complementarios de la información imperfecta [22, 69]. Desde un punto de vista práctico (Figura 3.1) un item de información puede ser representado como una cuádrupla (atributo, objeto, valor, confianza). El atributo es una función que asigna un valor (o conjunto de valores) al objeto. El valor es un subconjunto del dominio de referencia asociado al atributo. Por último, la confianza

indica la veracidad del item de información. En este contexto, la imprecisión está relacionada con el valor del item de información, mientras que la incertidumbre está relacionada con la confianza en el item.

Así, un item de información será preciso cuando su valor no pueda ser subdividido. En otro caso, se habla de imprecisión. Además, cuando no hay unos límites exactos (crisps) en el conjunto de valores que un item impreciso puede tomar, entonces hablamos de imprecisión fuzzy.

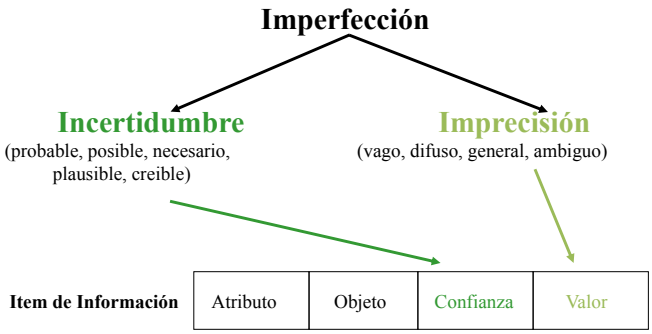


Figura 3.1: Descripción de un item de información

Por otro lado, la incertidumbre es una propiedad de la creencia. Decimos que tenemos certeza en un evento cuando le asignamos un valor de creencia máximo. Así, podemos definir incertidumbre como la ausencia de certeza y esto puede surgir de la aleatoriedad de algunos experimentos (incertidumbre objetiva) o de los juicios subjetivos del razonamiento humano (incertidumbre subjetiva).

En [68] los conceptos de imprecisión e incertidumbre se describen en términos de incertidumbre estocástica y epistémica. La incertidumbre estocástica surge de la variabilidad aleatoria relacionada con los procesos naturales. Por su parte, la incertidumbre espistémica aparece desde la naturaleza incompleta/imprecisa de la información disponible. Si bien la incertidumbre estocástica se trata adecuadamente utilizando la teoría de la probabilidad clásica, varias teorías sobre la incertidumbre epistémica se han desarrollado con el fin de manejar de forma explícita la información incompleta/imprecisa y que básicamente son las teorías de los conjuntos convexos de probabilidad, conjuntos aleatorios y la teoría de la posibilidad.

Sería de utilidad incorporar el tratamiento de datos imperfectos tanto en la fase de aprendizaje como en la fase de inferencia de las técnicas del AID. El objetivo es que estas técnicas puedan trabajar con atributos heterogéneos (nominales y numéricos) expresados por valores exactos o crisps, valores missing, valores con incertidumbre objetiva, valores con incertidumbre subjetiva, valores imprecisos o intervalos y valores imprecisos fuzzy o sus etiquetas lingüísticas asociadas.

Con respecto a la heterogeneidad de los atributos, podemos decir que algunas técnicas solamente pueden trabajar con atributos con dominios nominales y con atributos con dominios numéricos que necesitan ser discretizados, como en es el caso de las redes Bayesianas y técnicas basadas en probabilidades imprecisas, pero actualmente se están desarrollando en este ámbito técnicas que permiten trabajar directamente con atributos numéricos sin que sean discretizados, [122]. Otras técnicas solamente pueden trabajar con atributos numéricos y solo permiten un atributo nominal considerado como clase, como en el caso de las redes neuronales o de los clasificadores que generan funciones discriminantes. Entre las técnicas que permiten trabajar con ambos tipos de atributos se incluyen las técnicas basadas en árboles de decisión tanto de clasificación como de regresión, las técnicas basadas en muestreo de patrones a través de la definición de funciones distancia heterogéneas, y en general aquellas técnicas que generan reglas, donde en las reglas se permiten ambos tipos de atributos, con excepción de las reglas de asociación que sólo trabajan con dominios nominales.

Respecto al tratamiento de datos imperfectos, podemos decir que hay muchas restricciones sobre el tipo de imperfección permitida en los datos de entrada. Por esta razón, nuestro propósito es incluir el tratamiento de información imperfecta en diferentes técnicas dentro del AID. Nos vamos a referir a la información/datos que contengan algún valor imperfecto como información/datos de baja calidad ("low quality data"), término que denotaremos por LQD. A continuación pasamos a detallar los datos de baja calidad con los cuales las técnicas que más adelante presentaremos van a ser capaces de trabajar de manera explícita.

3.2 Datos de baja calidad

Como acabamos de comentar, en muchos problemas reales los datos tienen un cierto grado de imperfección. Debido a esta situación, los conjuntos de datos a partir de los cuales se extraen los modelos o sistemas de conocimiento podrán contener información de baja calidad.

En esta sección vamos a realizar un estudio sobre algunos tipos de datos de baja calidad y algunas opciones de cómo las técnicas del AID pueden trabajar con estos tipos de datos.

3.2.1 Valores faltantes o missing

Uno de los tipos de valores imperfectos que se encuentran con mayor frecuencia en los datos y que quizás por ello han recibido una mayor atención, es la existencia de valores faltantes o missing.

Según se propone en [13] y [129] los valores missing existentes en los datos se pueden clasificar según su aleatoriedad en:

- Missing completamente aleatorios ("missing completely at random" MCAR). Este es el nivel de aletoriedad más alto y ocurre cuando la probabilidad de que un ejemplo con un valor missing para un atributo no depende ni de los valores conocidos ni del atributo. En este nivel de aleatoriedad es posible utilizar cualquier técnica de tratamiento de datos sin riesgo de introducir sesgo en los datos.

- Missing aleatorio ("missing at random" MAR). Este tipo de missing se produce cuando la probabilidad de que un ejemplo contenga un valor missing depende de los valores conocidos pero no del valor missing en sí mismo.

- No missing aleatorio ("not missing at random" NMAR). Cuando la probabilidad de que un ejemplo contenga un valor missing puede depender del propio valor del atributo.

En este trabajo nos vamos a centrar en el segundo tipo de missing, es decir, en valores missing aleatorios. En este caso, las técnicas de AID pueden:

- Permitir los valores missing en el conjunto de datos cuando la técnica es robusta a la existencia de estos valores.

- Eliminar los atributos con valores missing del conjunto de datos.

- Eliminar los ejemplos con valores missing del conjunto de datos.

- Reemplazar los valores missing manualmente (si no son muchos) o automáticamente por un valor que preserve la media o varianza, global o por clases, en el caso de los atributos numéricos, o por la moda en el caso de los atributos nominales. Otra forma para estimar el valor es predecirlo a partir de otros ejemplos (imputación de valores missing), utilizando una técnica de preprocesamiento predictiva.

- Reemplazar los valores missing por un intervalo o conjunto crisp/fuzzy que abarque todo el dominio del atributo correspondiente reflejando la ausencia total de información.

3.2.2 **Ruido**

El ruido es un término amplio que ha sido interpretado de diferentes maneras. Por ejemplo, ha sido definido como un error aleatorio en la medida de un atributo. Otra definición establece que el ruido es cualquier propiedad detectada en un patrón que no se debe al modelo real subyacente, sino a una aleatoriedad en el mundo o en los sensores, [98]. Si bien una descripción de la taxonomía completa del ruido es un tema de investigación abierto, generalmente podemos encontrar dos tipos de ruido en un conjunto de datos, [28]: ruido en la clase, o ruido en el

atributo. El ruido en la clase ocurre cuando un ejemplo pertenece a una clase incorrecta. El ruido en la clase puede ser atribuido a varias causas, incluyendo la subjetividad en el proceso de etiquetado, los errores en la entrada de datos o la ausencia de algunos atributos representativos. Por el contrario, el ruido en el atributo refleja valores erróneos para uno o más atributos (atributo independiente) del conjunto de datos. La complejidad de la detección del ruido de clase es relativamente menor a la detección del ruido en un atributo.

En estas definiciones, el ruido es considerado como un error que ocurre aleatoriamente. El error en los datos puede ocurrir por varias razones, por ejemplo, por problemas en los instrumentos o equipos de medición o por el hecho de que grandes conjuntos de datos se obtienen mediante técnicas automáticas. Dependiendo de la técnica de AID empleada, se puede tratar el ruido utilizando las siguientes opciones:

- Si es posible detectar qué valores contienen ruido, el tratamiento puede ser similar al caso de los valores missing.
- Algunas veces es posible conocer el error del instrumento de medida, (media y desviación estándar). Esto permite incorporar esta información en el conjunto de valores de un atributo con ruido mediante alguna transformación. Por ejemplo, en un sensor de medición si conocemos la media μ y la desviación típica σ del error en la medida de un atributo, podemos transformar los valores de dicho atributo en valores imprecisos que recojan ese error. Por ejemplo, el valor crisp v puede ser transformado en el valor fuzzy $v = [\mu - 2\sigma, \mu - \sigma, \mu + \sigma, \mu + 2\sigma]$.

3.2.3 **Imprecisión / incertidumbre**

Algunas representaciones simples de la información imprecisa/incierta basada en intervalos y sus generalizaciones son:

- Utilizando un intervalo $[a, b]$, donde asumimos que el valor del atributo se encuentra dentro del intervalo.
- Utilizando un conjunto fuzzy que asigna un grado de posibilidad entre 0 y 1 para cada valor del conjunto como un posible valor del atributo.

Otra forma de representar la imprecisión/incertidumbre es mediante subconjuntos crisp/-fuzzy.

- En el caso de los atributos nominales, podemos utilizar un subconjunto crisp a partir de los valores del dominio, tal como {rojo,azul}, si el dominio es por ejemplo {rojo,azul,verde}

o podemos utilizar un subconjunto fuzzy asignando un grado de posibilidad a cada valor del subconjunto, por ejemplo, suponiendo el mismo dominio anterior, {0.2/rojo,0.8/azul}.

- En el caso de los atributos numéricos, podemos utilizar un subconjunto de etiquetas lingüísticas que están asociadas con una partición fuzzy del atributo. Por ejemplo, un valor de temperatura de 45 grados puede ser transformado en un subconjunto fuzzy como {0.0/frío,0.2/templado,0.8/caliente}. Si la partición es una partición de Ruspini, la suma de los grados de pertenencia es 1. Para cualquier otro tipo de partición fuzzy, la suma puede tomar cualquier otro valor, así que un posible subconjunto fuzzy también podría ser {0.1/frío,0.3/templado,0.9/caliente}. A lo largo de este trabajo, trabajaremos siempre con particiones de tipo Ruspini.

Dependiendo de la técnica de AID con la que trabajemos se podrá:

- Permitir estos valores en el conjunto de datos cuando la técnica es robusta a la existencia de tales valores.
- Eliminar los atributos con imprecisión/incertidumbre del conjunto de datos.
- Eliminar los ejemplos con imprecisión/incertidumbre del conjunto de datos.
- Reemplazar/imputar los valores con imprecisión/incertidumbre por otro tipo de valores con los cuales la técnica pueda trabajar. Por ejemplo, en el caso de un valor fuzzy, un intervalo que contenga su soporte o el centro de gravedad de dicho valor fuzzy, etc..

Como conclusión podemos establecer que los datos de baja calidad están presentes en las aplicaciones y bases de datos del mundo real. De forma general, hay dos aproximaciones para abordar su tratamiento: 1) eliminarlos/transformarlos cuando la técnica de AID no puede trabajar con ellos y 2) diseñar/desarrollar técnicas que sean capaces de tratar con ellos, es decir, que puedan trabajar explícitamente con LQD. La primera opción es inmediata como solución ya que sólamente oculta la verdadera naturaleza de los datos, de forma que hay una pérdida de información. La segunda requiere del desarrollo de nuevas técnicas o extensión de las existentes para que sean capaces de interpretar las distintas fuentes de imperfección que aparecen en los datos. Este esfuerzo tiene como ventaja una mayor riqueza de información en los mismos. En esta tesis se ha optado por la segunda alternativa llevando a cabo el desarrollo de nuevas técnicas y la extensión de otras para que puedan trabajar con datos que expresan su verdadera naturaleza lo más fielmente posible, abordando tanto la fase de Minería de Datos como de Preprocesamiento de Datos dentro del AID.

CAPÍTULO

4

Preprocesamiento de Datos

4.1 Introducción

El preprocesamiento de datos es la segunda fase dentro del AID, [92, 97, 136, 173, 201]. Esta fase puede tener un gran impacto sobre el rendimiento generalizado en los algoritmos supervisados, [118]. La calidad del conocimiento descubierto no sólo depende del algoritmo de minería de datos usado, sino también de la calidad del conjunto de ejemplos empleados para el aprendizaje del modelo. En la fase de preprocesamiento de datos se selecciona y preprocesa el conjunto de ejemplos de entrada, es decir, se trata de preprocesar los conjuntos de datos para conseguir eliminar aquellos atributos de los mismos que posteriormente durante la minería de datos puedan causar algún contratiempo en los resultados esperados. Este paso es necesario ya que algunos de los datos coleccionados en la fase anterior son irrelevantes o innecesarios para la tarea de desarrollar. Además, a veces hay datos que afectan a la calidad de la información: errores en los datos, datos faltantes, etc.. Un adecuado preprocesamiento de los datos puede eliminar estos efectos, para así evitar resultados nos deseados.

Por lo tanto, el preprocesamiento de datos debe tratar con un número de fenómenos que deben de ser modelados. Uno de los problemas que se debe de afrontar en la fase de preprocesamiento de datos es comprender y analizar la naturaleza de los datos evitando la pérdida de información útil durante el proceso. Además, hay que tener en cuenta que el preprocesamiento de datos es una fase que consume un gran cantidad de tiempo computacional comparado con otras fases del AID. Dado que tras esta fase, la calidad de los datos debe ser la esperada, es posible que algunas técnicas de preprocesamiento de datos lleguen a ser iterativas ya que hasta que no se consigue la calidad de los datos esperada la técnica prosigue la búsqueda.

Algunos de los procesos más importantes en el preprocesamiento de datos son la normalización, la discretización, la selección de atributos, la imputación de valores missing y la selección de ejemplos. Un aspecto a tener en cuenta en estos procesos es el tipo de datos de entrada con los que se trabaja. En la literatura podemos encontrar una gran variedad de técnicas para preprocesar datos, sin embargo, la mayoría de ellas solamente puede tratar con datos precisos y ciertos y algunas de ellas tratan con datos imprecisos pero no de forma explícita en los datos sino en el modelo que se construye. En este trabajo siempre vamos a centrar la atención en aquellas técnicas que trabajen con LQD, aunque es importante reseñar que no existen muchas actualmente en la literatura.

De forma resumida, la fase de preprocesamiento de datos comprende aquellas técnicas que se ocupan de analizar los datos en bruto con el fin de producir datos de calidad. Esta fase incluye principalmente la recogida de datos, integración de los datos, transformación de datos, limpieza de datos, reducción de datos y la discretización de los datos, [97, 118].

En las siguientes secciones vamos a centrarnos en tres de los procesos más usados en la fase de preprocesamiento de datos, la discretización, la selección de atributos y la imputación de valores missing. Para cada proceso vamos a presentar una definición del mismo, una taxonomía de las diferentes técnicas que desarrollan el proceso y, por último, nos vamos a centrar de forma no exhaustiva en aquellos trabajos presentes en la literatura que tratan con LQD. Es importante señalar que la cantidad de técnicas que pueden trabajar de forma directa con este tipo de datos es bastante escasa, de ahí que nuestra propuesta sea el diseñar técnicas de discretización, selección de atributos e imputación de valores missing que trabajen de forma explícita con información de baja calidad.

4.2 La discretización

Un aspecto importante a la hora de llevar a cabo el preprocesamiento de los datos es tener en cuenta el tipo de datos con los cuales la técnica de minería de datos aplicada posteriormente puede trabajar. Hay técnicas que requieren tipos de datos específicos. Así, algunos atributos se pueden numerizar, lo que reduce el espacio y permite usar técnicas numéricas. El proceso inverso consiste en discretizar los atributos numéricos, es decir, transformar atributos numéricos en atributos discretos o nominales.

Los atributos discretizados pueden tratarse como atributos categóricos con un número más pequeño de valores. La idea básica es partir los valores de un atributo numérico en una pequeña lista de intervalos, tal que cada intervalo es visto como un valor discreto del atributo. Cuando los intervalos de discretización permiten una graduación en la pertenencia de valores a dichos intervalos, hablamos de intervalos fuzzy o discretización fuzzy. Mientras que la lógica fuzzy

es multivaluada, la lógica clásica es binaria o bivaluada. En los intervalos clásicos (intervalos crisp) un determinado valor del dominio solamente pertenece a un intervalo, es decir, la pertenencia a cada intervalo será 0 ó 1, mientras que en los intervalos fuzzy, un determinado valor puede pertenecer a más de un intervalo con un grado de pertenencia distinto de 0. En este sentido, el uso de la Teoría de Conjuntos Fuzzy ha dado muy buenos resultados para el tratamiento de información de forma cualitativa, [216]. El modelado lingüístico fuzzy es una herramienta que permite representar aspectos cualitativos y que está basada en el concepto de variables lingüísticas, es decir, variables cuyos valores no son números, sino palabras o sentencias expresadas en lenguaje natural o artificial, [216]. Cada valor lingüístico se caracteriza por un valor sintáctico o etiqueta y un valor semántico o significado. La etiqueta es una palabra o sentencia perteneciente a un conjunto de términos lingüísticos y el significado es un subconjunto fuzzy en un universo de discurso. Se ha demostrado que es una herramienta muy útil en numerosos problemas, como por ejemplo en recuperación de información, evaluación de servicios, toma de decisiones, procesos de consenso, etc., [208]. Es por esto, que la lógica fuzzy se ha introducido en técnicas y algoritmos de clasificación. Sin embargo, al introducir la lógica fuzzy en estas técnicas muchas veces es necesario realizar una discretización fuzzy de los atributos numéricos. La construcción de los conjuntos fuzzy en los que se discretiza o particiona un dominio numérico supone un importante problema en el campo de la minería de datos y el Softcomputing, debido a que la determinación de dichos conjuntos puede afectar profundamente al rendimiento de los distintos modelos obtenidos para la tarea de clasificación, [9].

Muchos algoritmos de minería de datos han sido orientados originalmente para manejar atributos nominales, [80, 130, 211], incluso algunos solamente pueden trabajar con este tipo de atributos. Por ejemplo, tres de las diez técnicas consideradas en el top 10 de la minería de datos, [206], necesitan una discretización embebida o externa de los datos. Estos algoritmos son C4.5 [169], APriori [2] y Naive Bayes [210]. También hay algoritmos que son capaces de trabajar con datos numéricos aunque su aprendizaje es más eficiente y efectivo cuando trabajan con datos discretizados [80], ya que cuando se trabaja con estos datos se produce una simplificación de los mismos haciendo el proceso de aprendizaje más rápido y preciso. Además la discretización compacta y acorta los resultados y si los datos contienen ruido, éste puede ser reducido. Sin embargo, cualquier proceso de discretización generalmente tiene que tratar con la perdida de información, por lo que el objetivo principal de cualquier algoritmo de discretización es minimizar al máximo la perdida de información relevante durante el proceso, [117].

4.2.1 El concepto de discretización

La discretización es una técnica de preprocesamiento esencial en el AID. La representación de la información a través de intervalos o conjuntos fuzzy es más concisa y más fácil de interpretar en ciertos niveles de conocimiento que la representación mediante valores numéricos. Hay que tener en cuenta que obtener una discretización óptima es un problema NP-completo, [56].

En aprendizaje supervisado, y especialmente en clasificación, la discretización puede ser definida como sigue, [80]: Suponiendo un conjunto de datos de $|E|$ ejemplos y $|C|$ clases, un algoritmo de discretización debería discretizar el atributo numérico a de dicho conjunto de datos en m intervalos discretos $D = \{[d_0, d_1], (d_1, d_2], ..., (d_{m-1}, d_m]\}$, donde d_0 es el valor mínimo, d_m es el valor máximo y $d_i < d_{i+1}$, para $i = 0, 1, ..., m-1$. Así, el resultado discreto D recibe el nombre de esquema de discretización sobre el atributo a, donde $P = \{d_1, d_2, ..., d_{m-1}\}$ es el conjunto de puntos de corte de dicho atributo.

Un típico proceso de discretización consiste en llevar a cabo las cuatro etapas siguientes [117, 130]:

1. Ordenar los valores de los atributos numéricos que vayan a ser discretizados.
2. Evaluar un punto de corte para dividir el dominio o para mezclar dos intervalos adyacentes.
3. Dependiendo del criterio, dividir o mezclar intervalos de valores numéricos.
4. Chequear si el criterio de parada es satisfecho y en caso afirmativo terminar el proceso.

En la Figura 4.1 se puede ver de forma gráfica y de forma general el proceso de discretización que dependiendo de la técnica utilizada puede ser un proceso de varias iteraciones. Además, este proceso está referido a la discretización de un atributo, pero también es posible aplicarlo a multiples atributos considerando varios atributos simultáneamente.

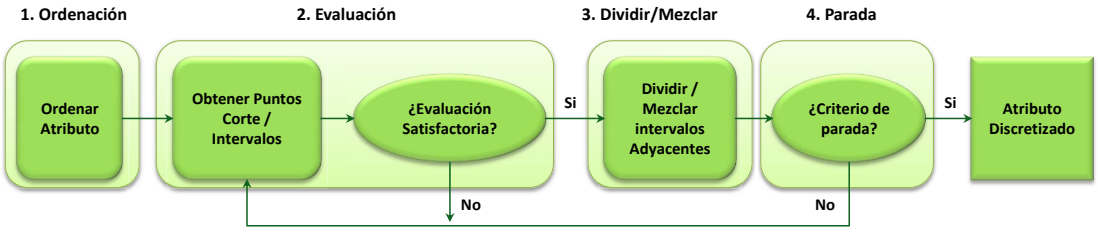


Figura 4.1: Proceso de discretización

A continuación, vamos a describir más detalladamente cada una de las etapas del proceso de discretización, [130]:

- **Ordenación**: Los atributos numéricos de cada uno de los atributos deben ser ordenados bien en orden ascendente o descendente. La ordenación puede ser computacionalmente costosa, por lo que es importarte elegir un buen algoritmo de ordenación. Un buen ejemplo es el "Quick Sort" cuya complejidad es de $O(N \log N)$.

- **Evaluación**: Después de ordenar, el siguiente paso en el proceso de discretización es encontrar o el mejor punto de corte para dividir el rango de valores numéricos o el mejor par de intervalos adyacentes para mezclarlos. Una función de evaluación debe determinar la correlación de la división o la mezcla respecto al atributo clase. Hay numerosas funciones de evaluación en la literatura que incluso sirven para categorizar las diferentes técnicas de discretización como veremos más adelante.

- **Dividiendo/Mezclando**: Dependiendo del enfoque utilizado, los intervalos serán divididos o mezclados respectivamente. Para dividir un intervalo, hay que evaluar los diferentes puntos de corte para seleccionar el mejor y dividir el rango de valores numéricos en dos particiones. De manera similar para mezclar un intervalo, los intervalos adyacentes son evaluados hasta encontrar el mejor par de intervalos para mezclarlos en cada iteración.

- **Criterio de parada**: El criterio de parada especifica cuando el proceso de discretización debe finalizar. Normalmente es dirigido por la compensación entre un bajo número de intervalos para una mejor comprensión con una menor precisión o por un número más alto de intervalos con una compresión menor pero una precisión mayor. Es importante que el criterio de parada sea sencillo, por ejemplo fijar un número específico de intervalos a definir.

Una vez descritas las fases que componen el proceso de discretización, vamos a detallar las categorías en las cuales se puede clasificar un algoritmo de discretización.

4.2.2 Categorización de las técnicas de discretización

En la descripción general del proceso de discretización, se ha diferenciado entre algoritmos para dividir el dominio en intervalos o para mezclar diferentes intervalos del dominio. Sin embargo, hay más y más completas taxonomías para las diferentes técnicas de discretización, [130], [80]. La taxonomía que vamos a mostrar a continuación toma como base las diferentes características de las técnicas de discretización:

- *Estática/Dinámica*: Esta característica se refiere al momento y a la independencia con la cual el algoritmo de discretización opera en relación con el algoritmo de aprendizaje. Un algoritmo de discretización es dinámico cuando actúa en la fase donde el algoritmo

de aprendizaje está construyendo el modelo. De esta forma solamente se tiene acceso de manera parcial a la información embebida en el algoritmo de aprendizaje, produciendo resultados precisos y compactos en conjunción con el algoritmo de aprendizaje asociado. Por otro lado, un algoritmo de discretización es estático cuando se ejecuta a priori de la tarea de aprendizaje y es independiente del algoritmo de aprendizaje que se va a usar. La mayoría de los algoritmos de discretización conocidos son estáticos debido al hecho de que muchos de los algoritmos dinámicos son realmente subpartes o etapas de los algoritmos de minería de datos cuando tratan con datos numéricos, [17]. Dos algoritmos bien conocidos de discretización dinámicos son el discretizador ID3 [168] y el algoritmo de particionamiento fuzzy de la teoría de la información (ITFP) [10].

- *Único atributo/multiatributo*: Las técnicas multiatributo, también conocidas como discretización 2D [145], consideran todos los atributos simultáneamente para definir un conjunto inicial o para decidir el mejor punto de corte en global. Estas técnicas pueden discretizar un atributo en un momento, estudiando las interacciones con los otros atributos. En contraste, hay técnicas que trabajan sólamente con un atributo cada vez, [120]. Recientemente ha surgido un interés por el desarrollo de discretizadores multiatributo, [183], ya que son una gran influencia en el aprendizaje deductivo, [15], y en los problemas complejos donde existe una alta interacción entre múltiples atributos. Estas características son obviadas por los discretizadores de un único atributo, [75].

- *Supervisada/No Supervisada*: Los algoritmos de discretización no supervisados no tienen en cuenta la etiqueta de clase [203, 210], mientras que los supervisados sí lo hacen. La forma en la cual los algoritmos supervisados toman en cuenta la etiqueta clase depende de la interacción entre los atributos de entrada, la etiqueta de clase y la medida heurística utilizada para determinar los mejores puntos de corte (entropía, interdependencia, etc.). Muchos de los discretizadores propuestos en la literatura son supervisados, [74, 168, 169, 182] y teóricamente, usando la información de clase, deberían automáticamente determinar el mejor número de intervalos para cada atributo. Si un algoritmo de discretización es no supervisado no significa que no pueda ser aplicado sobre una tarea supervisada. Sin embargo, un algoritmo supervisado solamente puede ser aplicado sobre problemas supervisados de minería de datos.

- *División/Mezcla*: Esta característica se refiere al procedimiento usado para crear o definir nuevos intervalos. Las técnicas de División establecen un punto de corte entre todos los posibles y dividen el dominio en dos intervalos. En contraste, las técnicas de Mezcla comienzan con un partición predefinida y borran un candidato a punto de corte para

mezclar dos intervalos adyacentes. Estas dos propiedades están muy relacionadas con las características top-down y bottom-up respectivamente. La idea subyacente es muy similar, excepto que las técnicas top-down y bottom-up utilizan procesos incrementales de acuerdo con una construcción de discretización jerárquica, [80]. Además, hay algoritmos de discretización que son considerados híbridos, ya que estos alternan la división y la mezcla de intervalos en tiempo de ejecución, [55].

- *Global/Local*: Un algoritmo de discretización puede o requerir todos los datos disponibles de un atributo o utilizar información parcial para tomar una decisión. Un algoritmo de discretización es local, [74, 169], cuando solamente basa sus decisiones en información local. Pocos algoritmos de discretización son locales, salvo algunas técnicas top-down y todos los algoritmos de discretización dinámicos. En los procesos top-down algunos algoritmos siguen el esquema de divide y vencerás y cuando encuentran un punto de corte, los datos a los que se accede posteriormente son parciales. Con respecto a los algoritmos de discretización dinámica, al encontrar los puntos de corte en operaciones internas de los algoritmos de minería de datos, no suelen tener acceso a la totalidad de los datos. Por el contrario, los algoritmos de discretización globales, [23, 133], tiene en cuenta todos los datos cuando llevan a cabo el proceso de discretización.

- *Directa/Incremental*: Los algoritmos de discretización directos, [99, 100], dividen el rango en k intervalos simultáneamente, pero requieren un criterio adicional para determinar el valor de k. Estos algoritmos no solo incluyen técnicas de discretización de una sola etapa sino que pueden estar formados por más de una. Por el contrario, las técnicas incrementales, [112, 132], comienzan con una discretización simple que se va refinando tras un proceso de mejora. En cada paso se encuentra el mejor punto de corte y después el resto de decisiones se toman en consecuencia a esos puntos de corte elegidos. En este caso, el criterio adicional necesario es saber cuándo que hay que detener dicho proceso.

- *Medida de Evaluación*: Esta característica se refiere a la métrica utilizada por el algoritmo de discretización para comparar dos esquemas candidatos y decidir cuál es el más adecuado. En la literatura, dependiendo de la referencia, las principales familias de medidas de evaluación difieren en los nombres de las familias. En [130] las diferentes medidas de evaluación en las que agrupan a los algoritmos de discretización son cuatro: Binning, Entropía, Dependencia y Precisión. Por el contrario en [80] son cinco las familias de medidas de evaluación que proponen: Información, Estadística, Rough Set, Wrapper y Binning. Esta última referencia incluye las medidas de evaluación anteriores, aunque

con diferente nombre. A continuación, vamos a describir brevemente en qué consiste cada una de estas medidas:

  - Información: Este grupo incluye la entropía como la medida de evaluación más utilizada en la discretización, [74, 169, 182] y otras medidas de la teoría de la información como el índice de Gini, [107].
  - Estadística: La evaluación estadística implica la medición de la dependencia y/o correlación entre los atributos, (algoritmo Zeta [99] y ChiMerge [112]), la interdependencia (CAIM [121]), el coeficiente de contingencia ([188]), la probabilidad y las propiedades bayesianas ([205]), etc..
  - Rough Set: Esta familia se compone de técnicas que evalúan los esquemas de discretización mediante el uso de medidas rough set y propiedades tales como las aproximaciones inferior y superior, la separabilidad de la clase, etc.. Dos técnicas incluidas en esta categoría son las presentadas en [57] y [54].
  - Wrapper: Esta colección comprende técncias que se basan en el error proporcionado por un clasificador que se ejecuta para cada evaluación. El clasificador utilizado puede ser muy simple, por ejemplo votar la clase mayoritaria, [190], o alguno más complejo como un sistema basado en reglas, [53].
  - Binning: Esta categoría se refiere a la falta de medida de evaluación. Es la técnica más simple para discretizar un atributo, ya que consiste en crear un número específico de intervalos. Cada intervalo es definido a priori y engloba un número específico de valores por atributo. Las técnicas de igual frecuencia y de igual amplitud, [203], usan el binning como medida de evaluación.

- *Paramétrica/No Paramétrica*: Esta característica se basa en la determinación automática del número de intervalos en los cuales cada atributo va a ser discretizado. Una técnica de discretización es no paramétrica cuando calcula el número apropiado de intervalos para cada atributo teniendo en cuenta un equilibrio entre la pérdida de información y la consistencia, obteniendo el número más bajo de ellos. Algunas técnicas no parámetricas son MDLP [74] y CAIM [121]. Por otra parte, una técnica de discretización es parámetrica cuando requiere un número máximo de intervalos deseados que debe de ser fijado por el usuario. Algunos ejemplos son la técnica ChiMerge [112] y la técnica presentada en [55].

- *Top-Down/Bottom-up*: Esta propiedad es observada en algoritmos de discretización incrementales, [80]. Las técnicas top-down empiezan con una discretización vacía y van

mejorando el proceso solamente añadiendo nuevos puntos de corte a la discretización. Una técnica clásica top-down es el MDLP [74]. Por otro lado, las técnicas bottom-up comienzan con una discretización que contienen todos los posibles puntos de corte y van mezclando dos intervalos y eliminando puntos de corte. El algoritmo de discretización ChiMerge [112] es una conocida técnica bottom-up.

- *Condición de Parada*: Esta característica está relacionada con el mecanismo utilizado para finalizar el proceso de discretización que debe de ser especificado en enfoques no parámetricos. Los criterios propuestos como condición de parada más conocidos suelen ser la medida de descripción de longitud mínima (MDLP), [74], los umbrales de confianza, [112], o los ratios de inconsistencia, [57].

- *Disjunta/No Disjunta*: Las técnicas de discretización disjuntas discretizan el rango de valores de los atributos en intervalos disociados, sin superponerse, mientras que por el contrario las técnicas de discretización no disjuntas discretizan el rango de valores en intervalos superpuestos. Las técnicas hasta el momento citadas son disjuntas mientras que técnicas no disjuntas son aquellas que llevan a cabo una discretización fuzzy, [43, 104, 126].

- *Ordinal/Nominal*: Antes de detallar esta característica, hay que definir varios conceptos relacionados con la misma. El primer concepto se refiere a los datos cuantitativos. Estos datos son valores numéricos en su naturaleza. Otro concepto son los datos cualitativos ordinales, que se refieren a datos categóricos pero con un orden en su significado. Por último, los datos cualitativos son datos categóricos donde no importa el orden de los mismos. Así, un algoritmo de discretización será ordinal cuando transforme los datos cuantitativos en datos cualitativos ordinales y será nominal cuando los datos cuantitativos sean transformados en datos cualitativos.

Al comentar las características que pueden tener las técnicas de discretización, hemos indicado técnicas que crean una discretización crisp, es decir, hacen uso de la lógica clásica y discretizan el dominio de los atributos en intervalos clásicos. También hemos indicado que algunas técnicas realizan una discretización fuzzy utilizando para ello la lógica fuzzy. Por lo tanto, otra característica para categorizar las técnicas de discretización consiste en determinar el tipo de particiones que generan, si son particiones crisp o por el contrario son particiones fuzzy. Dado que la gran mayoría de las técnicas comentadas hasta ahora generan particiones crisp, vamos a centrarnos a continuación en aquellas técnicas que discretizan los atributos numéricos en particiones fuzzy. A continuación, presentamos de forma no exhaustiva algunas

de estas técnicas agrupándolas de acuerdo al algoritmo que toman como base para llevar a cabo la discretización:

- *Técnicas basadas en árboles de decisión*

  Un algoritmo de particiones basado en árbol de decisión se presenta en [110]. En este trabajo se propone la creación de particiones fuzzy jerárquicas basadas en la descomposición de $2^{|A|}$-árboles de decisión, donde $|A|$ es el número de atributos. Esta descomposición es controlada por el grado de certeza de la regla que se genera en cada subespacio fuzzy y por nivel jerárquico más profundo permitido. Las particiones fuzzy formadas para cada dominio son simétricas y triangulares.

  En [154], se propone una técnica para construir particiones fuzzy utilizando un árbol de decisión que se caracteriza por no usar en cada nodo todos los ejemplos que contiene. En lugar de usar todos los ejemplos, selecciona aleatoriamente un subconjunto de estos, para obtener los puntos de corte para dividir el nodo y para crear la partición. Después, aplicando el algoritmo de clustering c-medias, [18], al conjunto de puntos de corte obtenidos, se seleccionan los puntos de corte más representativos y se forman las particiones fuzzy correspondientes.

  Otra forma de discretizar atributo numéricos por medio de conjuntos fuzzy se presenta en [165]. En este trabajo se obtienen como puntos de corte para los atributos, aquellos que maximizan la ganancia de información y a partir de estos puntos de corte se forman las particiones fuzzy, donde el solapamiento de estos se determina de acorde a una incertidumbre asociada con el atributo, es decir, un atributo con una gran incertidumbre tendrá un solapamiento más amplio.

  Otra técnica que utiliza un árbol de decisión se describe en [170]. Una particularidad de esta técnica es que se centra en aquellos conjuntos de datos donde el número de ejemplos es mucho menor en comparación con el número de atributos y el número de clases. Para tratar con estos tipos de conjuntos de datos, utilizan el muestreo para generar los puntos de corte. Así, en este trabajo se hace uso de un árbol de decisión fuzzy y como técnica de remuestreo un bootstrap ordinario, [72]. El añadir el bootstrap al árbol de decisión permite obtener unas particiones más globales y una mejor estimación de todos los valores.

- *Técnicas basadas en clustering*

  Las técnicas de este grupo están basadas en la discretización del dominio de un atributo numérico mediante particiones fuzzy utilizando algoritmos de clustering. Una de las

técnicas más ampliamente utilizada para el clustering fuzzy es el algoritmo fuzzy de las c-medias (FCM), [19]. Este algoritmo asigna un conjunto de ejemplos caracterizados por sus respectivos atributos, a un conjunto de clusters. Cada ejemplo tiene un grado de pertenencia a cada cluster, representado por el centro del cluster. Algunas de las técnicas de clustering fuzzy desarrolladas para crear una discretización fuzzy comienzan con el algoritmo FCM y después añaden alguna heurística o extensión para optimizar las particiones fuzzy, [147, 155]. En [114] se describe un enfoque de clasificación novedoso que integra las reglas de asociación de clase fuzzy y las máquinas de soporte vectorial. Además en este enfoque se desarrolla una técnica de discretización fuzzy basada en el algoritmo FCM, que es empleada para transformar un conjunto de entrenamiento, con atributos numéricos, a un formato más apropiado para las reglas de asociación. Además, para ajustar los umbrales se utiliza un procedimiento de ascensión de colinas. La particularidad de este trabajo es que trabaja con conjuntos de datos de microarray, los cuales se caracterizan por tener el número de atributos mucho mayor que el número de ejemplos.

- *Técnica basadas en algoritmos genéticos*

  Los algoritmos genéticos surgieron como una herramienta para solucionar problemas de búsquedas complejas y de optimización, [87]. Estos algoritmos son combinados con otras técnicas existentes para crear algoritmos híbridos que mejoran los resultados esperados. En referencia a la discretización fuzzy, en [58] se utiliza un algoritmo genético para crear particiones fuzzy. En este trabajo toman como objetivo optimizar el número de particiones, los límites de las particiones y el grado de solapamiento entre ellas.

  Otra técnica para crear particiones fuzzy con funciones triangulares, se propone en [166]. El proceso de construcción de las particiones fuzzy se divide en dos etapas: en la primera etapa se construyen las funciones triangulares y en la segunda, estas funciones son ajustadas con un algoritmo genético.

- *Técnicas híbridas*

  Aparte de crear particiones con algoritmos concretos, también podemos encontrar técnicas en la literatura que discretizan el dominio de los atributos numéricos mediante la combinación de dos o más algoritmos. Un ejemplo se presenta en [209], donde se describe una técnica que combina un algoritmo de clustering con una red neuronal. Más concretamente, utiliza la red de clustering de Kohonen fuzzy (FKCN) integrada con un modelo FCM para el aprendizaje y la modificación de las estrategias de la red de Kohonen.

Otra técnica que combina dos algoritmos es propuesto en [176]. En este trabajo se combina el algoritmo de clustering FCM, con un algoritmo genético para determinar de forma automática el número óptimo de particiones para cada atributo. Esta técnica llamada Fuzzy-VGAPS, no utiliza la distancia euclídea en el algoritmo FCM, sino que utiliza una distancia basada en simetría. Además, en lo referente al algoritmo genético, cada cromosoma está compuesto de números reales que indican los centros de los clusters y la función fitness se define por medio de la evaluación del cromosoma utilizando el algoritmo FCM.

Otra técnica basada en el algoritmo FCM, se presenta en [125], donde se combina el algoritmo FCM con dos variantes del mismo, el PFCM [89], y el AFCM [204]. Además, estos algoritmos de clustering fuzzy son inicializados utilizando diferentes técnicas de inicialización, en concreto las técnicas CCIA [113], kd-tree [172] y MST [199]. El resultado de la combinación de todas las técnicas es un conjunto de particiones fuzzy obtenidas mediante un consenso que utiliza el voto mayoritario fuzzy y el algoritmo de k-vecinos más cercanos. Otra técnica similar a la descrita se presenta en [126], donde el método de consenso utilizado es el voto mayoritario ponderado.

Tras exponer la taxonomía en la cual podemos categorizar una técnica de discretización y tras citar algunas técnicas entre la gran variedad de técnicas de discretización presentes en la literatura, si debemos trabajar con un algoritmo de minería de datos que necesita una discretización previa de los datos numéricos, ¿qué criterios debemos de tener en cuenta a la hora de elegir un algoritmo de discretización de entre los algoritmos presentes en la literatura?

Para responder a esta pregunta, bien porque queremos diseñar o bien seleccionar un algoritmo de discretización, debemos de seguir los siguientes criterios para comparar los algoritmos de discretización y decidir el mejor según las necesidades del algoritmo de minería de datos, [80]:

- Número de intervalos: Una característica deseable para una discretización práctica es que la discretización de atributos tenga el menor número de valores posible, ya que un gran número de intervalos puede hacer el aprendizaje más lento e ineficaz, [52].

- Inconsistencia: Una medida supervisada utilizada para calcular el número de errores inevitables producidos por el conjunto de datos. Un error inevitable es aquel que asocia dos ejemplos con los mismos valores de entrada pero con diferentes valores en la etiqueta clase. Generalmente, los conjuntos de datos con atributos numéricos son consistentes, pero cuando se aplica un proceso de discretización se puede obtener un conjunto de datos inconsistentes. El nivel de inconsistencia deseado es 0.

- Tasa de errores en clasificación: Un algoritmo de éxito será capaz de discretizar un conjunto de datos de entrenamiento sin reducir significativamente la capacidad de predicción en los algoritmos de minería de datos aplicados posteriormente.
- Tiempo: En un proceso de discretización estático los datos de entrenamiento sólo son evaluados una vez, por lo que el método de evaluación no parece ser un método muy importante. Sin embargo, si el proceso de discretización tarda demasiado, el tiempo puede llegar a ser poco práctico en aplicaciones reales. En los procesos de discretización dinámicos la operación se repite tantas veces como el algoritmo requiera, por lo que debe de ser implementado de manera eficiente.

Tomando en cuenta estos criterios podemos seleccionar la técnica que más se adapte a las necesidades presentes en un problema dado. Además de estos criterios que hemos comentado, otro aspecto importante que debemos de tener en cuenta es el tipo de datos presente en el conjunto de datos a discretizar. Es conocido que si un conjunto de datos contiene información imperfecta, ésta suele ser ignorada por las técnicas tradicionales. Sin embargo, este tipo de información cada vez aparece con más frecuencia en los problemas del mundo real y son pocas las técnicas de discretización que pueden trabajar de forma directa con dicha información. En la siguiente sección nos vamos a centrar en aquellas técnicas que llevan a cabo el proceso de discretización trabajando de forma explícita con algún tipo de LQD.

4.2.3 **Técnicas de discretización desde datos de baja calidad**

En este apartado vamos a describir dos técnicas de discretización que trabajan de forma explícita con LQD.

En el primer trabajo, [104], se desarrolla una técnica de discretización de atributos numéricos basándose en un árbol de decisión fuzzy. Esta técnica de discretización crea un árbol de decisión fuzzy al mismo tiempo que construye sus correspondientes particiones fuzzy. El algoritmo puede crear particiones fuzzy sólo de un conjunto de atributos numéricos siempre y cuando para el resto de atributos numéricos que no se quieran discretizar el usuario aporte una partición fuzzy. Durante la construcción del árbol de decisión fuzzy, se van considerando en primer lugar aquellos nodos que contienen más ejemplos, ya que la base de la construcción de las particiones fuzzy se basa en el punto de corte que divide cada nodo, siguiendo la filosofía de un árbol de decisión C4.5. Entre los tipos de LQD con los cuales puede trabajar esta técnica de discretización están los valores missing y los valores fuzzy. Los valores fuzzy deberán de estar representados por el valor de la etiqueta lingüística que represente al valor fuzzy en la partición asociada, es decir, solamente permite valores fuzzy que se encuentren en la partición fuzzy proporcionada por el usuario o creada previamente con dicha técnica.

El segundo trabajo que lleva a cabo la construcción de particiones fuzzy se presenta en [180]. En este caso, las particiones fuzzy se obtienen dentro de un proceso de selección de atributos, trabajando con LQD tales como datos intervalares y fuzzy. Las particiones fuzzy que se crean en este trabajo son utilizadas en un sistema basado en reglas fuzzy que utiliza la información mutua como medida. Según este trabajo, la mejor discretización fuzzy de una variable de entrada en un sistema basado en reglas fuzzy, desde el punto de vista de la información mutua, es la que maximiza la dependencia entre la entrada fusificada y la variable de salida, de forma que la partición pierda la menor información posible en el proceso de discretización. Para crear las particiones utilizan una versión modificada el algoritmo genético multiobjetivo NSGA-II, [178], donde la función que permite al algoritmo trabajar con los LQD es la función fitness del algoritmo, donde hacen uso de la información mutua fuzzy para tratar con dichos tipos de datos.

En las siguiente sección vamos a describir el proceso de selección de atributos, otro proceso importante dentro de la fase del preprocesado de datos. Al igual que para la discretización, vamos a presentar la categorización de las diferentes técnicas que llevan a cabo la selección de atributos y finalmente nos centraremos en aquellos trabajos presentes en la literatura que trabajar con LQD.

4.3 Selección de atributos

En el mundo real, se suelen representar datos utilizando muchos atributos, pero sólo unos pocos serán finalmente relacionados con el concepto de destino, [118]. En muchas aplicaciones del AID, el disponer de una gran cantidad atributos supone un alto coste computacional, así como el posible riesgo de que las técnicas sufran de sobreaprendizaje. Por estas razones entre otras muchas, otro proceso inmerso en la fase de preprocesamiento de datos es la selección de atributos, [92, 97, 201]. Este proceso aborda el problema de reducción de dimensionalidad mediante la determinación de un subconjunto de atributos disponibles, los cuales son los más esenciales para clasificación.

Existen dos enfoques diferentes a la hora de reducir la dimensionalidad de los atributos de un conjunto de datos. Uno de los enfoques es relativo a la extracción de atributos y el otro es relativo a la selección de atributos. La extracción de atributos implica una transformación lineal o no lineal de los atributos del espacio original a un espacio nuevo de menor dimensión. A pesar de que se reduce la dimensionalidad de los atributos con los cuales el clasificador debe tratar, el número de atributos a medir son los mismos. Sin embargo, la selección de atributos reduce directamente el número de atributos originales al seleccionar un subconjunto de éstos que se consideran los más importantes para la clasificación, [134].

Si nos centramos en la selección de atributos, el objetivo es eliminar aquellos atributos irrelevantes o que introduzcan ruido al conjunto de datos, siendo deseable que la técnica de selección de atributos pueda tratar con LQD para evitar perdidas de información al ignorarlos o al tener que transformarlos en datos crisp. En los siguientes apartados vamos a definir más formalmente el concepto de selección de atributos, vamos a mostrar una taxonomía de las diferentes técnicas que llevan a cabo la selección de atributos y nos vamos a centrar en presentar algunas de las técnicas presentes en la literatura que realizan el proceso de selección de atributos y que pueden tratar con información de baja calidad de forma explícita en los conjuntos de datos.

4.3.1 El concepto de selección de atributos

La reducción de la dimensionalidad de los atributos de un conjunto de datos es considerablemente importante en el mundo del AID como ya hemos comentado. La razón para ser tan importante es doble. Por un lado la reducción de la complejidad computacional y por otro la mejora de las habilidades generales de los clasificadores. La primera motivación es evidente, ya que con un menor número de atributos el tiempo de computación para construir modelos es menor. La segunda razón indica que cuando se disponen de pocas dimensiones se reduce el riesgo de sobreaprendizaje. Como regla general, se requieren un mínimo de $10 \cdot |A| \cdot |C|$ ejemplos de entrenamiento para un problema de clasificación con $|A|$ dimensiones y $|C|$ clases, [103]. Cuando es prácticamente imposible obtener el número requerido de ejemplos de entrenamiento, la reducción de atributos ayuda a disminuir el tamaño de los ejemplos de entrenamiento requeridos y, por consiguiente, a mejorar el rendimiento de forma general en el algoritmo de clasificación. Así, la selección de atributos se define como el proceso para identificar y eliminar tantos atributos irrelevantes e innecesarios como sea posible.

El problema de selección de atributos podría describirse como sigue. Dada una serie de atributos en un conjunto de datos, ¿cómo seleccionamos los atributos más importantes con el fin de reducir el número de éstos y al mismo tiempo conservar en la medida de lo posible toda la información de predicción? La respuesta a la pregunta consiste en definir un proceso llamado selección de atributos cuyo objetivo es eliminar aquellos atributos menos interesantes desde el punto de vista de una tarea a realizar, [193]. La selección de atributos es esencial en el modelado y la identificación. Para construir un modelo es necesario tener un conjunto de datos de entrenamiento, descrito a través de atributos. Hay que destacar que puesto que estamos hablando desde un punto de vista predictivo, cada ejemplo debe de ir etiquetado con una clase $c \subset C$, siendo C el conjunto de clases posibles del conjunto de datos. El modelo que se obtiene a través de un clasificador se encuentra inherentemente determinado por los valores de

los atributos de los ejemplos, por lo tanto hay que definir una buena medida de evaluación a la hora de determinar la relevancia o importancia de un atributo. Desde un enfoque estadístico un determinado atributo x_a puede decirse que es relevante si no es estadísticamente independiente de C. Desde un enfoque de selección de modelos un determinado atributo x_a es relevante si al construir un modelo f usando x_a el modelo muestra un buen rendimiento, es decir, el valor de la medida de error usada es mínimo.

En resumen, una técnica de selección de atributos determina cómo de relevante es un subconjunto de atributos s para una tarea y, (generalmente la tarea es de clasificación o de aproximación de datos).

Tras definir el concepto de selección de atributos, en la siguiente sección vamos a presentar las diferentes taxonomías que nos encontramos en la literatura para clasificar a las técnicas de selección de atributos.

4.3.2 **Categorización de las técnicas de selección de atributos**

En presencia de cientos o miles de atributos, es posible que un gran número de éstos no sean de carácter informativo, sino que sean irrelevantes o redundantes con respecto al concepto de clase, [193]. En otras palabras, el aprendizaje se puede lograr con mayor eficiencia y eficacia sólo con los atributos relevantes y no redundantes. Sin embargo, el número de subconjuntos de atributos posibles crece exponencialmente con el aumento de la dimensionalidad. Encontrar un subconjunto óptimo suele ser un problema intratable. Además, muchos problemas relacionados con la selección de atributos son NP-Completos, [116]. Se han estudiado diversos aspectos de la selección de atributos. Uno de los aspectos clave es disponer de una medida que indique la bondad de un subconjunto de atributos para intentar determinar el subconjunto óptimo de éstos, [131]. Por lo tanto, una manera de clasificar las técnicas de selección de atributos es utilizar el criterio de evaluación empleado. Así, dependiendo del criterio de evaluación utilizado, las técnicas de selección de atributos se pueden dividir en las siguientes categorías, [193]:

- Las técnicas de filtrado: Estas técnicas utilizan una medida externa como criterio de evaluación de la calidad de los subconjuntos de atributos. Las técnicas de filtrado seleccionan un subconjunto de atributos en una etapa de preprocesamiento, independientemente del predictor elegido.

- Técnicas wrapper: En este caso, se utiliza la precisión de clasificación para evaluar subconjuntos de atributos. Las técnicas wrapper utilizan el algoritmo de aprendizaje como una caja negra para anotar los subconjuntos de atributos de acuerdo a su poder predictivo.

- Las técnicas integradas: En estas técnicas la selección de atributos se lleva a cabo en el proceso de entrenamiento y suelen ser específicos de la técnica de minería de datos empleada.

- Las técnicas híbridas: Estas técnicas son una combinación de técnicas de filtrado y técnicas wrapper. Las técnicas híbridas utilizan el ranking de información obtenido mediante las técnicas de filtrado para guiar la búsqueda en los algoritmos de optimización utilizados por las técnicas wrapper.

Otra forma de categorizar a las técnicas de selección de atributos es según las estrategias de búsqueda empleadas en las mismos. Así, en [152] se muestra que las estrategias más utilizadas en las técnicas de selección de atributos son:

- Selección hacia delante: Se comienza con un conjunto vacío de atributos y se van añadiendo.

- Eliminación hacia atrás: Se comienza con todos los atributos y se van eliminando aquellos que no son necesarios.

- Selección por pasos hacia delante: Se comienza con un conjunto vacío de atributos y éstos se van añadiendo o eliminando uno a uno.

- Eliminación por pasos hacia atrás: Se comienza con todos los atributos y éstos se van añadiendo o eliminando uno a uno.

- Mutación aleatoria: Se comienza con un conjunto de atributos seleccionados aleatoriamente. Después, estos atributos seleccionados aleatoriamente se eliminan o añaden uno a uno hasta que el proceso se detiene tras un número de iteraciones.

A continuación, presentamos algunos ejemplos de técnicas que incorporan estas estrategias de búsqueda para seleccionar atributos. Por ejemplo, en [134] se detalla un algoritmo de selección de atributos que utiliza una version más eficiente de la técnicas wrapper de búsqueda secuencial hacia delante (SFS) para máquinas de soporte vectorial, el cual es llamado soporte y filtrado de búsqueda secuencial hacia delante (FS_SFS). FS_SFS es una técnica diseñada especialmente para máquinas de soporte vectorial y tiene las siguientes propiedades sobre las técnicas convencionales wrapper/SFS: FS_SFS combina las ventajas de las técnicas de filtrado y las técnicas wrapper, puesto que introduciendo un filtro en cada iteración se reduce el número de atributos a validar. Otra propiedad es que FS_SFS introduce un nuevo criterio que valora los atributos de una manera colectiva. La última propiedad es que la técnica está diseñada especialmente para máquinas de soporte vectorial para mejorar la eficiencia en el proceso de selección

de atributos. El tema de la eficiencia durante este proceso es importante porque hay máquinas de soporte vectorial las cuales tienen un coste cuadrático.

En [90], se propone también una técnica para tratar con máquinas de soporte vectorial y llevar a cabo el proceso de selección de atributos, donde se van eliminado recursivamente atributos en base a un criterio de ranking de atributos el cual se realiza a través de las magnitudes de los pesos.

Otro estudio donde también hacen uso la técnica SFS se presenta en [14]. En dicha técnica se utiliza como medida la información mutua entre un atributo y la clase del ejemplo y entre cada par de atributos. En cada paso, el mejor atributo evaluado es seleccionado para formar parte del subconjunto si su valor de bondad es más bajo que una bondad predefinida. Una aplicación para seleccionar atributos similar a la anterior es propuesta en [213].

Otra técnica SFS es presentada en [141] donde el algoritmo de clustering fuzzy FCM es usado para seleccionar atributos. Este algoritmo se basa en el índice de discriminación de un atributo con respecto al prototipo del cluster. El atributo con más alto índice es incluido en el subconjunto de los seleccionados.

También es posible usar la metodología contraria a la SFS que es la SBS, búsqueda secuencial hacia atrás. En [164] se presenta un algoritmo para seleccionar atributos que utiliza la metodología SBS, donde se hace uso de un algoritmo de clustering en el cual cada atributo es indexado acorde a su importancia. La importancia es evaluada como la diferencia entre la distancia Euclídea de los ejemplos al cluster teniendo y sin tener en cuenta el atributo. Cuanto más grande sea dicha diferencia más importante es el atributo.

Además de por las estrategias de búsqueda, también podemos categorizar la selección de atributos por su ámbito de aplicación. Las técnicas de selección de atributos se han usado y se usan actualmente en varios ámbitos.

Uno de los ámbitos de aplicación consiste en utilizar la selección de atributos para la categorización de texto. El problema de la categorización de texto es la alta dimensionalidad del espacio de atributos. La mayoría de estas dimensiones no son relativas a la categorización de texto, sino que muchos atributos pueden introducir ruido que puede afectar al rendimiento de un clasificador. En [1], se propone la optimización de una colonia de hormigas para reducir la dimensionalidad de los atributos de un texto.

Otro ámbito donde la selección de atributos es muy aplicada es en biología, donde el análisis de microarrays o el análisis de datos biológicos requieren de una selección de atributos previa para seleccionar los genes más relevantes. Los datos de microarrays normalmente contienen miles de genes y un número pequeño de ejemplos y la mayoría de dichos genes son irrelevantes o insignificantes para el diagnóstico clínico. En [124] se lleva a cabo el desarrollo

de un algoritmo híbrido para seleccionar atributos en el análisis de datos de microarrays. El algoritmo está compuesto de dos etapas. En una primera etapa se usa un algoritmo genético con configuración de parámetros dinámicos, para generar un número de subconjuntos de genes y un ranking de importancia de los mismos de acuerdo a la frecuencia con la que aparece cada gen en los subconjuntos. En la segunda etapa se usa el test de homogeneidad χ^2 para determinar un umbral que indique, según la frecuencia de las ocurrencias de los genes, el número de genes seleccionados partiendo de los genes generados en la primera etapa. Una vez que se obtienen el conjunto de genes seleccionados, para validar la efectividad de dichos genes se utiliza una máquina de soporte vectorial como test.

Otro algoritmo usado para la selección y clasificación de microarrays es el presentado en [65]. En este trabajo se utiliza el ensamble random forest, [26], para llevar a cabo el proceso de selección de atributos. Random forest es un multiclasificador formado por un conjunto de árboles de decisión. Para construir cada árbol de decisión se toma una muestra aleatoria con reemplazamiento del conjunto de ejemplos y los ejemplos no seleccionados forman parte de un subconjunto llamado OOB. La selección de atributos con este multiclasificador es llevada a cabo utilizando el conjunto de ejemplos OOB como medida para saber si un atributo es relevante o no. La medida consiste en clasificar cada árbol usando el conjunto OOB como test, y después permutar los valores de los atributos uno a uno y clasificar de nuevo. La diferencia en clasificación indica la relevancia de un atributo. Cuanto mayor sea esa diferencia más relevante es el atributo en cuestión.

Otro trabajo que también utiliza random forest para seleccionar atributos se presenta en [82]. En este trabajo se plantean dos objetivos. El primero de ellos es encontrar un conjunto de atributos altamente relacionado con el atributo de salida al propósito de la interpretación. El segundo consiste en encontrar un conjunto pequeño de atributos lo suficientemente buenos para una predicción moderada del atributo de salida.

Sin centrarnos en las posibles aplicaciones, también podemos clasificar las técnicas de selección de atributos en función de la técnica que utilizan. Por ejemplo, en [140] se describe una técnica para seleccionar atributos utilizando tres medidas diferentes de la entropía fuzzy y un clasificador de similitud. Esta técnica a grandes rasgos se puede dividir en tres fases. La primera fase consiste en utilizar el principio básico de un clasificador de similitud que es crear un vector ideal $v_c = (v_c(a_1), ..., v_c(a_{|A|}))$. Hay un vector v_c para cada clase c y este vector indica para cada atributo a la representación que tiene a para cada clase c. Una segunda fase calcula para cada ejemplo de entrenamiento, el cual definimos como $e = (x(a_1), ..., x(a_{|A|}))$, la similitud de dicho ejemplo con cada vector v. Para calcular la similitud de un ejemplo e con un vector v_c se calcula la similitud entre cada par de atributos $v_c(a)$ y $x(a)$. A esta similitud entre cada par de atributos se la denomina $\mu_F(x_a)$, siendo F un conjunto fuzzy. La última fase

de este algoritmo consiste en llevar a cabo la selección de atributos tras calcular la similitud entre cada par de atributos del vector ideal y un ejemplo de entrenamiento, usando las medidas de entropía que proponen. Para cualquier medida de entropía fuzzy usada, es necesario calcular el valor de pertenencia de x_a a algún conjunto fuzzy ($\mu_F(x_a)$). La medida de entropía fuzzy de todas los atributos de todos los ejemplos de entrenamiento proporcionan $|A|$ medidas de entropía. Es en este caso donde los atributos que tengan un menor valor de entropía son descartados por ser menos relevantes para la clasificación.

Podemos encontrar varias técnicas que llevan a cabo la selección de atributos utilizando la técnica de optimización de las colonias de hormigas (ACO). Por ejemplo, en [108] se detalla un algoritmo para seleccionar atributos que hace uso de una optimización híbrida de una colonia de hormigas, ACOFS. Para seleccionar los atributos más relevantes se utiliza un algoritmo ACO y una estrategía híbrida de búsqueda en el espacio de atributos. El principal foco de este algoritmo es que genera subconjuntos de atributos de salida de tamaño reducido. La estrategia híbrida combina los enfoques de las técnicas de filtrado y de las técnicas wrapper. El algoritmo ACO modifica la actualización de feromona estándar y la medición de la información heurística mediante dos enfoques. El primero de ellos es relativo a que no se enfatiza solamente la selección del número de atributos de salida, sino que también se valora el logro o rendimiento al reducir el número de ellas. El segundo enfoque es que ACOFS utiliza una búsqueda híbrida para seleccionar los atributos de salida combinando las ventajas de los enfoques de filtrado y wrapper. Este algoritmo entra en contraste con otros algoritmos existentes en la literatura que también utilizan la optimización de colonia de hormigas como técnica para seleccionar atributos, [109, 111]. En estos últimos trabajos se intenta diseñar reglas sin distinguir entre un comportamiento aleatorio y uno probabilístico de las hormigas durante la construcción de un subconjunto de atributos y consecuentemente las hormigas pueden ser privadas de la oportunidad de utilizar suficiente experiencia previa o de investigar más atributos de salida durante la construcción de sus soluciones.

En [193] se propone usar la optimización de colonia de hormigas para seleccionar atributos. El número de los mejores atributos es determinado automáticamente. Este algoritmo considera dos objetivos, por un lado reducir el número de atributos y por otro minimizar el error en clasificación. Este enfoque considera dos colonias de hormigas que cooperan entre sí para optimizar cada objetivo. La primera colonia determina el número de atributos, es decir, la cardinalidad, y la segunda colonia selecciona los atributos utilizando la cardinalidad proporcionada por la primera colonia de hormigas. Así, es necesario tener dos matrices de feromona y dos heurísticas diferentes. La heurística utilizada para la primera colonia de hormigas es la propuesta en [194] y con ella se determina la cardinalidad de los atributos. El otro valor heurístico es computado

usando el criterio de discriminación de Fisher para selección de atributos [70]. Un último aspecto a destacar es que para calcular el error en clasificación se hace uso de un clasificador fuzzy.

Otra técnica bastante conocida para seleccionar atributos es la propuesta en [115] donde se propone una técnica de filtrado llamada Relief que utiliza una red neuronal y la ganancia de información para seleccionar un conjunto de atributos. En [51] también se hace uso de una red neuronal aunque se utiliza para evaluar el subconjunto de atributos que previamente un algoritmo genético ha seleccionado.

También es posible en un mismo algoritmo seleccionar atributos y ejemplos simultáneamente. En [95] y [64] se describen dos algoritmos que seleccionan atributos y ejemplos al mismo tiempo. En ambos algoritmos se hace uso del termino vecindad. Una de las principales diferencias es que en [95], el algoritmo es desarrollado para ser aplicado a máquinas de soporte vectorial e introduce el concepto de vecindad basándose en conjuntos de modelos rough para seleccionar atributos y por su parte en [64], se utiliza la metodología vecino más cercano como clasificador, ya que para seleccionar ejemplos y atributos se hace uso de un algoritmo evolutivo.

Hasta el momento todos los algoritmos presentados llevan a cabo la selección de atributos sin prestar especial atención al tipo de datos que componen los conjuntos de datos. En el siguiente apartado, vamos a centrarnos en aquellas técnicas que encontramos en la literatura capaces de trabajar directamente con LQD y que como veremos constituyen un grupo bastante reducido.

4.3.3 **Selección de atributos desde datos de baja calidad**

Analizando globalmente las técnicas que hemos descrito en el apartado anterior, aunque muchas de ellas utilizan la teoría de los conjuntos fuzzy en sus desarrollos, no tienen en cuenta durante el proceso de selección de atributos los posibles LQD que pueden aparecer en los conjuntos de datos. En este apartado vamos a presentar las técnicas existentes en la literatura que permiten en cierto grado trabajar con LQD.

Un trabajo que maneja LQD lo encontramos en [198], donde se presenta una técnica de selección de atributos a través de los conjuntos fuzzy-rough. Esta técnica emplea este tipo de conjuntos para proporcionar un medio por el cual los datos discretos o numéricos con ruido o la mezcla de ambos pueden ser efectivamente reducidos sin la necesidad de que el usuario aporte información adicional. Además, esta técnica puede ser aplicada a datos con atributos de decisión nominales o numéricos, llevando así a cabo tanto regresión como clasificación

de conjuntos de datos. La única información requerida por este algoritmo es un conjunto de particiones fuzzy para cada atributo.

Una medida utilizada para llevar a cabo el proceso de selección de atributos es la información mutua, como ya hemos visto en el apartado anterior. Hay dos trabajos donde se hace uso de esta medida y además se trabaja directamente con conjuntos de datos que contienen LQD. Concretamente en [180], se extiende la medida de la información mutua a la medida de la información mutua fuzzy entre dos atributos numéricos fusificados para poder manejar la imprecisión de los datos. Además, en este trabajo la medida es utilizada en combinación con una extensión del algoritmo genético multiobjetivo NSGA-II, [61], para definir una técnica de selección de atributos que trabaja directamente con LQD.

Otro trabajo donde también se utiliza la información mutua fuzzy se describe en [186], donde además de utilizar la información mutua fuzzy como medida, se extiende el algoritmo de filtrado propuesto en [14] para que la técnica pueda seleccionar atributos a partir de conjuntos de datos que contienen información de baja calidad.

Por último, otra técnica que también trabaja con LQD se propone en [212]. En el trabajo se presenta un estudio teórico para seleccionar atributos en un sistema de decisión fuzzy. La propuesta realizada se basa en la generalización de la teoría de la evidencia fuzzy.

Una vez analizados algunas de las técnicas presentes en la literatura que llevan a cabo el proceso de selección de atributos y habiendo presentando las escasas técnicas que son capaces de trabajar directamente con LQD, en la siguiente sección vamos a tratar otro proceso dentro del preprocesamiento de datos, la imputación de valores missing.

4.4 Imputación de valores missing

La adquisición de datos basados en la nuevas tecnologías de alto rendimiento a menudo se enfrenta con el problema de los datos missing. Este tipo de datos aparecen por diferentes razones tales como mal funcionamiento de los equipos, errores de medición, introducción manual de los datos, etc.. La presencia de tales valores missing en los datos requieren una etapa de preprocesamiento en la cual los datos queden preparados y limpios, [97, 167], con el fin de que sean útiles y suficientemente claros para la extracción de conocimiento.

El problema que surge con los datos missing, es que hay una gran cantidad de técnicas de minería de datos que no pueden manejar este tipo de datos, por lo que la manera más simple de tratar con los valores missing es descartar los ejemplos que los contengan. Sin embargo, este procedimiento sólo es práctico cuando el número de valores missing es relativamente pequeño con respecto al número de ejemplos y cuando el análisis de todos los ejemplos no conduzca a un sesgo importante durante la inferencia, [129]. De ahí que cuando el número de valores

missing es considerable en relación al número de ejemplos, haya que aplicar alguna técnica de imputación de valores missing a fin de asignar un valor plausible a dichos valores y así conseguir que las técnicas puedan trabajar con los conjuntos de datos ya sin valores missing.

Al igual que para las técnicas de preprocesamiento de datos expuestas en las anteriores secciones, vamos a realizar un análisis del concepto de imputación, para posteriormente presentar una categorización de estas técnicas y describir algunas de dichas técnicas presentes en la literatura que llevan a cabo el proceso de imputación de valores missing centrándonos en aquellos que trabajan de forma directa con LQD.

4.4.1 El concepto de imputación de valores missing

Los valores missing dificultan el análisis de los datos. La presencia de valores missing puede incluso suponer serios problemas para los investigadores. De hecho un inapropiado manejo de los valores missing en el AID puede introducir sesgos y dar lugar a conclusiones erróneas derivadas de un estudio de investigación. Además también estos valores pueden limitar la generalización de los resultados de investigación, [138, 196]. Para el caso particular de la clasificación, el aprendizaje desde datos missing puede llegar a ser más importante, ya que los datos missing, bien en el conjunto de entrenamiento o en el de test pueden afectar a la precisión de los resultados clasificados, [84]. De forma general, los tres siguientes problemas suelen estar asociados con los valores missing en el AID, [11]: (1) Pérdida de eficiencia, (2) complicaciones en el manejo y en el análisis de datos y (3) resultados sesgados con diferencias entre los datos missing y los datos completos. Estos problemas han sido demostrados mediante varios estudios [137, 139], en los cuales, ciertos algoritmos que obtienen muy buenos resultados trabajando con datos completos, pierden precisión y rendimiento cuando trabajan con datos missing. Un ejemplo de este tipo de algoritmo es el presentado en [139], donde se presenta un sistema de clasificación basado en reglas fuzzy que es capaz de trabajar directamente con ciertos datos de baja calidad obteniendo resultados interesantes. Sin embargo el rendimiento en los resultados de este sistema puede decaer si los conjuntos de datos contienen una gran cantidad de valores missing, de ahí el imputar dichos valores missing para evitar el perjuicio que los valores missing pueden causar.

A continuación y una vez que hemos definido el concepto de imputación de valores missing, vamos a exponer una categorización de las técnicas que llevan a cabo este proceso en la fase de preprocesamiento de datos.

4.4.2 Categorización de las técnicas de imputación de valores missing

De forma general, analizando el tratamiento que hacen las técnicas sobre los valores missing los podemos dividir en tres categorías, [12, 129]:

- Técnicas que ignoran o descartan valores missing. Hay dos maneras principalmente para descartar datos con valores missing. La primera se conoce como análisis de casos completos, está disponible en todos los programas estadísticos y es la técnica por defecto en muchos programas. Esta técnica consiste en descartar todos los ejemplos con datos missing . La segunda técnica es conocida como descarte de instancias y/o atributos. Esta técnica consiste en determinar las instancias y atributos que tengan datos missing y eliminarlos. Antes de eliminar cualquier atributo, es necesario evaluar la relevancia del mismo para el análisis de datos. Por desgracia, los atributos relevantes deben mantenerse incluso con un alto nivel de valores missing. Las dos técnicas deben de aplicarse únicamente si los datos missing son MCAR, porque los datos que no son MCAR tienen elementos no aleatorios que pueden sesgar los resultados.

- Técnicas de estimación de parámetros. Los procedimientos de máxima verosimilitud se utilizan para estimar los parámetros de un modelo definido para los datos completos. Los procedimientos de máxima verosimilitud que utilizan variantes del algoritmo maximización de la esperanza, [62], pueden manejar la estimación de parámetros en presencia de datos missing.

- Técnicas de imputación. La imputación es una clase de procedimiento que consisten en rellenar los valores missing con valores estimados. El objetivo es emplear las relaciones conocidas que puedan ser identificadas en los datos válidos del conjunto de datos para ayudar en la estimación de los valores missing. En otras palabras, la imputación es un concepto que se refiere al procedimiento mediante el cual se sustituyen los valores missing del conjunto de datos por valores plausibles.

En este trabajo nos vamos a centrar en la imputación, puesto que dentro de la minería de datos hay muchos algoritmos que no pueden trabajar directamente con datos missing, de ahí que haya que utilizar técnicas de imputación para tratar con estos datos y adaptar la información a las necesidades de las técnicas de clasificación/regresión. Una ventaja de utilizar la imputación es que el tratamiento de los datos missing es independiente del algoritmo de aprendizaje a utilizar. Esto permite al usuario seleccionar la técnica de imputación más adecuanda en función de cada situación o problema, [138]. Entre toda la variedad de técnicas de imputación, las cuatro opciones más comunes son, [12]:

- Caso de sustitución: Esta opción se utiliza en encuestas por muestreo. Si una instancia tiene datos missing, por ejemplo, por una persona con la que no se pudo contactar, esa instancia se sustituye por otra que no contenga datos missing y que no haya sido ya muestreada.

- La media y/o moda: Esta opción es la que se utiliza con más frecuencia. Consiste en la sustitución de los valores missing por la media para atributos numéricos o por la moda para atributos nominales. Hay alguna variante a esta opción, como por ejemplo, obtener la media solamente de aquellos ejemplos con la misma etiqueta de clase del ejemplo que contiene un valor missing.

- Hot deck y cold deck: En una técnica hot deck, un valor missing se completa con el valor de una distribución estimada para los valores missing de los datos actuales. Este tipo de técnicas se implementan en dos etapas. En una primera etapa los datos se particionan en clusters, mientras que en la segunda cada ejemplo con valor missing se asocia a un cluster. Los ejemplos completos de un cluster se utilizan para completar los valores missing. Esto se puede hacer calculando la media o la moda del atributo dentro del cluster. Las técnicas de imputación cold deck son similares a las hot deck sólo que el origen de los datos debe de ser diferente a los datos actuales.

- Técnicas de predicción. Los algoritmos de predicción suelen ser procedimientos más sofisticados para el manejo de los datos missing. Estas técnicas consisten en la creación de un modelo de predicción para estimar los valores que van a sustituir a los valores missing. El atributo que contiene datos missing se utiliza como atributo a predecir y el resto de atributos se utilizan como entrada para llevar a cabo la predicción.

De las cuatro opciones de imputación posibles nos vamos a centrar en las técnicas de imputación mediante predicción. Dentro de las técnicas de predicción, existen técnicas de imputación individual o múltiple. Las técnicas de imputación individual solamente imputan un valor missing cada vez. Por el contrario, el procedimiento de imputación múltiple sustituye cada valor missing por un conjunto de valores plausibles que representan la incertidumbre sobre el valor correcto a imputar, [214]. La inferencia de la imputación múltiple conlleva tres fases: (1) Los datos missing se completan m veces para disponer de m conjuntos de datos completos; (2) los m conjuntos de datos se analizan mediante el uso de procedimientos estándares y (3) los resultados de los m conjuntos de datos completos se combinan para la inferencia.

Vamos a centramos en las técnicas de imputación individual debido principalmente a la gran cantidad de tiempo computacional que requieren los esquemas de imputación múltiple y

a las hipótesis que estos esquemas hacen respecto a la distribución de los datos y la aleatoriedad de los valores missing. En otras palabras, para aplicar una técnica de imputación múltiple debemos de conocer las distribuciones subyacentes de los datos completos y de los valores missing antes de su aplicación, [138]. Además, en [84] se indica que las técnicas de imputación individual son capaces de mostrar mejores capacidades predictivas que las técnicas de imputación múltiples para un amplio rango de conjuntos de datos debido a que tienen un menor sobreajuste.

A continuación vamos a describir algunas técnicas presentes en la literatura que llevan a cabo el proceso de imputación de valores missing. Por ejemplo, en [181], se hace uso del algoritmo EM (maximización de la esperanza) para imputar los valores missing. En una iteración del algoritmo EM, las estimaciones dadas de las medias y de la matriz de covarianza se revisan en tres pasos. En el primer paso, para cada registro con valores missing, los parámetros de regresión de las variables con valores missing se calculan entre las variables con valor disponible a través de la estimación de la media y la matriz de covarianza. Después en el segundo paso los valores missing se completan con sus valores esperados condicionales. Estos valores esperados condicionales se obtienen con los valores disponibles y las estimaciones de la media y de la matriz de covarianza. En el tercer paso, la media y la matriz de covarianza son reestimadas, la media como la media de las muestras de los datos completos establecidos y la matriz de covarianza como la suma de la matriz de covarianza de la muestra de los datos completos establecidos y una estimación de la matriz de covarianza condicional del error de sustitución. El algoritmo EM se inicia con las primeras estimaciones de la media y de la matriz de covarianza e itera a través de estos pasos hasta que los valores imputados y las estimaciones de la media y de la matriz de covarianza se modifican sensiblemente de una iteración a la siguiente.

Otro trabajo para imputar valores missing se describe en [67]. En este trabajo se investigan ocho patrones de desconocimiento diferentes dependiendo de la relación entre el desconocimiento y tres tipos de variables. Los tres tipos de variables son: los predictores observados, los predictores no observados (los valores missing), y la variable de respuesta. Se centran en el caso de árboles de clasificación para los datos binarios (C4.5 [169] y CART [27]).

En [84], se propone una técnica de imputación individual y una técnica de imputación múltiple, ambas basadas en una generalización de una red neuronal de regresión (GRNN). Dicha red neuronal se basa en el algoritmo presentado en [184]. Estas dos técnicas pueden tratar con conjuntos de datos multidimensionales. Para validar las técnicas los autores las comparan con otras veinticinco técnicas de imputación de diferente naturaleza. Además, los autores llegaron a la conclusión de que la técnica de imputación múltiple obtiene buenos resultados pero es una técnica muy pesada tanto en memoria como en tiempo de computación.

Otro trabajo centrado en la imputación de los valores missing, se presenta en [185]. En este caso, para llevar a cabo la imputación de valores missing se hace uso del ensamble Random Forest. La elección de dicho ensamble según los autores se debe a la capacidad de poder tratar tanto con atributos numéricos como con atributos nominales indistintamente y por tratarse de una técnica no paramétrica que permite efectos interactivos y no lineales (regresión). Para llevar a cabo la imputación utilizan un esquema iterativo, donde en la fase de entrenamiento del Random Forest se lleva la cabo la observación de los valores disponibles, para después llevar a cabo la predicción de los valores missing.

Una de las técnicas por excelencia para imputar valores missing es la técnica de k-vecinos más cercanos (KNN). En la literatura podemos encontrar una gran variedad de procedimientos que con ciertas variaciones imputan valores missing con el algoritmo KNN. Por ejemplo en [13], se utiliza el algoritmo KNN para imputar los valores missing. Cada vez que el algoritmo encuentra un valor missing, calcula sus vecinos más cercanos y le asigna al missing un valor. Para los valores nominales imputa el valor más común entre los vecinos y para los numéricos utiliza el valor medio. Otra técnica que utiliza el algoritmo KNN se presenta en [187]. En este caso se presenta una técnica de imputación ponderada utilizando KNN. La técnica k-vecinos más cercanos ponderada selecciona los ejemplos con valores similares (en términos de distancia) al ejemplo que tiene el valor missing, por lo que imputa como la técnica KNN. Sin embargo, el valor imputado ahora tiene en cuenta las diferentes distancias con los vecinos, utilizando una media ponderada para los numéricos o el valor que más se repite de acuerdo a la distancia para los nominales. La principal diferencia entre las diferentes propuestas para imputar valores missing con la técnica KNN, es la métrica utilizada para calcular las distancias entre los diferentes ejemplos. Una revisión de las técnicas de imputación basadas en KNN es llevada a cabo en [138].

Hasta el momento hemos presentado de forma no exhaustiva algunas de las técnicas presentes en la literatura para llevar a cabo la imputación de valores missing los cuales sólo pueden trabajar con datos crisp. A continuación, nos vamos a centrar en las técnicas para imputar valores missing que sean capaces de manejar LQD. En el siguiente apartado vamos a describir las técnicas que podemos encontrar en la literatura que llevan a cabo el proceso de imputación de valores missing desde LQD y que como veremos son escasos.

4.4.3 **Técnicas de imputación de valores missing desde datos de baja calidad**

Centrándonos en las técnicas que imputan valores missing desde LQD, debemos de indicar que hay una escasa variedad de trabajos en la literatura que puedan llevar a cabo la imputación manejando explícitamente LQD. Un trabajo que es capaz de llevar a cabo el proceso de

imputación de valores missing se presenta en [81]. Esta técnica puede imputar valores missing desde datos de entrada que pueden contener valores fuzzy e intervalares en atributos numéricos y distribuciones de probabilidad en atributos nominales. Esta técnica obtiene un modelo el cual utiliza una extensión del algoritmo EM que toma como base la Teoría de Evidencias de Demspter-Shafer. El modelo es una combinación de gaussianas factorizadas generalizadas que usa probabilidades condicionadas con el fin de imputar los valores missing a partir de otros valores conocidos. Una ventaja de esta técnica es que el modelo obtenido es una probabilidad conjunta de todos los atributos, por lo tanto no es necesario obtener un modelo diferente para cada atributo a imputar. La desventaja de esta técnica es que la convergencia del algoritmo se ve afectada en gran medida por los valores de inicialización utilizados.

Otra de las técnicas que nos encontramos en la literatura para imputar valores missing trabajando con LQD es el árbol de decisión fuzzy presentando en [106]. El árbol de decisión fuzzy presentado en este trabajo es capaz de trabajar con valores fuzzy en atributos numéricos. A la hora de imputar los valores missing, el árbol de decisión fuzzy considera el atributo que tiene valores missing como atributo a predecir. Por lo tanto si hay valores missing en varios atributos, hay que construir un modelo distinto para cada uno de ellos.

En resumen, durante este capítulo hemos presentado los procesos de discretización, selección de atributos e imputación de valores missing dentro de la fase de preprocesamiento de datos del AID. Hemos definido cada uno de estos procesos, hemos categorizado las técnicas que llevan a cabo cada proceso y nos hemos centrado en las técnicas presentes en la literatura que llevan a cabo el proceso desde LQD. Como hemos comentado, aunque cada vez son más las técnicas adaptadas o extendidas para trabajar con LQD, todavía en algunos de los procesos, estas técnicas son bastante escasas y raramente son estudiadas, [180], de ahí que nuestro objetivo en este trabajo sea proponer técnicas dentro de la fase de preprocesamiento de datos que sean capaces de tratar LQD que se encuentren explícitamente en los conjuntos de datos.

CAPÍTULO

5

Minería de Datos

5.1 Introducción

La fase de minería de datos es la más característica y una de las más importantes del AID, [92, 97, 150, 173, 202]. La minería de datos es un campo interdisciplinar cuyo objetivo general es predecir salidas y averiguar las relaciones entre los datos, es decir, el objetivo de esta fase es producir nuevo conocimiento que pueda utilizar el usuario.

A pesar de las multiples definiciones que podemos encontrar en la literatura, [93, 123, 136, 151, 173], sobre el término minería de datos, hay una visión general que las engloba a todas. Así que podemos considerar la minería de datos de forma general como un proceso o una tarea mediante la cual se estudian los datos con el fin de conseguir información adicional a partir de ellos. Dado el crecimiento de los conjuntos de datos en los últimos años se ha creado la necesidad de crear nuevas tecnologías que utilizan la información y la inteligencia con conocimiento, [128]. Por lo tanto, las técnicas de minería de datos que mezclan la información con la inteligencia se han convertido en un área de investigación cada vez más importante.

En los siguientes apartados de este capítulo vamos a describir más detenidamente algunos aspectos de la fase de minería de datos. En concreto, las tareas que resuelve esta fase, centrándonos en la tarea de clasificación que es de mayor interés para este trabajo. Además, describiremos los distintos tipos de técnicas de minería de datos. Por último, dedicaremos una especial atención a aquellas técnicas de minería de datos que pueden trabajar con LQD.

5.2 La fase de minería de datos

Como hemos comentado, la minería de datos a grandes rasgos trata de construir un modelo basado en los datos recopilados para este efecto. El modelo es una descripción de los patrones y relaciones entre los datos que pueden usarse para hacer predicciones, para entender mejor los datos o para explicar situaciones pasadas, [92]. Antes de comenzar con tal proceso es necesario tomar una serie de decisiones, específicamente las siguientes:

- Determinar qué tipo de tarea de minería de datos es la más apropiada para resolver el problema.
- Elegir el tipo de modelo con el que se desea modelar los datos.
- Elegir el algoritmo de minería que resuelva la tarea y obtenga el tipo de modelo que estamos buscando.

Vamos a comenzar describiendo las tareas que se pueden resolver con la minería de datos, centrándonos en la tarea de clasificación objeto de este trabajo.

5.2.1 Tareas de la minería de datos

Teniendo en cuenta la primera decisión a tomar antes de empezar el proceso de minería de datos debemos de definir las diferentes tareas que se encuentran dentro de la minería de datos. Una tarea dentro de la minería de datos puede considerarse como un tipo de problema a resolver por un algoritmo. Las tareas que nos encontramos en la minería de datos pueden ser predictivas o descriptivas. Entre las tareas predictivas encontramos la clasificación y la regresión y entre las tareas descriptivas se encuentran el agrupamiento o clustering, las correlaciones, las reglas de asociación y las reglas de asociación secuenciales, [202]. A continuación, vamos a analizar brevemente cada una de estas tareas:

- En la tarea de clasificación cada ejemplo pertenece a una clase expresada mediante un atributo. El resto de atributos del ejemplo se utilizan para predecir la clase. El objetivo de la clasificación es predecir la clase de nuevos ejemplos a partir de la información proporcionada por el resto de atributos. Más concretamente, el objetivo del algoritmo empleado para llevar a cabo la clasificación es maximizar la razón de precisión de la clasificación de los nuevos ejemplos, calculada como el cociente entre las predicciones correctas y el número total de predicciones.

- La regresión consiste en aprender una función real que asigna a cada ejemplo un valor real, es decir, el valor a predecir es numérico. En este caso se trata de minimizar el error entre el valor predicho y el valor real de los ejemplos.

- El agrupamiento (clustering) trata de obtener grupos naturales a partir de los datos. Hablamos de grupos y no de clases, porque a diferencia de la clasificación, en lugar de analizar datos etiquetados con una clase, los datos son analizados para generar una etiqueta. Los datos son agrupados basándose en el principio de maximizar la similitud entre los ejemplos de un grupo minimizando la similitud entre los distintos grupos. Es decir, se forman grupos tales que los objetos de un mismo grupo son muy similares entre sí y, al mismo tiempo, son muy diferentes a los objetos de otro grupo.

- Las correlaciones se usan para examinar el grado de similitud de los valores de dos variables numéricas. Una fórmula estándar para medir la correlación lineal es el coeficiente de correlación r, el cual es un valor real entre -1 y 1. Si r es 1 (respectivamente -1) las variables están perfectamente correlacionadas (perfectamente correlacionadas negativamente), mientras que si es 0 no hay correlación. Esto quiere decir que cuando r es positivo, las variables tienen un comportamiento similar (ambas crecen o decrecen al mismo tiempo) y cuando r es negativo si una variable crece la otra decrece.

- Las reglas de asociación son también una tarea descriptiva, muy similar a las correlaciones, que tiene como objetivo identificar relaciones no explícitas entre atributos nominales. Pueden ser de muchas formas, aunque la formulación más común es del estilo "si el atributo X toma el valor d entonces el atributo Y toma el valor b". Las reglas de asociación no implican una relación causa-efecto, es decir, puede no existir una causa para que los datos estén asociados.

- Las reglas de asociación secuenciales son un caso especial de reglas de asociación y se usan para determinar modelos secuenciales en los datos. Los modelos se basan en secuencias temporales de acciones y difieren de las reglas de asociación en que las relaciones entre los datos se basan en el tiempo.

5.2.1.1 **La tarea de clasificación**

La tarea de clasificar se presenta en un gran rango de actividades humanas. En este aspecto podemos definir la tarea de clasificación desde diferentes y variados puntos de vista. Usando el término de manera general podríamos indicar que la clasificación cubre cualquier contexto en el cual hay que realizar algún pronóstico o tomar alguna decisión en base a la información que se posee en un momento dado. En este sentido, un procedimiento de clasificación es una

técnica formal que repetidamente juzga nuevas situaciones. En un sentido más restrictivo, se puede enfocar como que el problema de la clasificación se engloba dentro de la construcción de procedimientos a los cuales se les aplica una secuencia continua de ejemplos, donde a cada nuevo ejemplo se le debe de asignar una de las clases ya predefinidas dentro de un conjunto, en base a las observaciones de sus atributos, [71].

Todo algoritmo que trate de resolver la tarea de la clasificación intenta construir un modelo (denominado clasificador) a partir de un conjunto de ejemplos o datos de entrada E, denominado conjunto de entrenamiento (este conjunto debe tener ejemplos de cada una de las clases del problema. Posteriormente, estos clasificadores son usados para inferir la clase de ejemplos desconocidos. Un clasificador se construye a partir de datos con clase conocida, posteriormente es aplicado para predecir valores de clase para ejemplos cuya etiqueta es desconocida. Internamente un clasificador es un algoritmo o expresión matemática que predice un valor discreto para cada entrada. Estos algoritmos trabajan buscando dentro de un espacio de posibles funciones llamadas hipótesis para encontrar una de ellas que sea la mejor aproximación a la función real de inferencia de la clase desconocida.

Generalmente se requieren varias pruebas para encontrar una técnica de clasificación que sea el mejor en las tareas o problemas a solucionar. Esto justifica el hecho de la existencia de diferentes modelos de clasificación, que abordan el problema desde distintas perspectivas y enfoques.

El proceso de clasificación, de manera simplificada consiste en:

1. Cada ejemplo de un conjunto de datos es rotulado con el valor de un atributo especial. Este atributo recibe el nombre de clase.

2. Cada uno de los valores (discretos) que puede tomar este atributo corresponde a una clase diferente.

3. El resto de los atributos de la instancia se utilizan para aprender su clase.

Todos los modelos de clasificación, tienden a describir (en función de su nivel de complejidad) un modelo del sistema que están aprendiendo, [97]. El aprendizaje del modelo es el proceso en el que se obtiene, a partir del conjunto de ejemplos, los parámetros necesarios que forman parte de la descripción del sistema. Se puede distinguir entre parámetros internos y parámetros externos del modelo. Los parámetros internos son aquellos que se estiman mediante un algoritmo de aprendizaje a partir del conjunto de ejemplos. Los parámetros externos son aquellos parámetros que forman parte del modelo pero no son aprendidos. Estos parámetros externos son dados como entrada al algoritmo de aprendizaje y son resultado de algún conocimiento a priori o experimentación previa. En algunos modelos existen estudios ya realizados

que aportan el conjunto de valores que deben tomar los parámetros externos para garantizar la convergencia del proceso, buen modelado, etc.

La base para el algoritmo de aprendizaje es la optimización de una función de coste que varía dependiendo del modelo. La forma general de aprendizaje es expresar este coste como una función de los parámetros deseados para la optimización.

Se distinguen dos fases en el aprendizaje, el entrenamiento y el test del modelo aprendido. En la primera fase se trata de hacer que la técnica de aprendizaje extraiga las conclusiones apropiadas del conjunto de ejemplos de entrenamiento y devuelva un modelo que sea capaz de mostrar lo aprendido. En la segunda fase se estudia la precisión del modelo aprendido probándolo con un conjunto de datos diferente y estimando, en la mayoría de los casos, unos coeficientes de error.

Una vez que está disponible un modelo del sistema, se puede inferir la clase de ciertos ejemplos a partir del resto de valores conocidos de dicho ejemplo dado del sistema.

Existen muchas características que un clasificador debería tener. Algunas de las más importantes a considerar son, [150]:

- *Precisión.* Es la fiabilidad del modelo, normalmente representada por la proporción de ejemplos correctamente clasificados. Aunque cabe recordar que puede haber unos errores más importantes que otros y también que puede ser importante controlar el porcentaje de error para cada clase.

- *Velocidad.* En algunas circunstancias, la velocidad del clasificador es muy importante.

- *Compresibilidad.* Si es un operador humano el que debe aplicar el clasificador, el procedimiento debe ser fácil de entender.

- *Tiempo en aprender.* Especialmente en sistemas que cambian rápidamente puede ser necesario aprender un modelo rápidamente o hacer ajustes de un modelo existente en tiempo real.

Tras describir de forma un poco más extensa las características de la tarea de clasificación de la fase de minería de datos, vamos a describir brevemente y sin ser exhaustivos, los distintos tipos de técnicas que llevan a cabo la minería de datos. Dado que todos ellos han sido ampliamente estudiados, sólo indicaremos algunas referencias bibliográficas representativas de cada una de ellas.

5.3 Técnicas de la minería de datos

Existen diferentes paradigmas detrás de las técnicas usadas en la fase de minería de datos, entre otras, encontramos, [92, 202]:

- técnicas de inferencia estadística
- árboles de decisión
- inducción de reglas
- redes neuronales artificiales
- aprendizaje basado en ejemplos
- aprendizaje bayesiano

A continuación, vamos a revisar brevemente los aspectos principales de algunos de los paradigmas mencionados.

- Inferencia estadística, [119, 201]: Hay muchos conceptos estadísticos que son la base de muchas técnicas de minería de datos, como la regresión lineal, no lineal, funciones discriminantes, etc., que pueden ser paramétricas o no paramétricas.
- Árboles de decisión, [105, 106, 168]: Son una serie de decisiones o condiciones organizadas en forma jerárquica a modo de árbol. Son muy útiles para encontrar estructuras en espacios de alta dimensión y en problemas que mezclen datos nominales y numéricos. Esta técnica se usa en las tareas de clasificación, regresión y agrupamiento. Los árboles de decisión usados para predecir valores nominales se denominan árboles de clasificación. Cuando se utilizan para predecir atributos numéricos se denominan árboles de regresión. Los árboles siguen una aproximación divide y vencerás para partir el espacio del problema en subconjuntos. En el nodo raíz tenemos el problema a resolver. Los nodos internos corresponden a particiones sobre atributos particulares, y los arcos que parten de un nodo corresponden a los posibles valores del atributo considerado en ese nodo. Los nodos hojas representan la predicción del problema para todos aquellos ejemplos que alcanzan esa hoja. Para predecir un atributo de un ejemplo nuevo, se recorre el árbol de arriba a abajo de acuerdo a los atributos en cada nodo, y cuando se llega a una hoja, el valor del atributo se estima de acuerdo a la información contenida en esa hoja. Existen muchos algoritmos para construit árboles de decisión que difieren entre sí en la forma de crear el árbol.

- Inducción de reglas, [2, 201]: Los árboles de decisión pueden considerarse sistemas de inducción de reglas, ya que cada rama del árbol puede interpretarse como una regla, donde los nodos internos en el camino desde el nodo raíz a las hojas definen los términos de conjunción que constituye el antecedente de la regla y la información asignada en la hoja es el consecuente. Sin embargo, aunque los árboles de decisión pueden producir un conjunto de reglas, las técnicas de inducción de reglas son diferentes ya que:
  - las reglas son independientes y no tienen porqué formar un árbol,
  - las reglas generadas pueden no cubrir todas las situaciones posibles,
  - las reglas pueden entrar en conflicto en sus predicciones y en este caso es necesario elegir la regla que se debe seguir. Una técnica para resolver los conflictos es asignar un valor de confianza a las reglas y usar la de mayor confianza.

- Redes neuronales artificiales, [96, 148]: Permiten modelizar problemas complejos en los que puede haber interacciones no lineales entre las variables. Una red neuronal puede verse como un grafo dirigido con muchos nodos (elementos de proceso) y arcos entre ellos (sus interconexiones). Cada uno de los elementos funciona independientemente de los demás usando sus datos locales para dirigir su procesamiento. La organización más popular de una red neuronal consta de una capa de entrada, en la que cada nodo corresponde a un atributo independiente a examinar, unos nodos internos organizados en una o varias capas ocultas y una capa de salida con los nodos de salida (los posibles valores del atributo objetivo). Los nodos de la capa oculta pueden estar conectados a nodos de otra capa oculta o a los nodos de la capa de salida. Cada arco está etiquetado por un peso de conexión y en cada nodo hay una función de activación que indica el efecto de ese nodo sobre los datos que entran en él. Para usar una red neuronal entrenada se introducen los valores de los atributos de un ejemplo en los nodos de entrada y los nodos de salida determinan las predicciones para dicho ejemplo. Los pesos de conexión son parámetros desconocidos que deben estimarse por una técnica de entrenamiento. La técnica más comúnmente utilizado es el de propagación hacia atrás. Las redes neuronales tienen una gran capacidad de generalización para problemas no lineales, aunque requieren bastantes datos para su entrenamiento. Su principal desventaja es que el modelo aprendido es poco comprensible.

- Aprendizaje basado en ejemplos, [71, 201]: Los ejemplos se almacenan en memoria, de tal forma que cuando llega un nuevo ejemplo cuya clase es desconocida, se trata de relacionar éste con los ejemplos almacenados (con clases conocidas) buscando los que más se parecen, con el objetivo de usar los valores de estos ejemplos similares para

estimar la clase del nuevo ejemplo en cuestión. Todo el trabajo en el aprendizaje basado en ejemplos se realiza cuando llega un nuevo ejemplo a clasificar y no cuando se procesa el conjunto de entrenamiento. Cada nuevo ejemplo se compara con los existentes usando una métrica de distancia, y los ejemplos más cercanos se usan para asignar la clase al ejemplo nuevo. La variante más sencilla de esta técnica de clasificación es conocido como "el vecino más cercano" donde la clase del vecino más cercano es asignada al nuevo ejemplo. Otra variante, conocida como la técnica de los "k vecinos más cercanos", usa los k vecinos más cercanos, en cuyo caso la clase mayoritaria de estos k vecinos se asigna al nuevo ejemplo.

- Técnicas bayesianas, [97, 122]: Basados en la idea de asignar a un ejemplo de entrada sin clasificar la clase de mayor probabilidad. Entre estas técnicas se encuentra Naive Bayes, que se basa en la regla de Bayes y que asume independencia entre los atributos dada la clase. Esta técnica funciona bien con conjuntos de datos reales, sobre todo combinado con otros procedimientos de selección de atributos que sirven para eliminar las redundancias.

Dado el objetivo de nuestro trabajo, no hemos pretendido realizar una revisión bibliográfica de técnicas de minería de datos en general, sino que nos centraremos en aquellas técnicas que han puesto cierta atención al tratamiento de LQD en el conjunto de datos de entrada en cualquiera de sus formas. Este es el objetivo que seguimos en la siguiente sección.

5.4 Minería de datos de baja calidad

Como hemos visto, hay un amplio rango de técnicas de minería de datos basadas en diferentes propuestas teóricas. Desafortunadamente, la mayoría de las técnicas convencionales no suelen considerar fuentes de imperfección. Como resultado los datos incompletos e imperfectos son normalmente descartados o ignorados en el aprendizaje de las distintas técnicas y en los subsequentes procesos de inferencia o predicción. Sin embargo, este tipo de datos aparece inevitablemente cuando tratamos con aplicaciones del mundo real, tal y como hemos comentado. Se han realizado grandes esfuerzos para incorporar el tratamiento de la imperfección en las fases de aprendizaje e inferencia de las técnicas de minería de datos. Por ejemplo, algunas técnicas que permiten el tratamiento de ejemplos con valores missing en los atributos, son las técnicas de aprendizaje basado en ejemplos, [201], técnicas de construcción de árboles de decisión para clasificación/regresión, [70, 169], clasificadores basados en funciones discriminantes formados por normales multivariantes, [119], técnicas de clustering basados en modelos de mezclas de gaussianas factorizadas, [174], redes bayesianas, [97], etc.. En [148] se propone el modelado

de un sistema experto basándose en una versión fuzzy del perceptrón multicapa. El sistema que se propone infiere el valor de pertenencia a la clase de salida de un ejemplo de entrada y también genera un cierto grado de seguridad que expresa la confianza en la decisión. El modelo es capaz de consultar al usuario la información de los atributos de entrada más importantes, siempre y cuando sea necesario, en el caso de tener datos de entrada parciales. Este sistema que utiliza una versión el perceptrón multicapa fuzzy también permite en los datos de entrada valores missing. Cuando el sistema se encuentra un valor missing, este valor pasa a tres neuronas, asignando un peso de valor 0.5 que representa el valor más ambigüo de pertenencia fuzzy. Otras técnicas que también permiten el tratamiento de valores missing se presentan en [105] y [106], donde se construye un árbol de decisión fuzzy el cual puede manejar atributos nominales y numéricos, pero estos últimos deben de estar discretizados mediante una partición fuzzy. Tanto en los atributos nominales como en los atributos numéricos el árbol de decisión fuzzy permite la existencia de valores missing, así cuando un nodo N del árbol de decisión fuzzy va a ser expandido y se encuentra con un ejemplo con un valor missing en el atributo test, el ejemplo desciende por todos los nodos hijos de N, siendo el peso asignado a ese ejemplo en cada nodo hijo, proporcional al número de hijos totales en los cuales se expande el nodo N. Un tratamiento similar se produce en el árbol de decisión C4.5, [169], donde cuando el árbol de decisión se encuentra un ejemplo con un valor missing en el atributo a por el que se expande un nodo N, el ejemplo desciende por todos los nodos hijos de N, con un peso diferente a cada nodo hijo, ya que el peso se calcula de forma proporcional al número de ejemplos con valor conocido en dicho atributo a.

En [81] y [174], además de tratar con valores missing se permite el tratamiento de otro tipo de información imperfecta. Más específicamente, en [174] se describe una versión extendida del algoritmo "Expectation Maximization" (EM) para estimar los parámetros de los modelos de mezcla a partir de ejemplos de entrenamiento inciertos. El modelo generado es una mezcla de normales factorizadas generalizadas y permite el tratamiento de valores missing e incertidumbre modelada mediante distribuciones de probabilidad. En [81] se extiende la técnica anterior para que pueda tratar con información de entrada con valores imperfectos expresados en el marco de una teoría más general a la Teoría de la Probabilidad y que es la Teoría de Evidencias de Dempster-Shafer. De esta forma se permite el tratamiento de valores de entrada expresados mediante valores fuzzy, intervalos, distribuciones de probabilidad, missing, etc.

Otros trabajos que también presentan técnicas de minería de datos que trabajan de forma directa con datos de baja calidad se presentan en [160], [161], [163], [177], [179] y [195]. Veamos de forma breve cómo manejan la información de baja calidad.

En [179] se da una descripción del uso de ciertos tipos de datos mal definidos en los sistemas genéticos fuzzy. Después se propone un modelo de variables fuzzy aleatorias que se

utilizan para justificar la representación de los datos vagos o de baja calidad en términos de conjuntos fuzzy. Además, se adapta la definición de la varianza de una variable aleatoria fuzzy de acuerdo al modelo presentado, por lo que se puede acotar el error cuadrático medio de una base de reglas en un conjunto de datos fuzzy. También definen los autores en este trabajo, los diferentes operadores de predecencia entre los valores de una función de fitness. Todos los elementos son combinados en una extensión del algoritmo NSGA-II para optimizar una combinación de objetivos crisp e intervalares capaces de aprender reglas fuzzy.

En [160] se propone la extensión de un sistema genético fuzzy, al cual denominan algoritmo genético de aprendizaje cooperativo-competitivo (GCCL), para que pueda tratar con LQD en problemas de clasificación. El sistema genético fuzzy obtiene un sistema de reglas fuzzy a partir de los datos, donde el rol de los conjuntos fuzzy en este sistema de reglas es modelar los LQD utilizando las herramientas que proporciona la lógica fuzzy. El sistema genético fuzzy que proponen los autores, junto con las definiciones de las diferentes funciones que adaptan del mismo, es evaluado con datos reales que contienen de forma explícita LQD, obteniendo un resultado bastante satisfactorio. Además, en [161] los mismos autores evalúan el sistema propuesto con varios conjuntos de datos de casos reales sobre el diagnóstico de la dislexia. Diversas evaluaciones con otros datos reales y ampliaciones del sistema presentado son estudiados más a fondo en [159] y [162]. Además, relacionado con los sistemas de reglas fuzzy, en [177] se propone un nuevo enfoque para obtener clasificadores comprensibles lingüísticamente a partir de datos intervalares. Para ello se define un caso particular de un sistema basado en un conjunto de reglas aleatorias fuzzy y la asignación óptima de los pesos a las mismas. Hay que mencionar que bajo ciertas circunstancias un sistema basado en un conjunto de reglas aleatorias fuzzy puede ser equivalente numéricamente a los sistemas basados en reglas fuzzy. Tras la definición del sistema basado en un conjunto de reglas aleatorias fuzzy, se usa un algoritmo descendente con un esquema co-evolutivo para buscar en paralelo el mejor conjunto de reglas y seleccionar los dos conjuntos de entrenamiento donde se alcanza la probabilidad más baja y más alta. Estos dos límites son utilizados para encontrar el modelo que se encuentra no dominado por el resto de modelos.

En [195] se propone el aprendizaje de modelos basados en ecuaciones para tratar con LQD. Los modelos basados en ecuaciones incluyen una representación de la incertidumbre y la evaluación de un modelo genera un valor fuzzy. El aprendizaje de este tipo de modelos lo definen los autores como un problema multi-objetivo utilizando funciones de fitness fuzzy y además proponen dos estrategias evolutivas de aprendizaje. En este caso para evaluar las propuestas se han utilizado problemas sintéticos. Los resultados indican que la propuesta que realizan obtiene un rendimiento similiar cuando trabaja con LQD y cuando trabaja sin este tipo de datos, sin embargo, el algoritmo multi-objetivo NSGA-II, [61], muestra una mayor dispersión en el

espacio del fitness. Además, los autores concluyen que la representación de la incertidumbre en los modelos de aprendizaje basados en ecuaciones parecen ser válidos para el modelado de sensores reales de baja calidad.

Otro trabajo que utiliza los datos de baja calidad, esta vez mediante un conjunto de clasificadores se propone en [163]. En este trabajo se presenta una extensión del algoritmo Adaboost para obtener clasificadores basados en reglas fuzzy a partir de LQD. Las reglas fuzzy aisladas son consideradas con aprendices débiles y las bases de conocimiento como ensambles. Las reglas son iterativamente añadidas a una base y la búsqueda de la mejor regla en cada iteración se lleva a cabo con un algoritmo genético dirigido por una función de fitness fuzzy. Los sucesivos pesos de las instancias son también fuzzy, sin embargo a cada regla se le asigna un número crisp de votos que se interpretan como el grado de importancia de esa regla.

Como hemos apreciado, aunque no son muchos, cada vez podemos encontrar nuevas técnicas de minería de datos que pueden trabajar directamente con LQD. En todos los trabajos presentados en esta sección se enfatiza la importancia de adaptar o diseñar nuevas técnicas de forma que puedan trabajar de forma directa con LQD, puesto que cada vez más estos tipos de datos aparecen en problemas de aplicación del mundo real.

Parte II

MINERÍA DE DATOS DE BAJA CALIDAD

En esta PARTE II vamos a presentar nuestra propuesta de tres técnicas de minería de datos capaces de tratar con datos de baja calidad. Como se ha comentado en la PARTE I se han desarrollado técnicas muy competitivas para llevar a cabo la fase de minería de datos del AID, sin embargo, la gran mayoría de ellas no son capaces de trabajar con datos de baja calidad que aparecen con frecuencia en las aplicaciones del mundo real. Por este motivo nuestro objetivo en esta PARTE es extender y/o diseñar técnicas de minería de datos que tengan la capacidad de poder tratar LQD. En concreto, las técnicas que proponemos son extensiones de un árbol de decisión fuzzy, del ensamble Fuzzy Random Forest y de la regla de vecinos más cercanos para clasificar. Para cada una de estas técnicas, en primer lugar presentaremos los algoritmos básicos de la técnica en la que se basan, para posteriormente presentar la extensión de dichos algoritmos para incorporarles el tratamiento de LQD. En segundo lugar, llevaremos a cabo la validación de las dos primeras técnicas mediante la realización de una serie de experimentos en el Caítulo 7. La tercera técnica será validada más adelante en el Capítulo 12 en su aplicación como técnica para llevar a cabo el preprocesamiento de datos.

Un aspecto que debemos aclarar en este punto es el motivo por el que hemos invertido el orden natural de presentación de las dos fases del AID en las que se centra este trabajo: la fase de preprocesamiento de datos y la fase de minería de datos. Si bien en el AID la fase de preprocesamiento de datos es previa a la de minería de datos, presentamos primero nuestro trabajo dentro de la fase de minería de datos por dos razones. La primera de ellas es que en el diseño de las técnicas de preprocesamiento de datos que proponemos forman parte ciertos elementos de las técnicas de minería de datos propuestas. La segunda razón es que para llevar a cabo la validación experimental de las técnicas de preprocesamiento propuestas necesitamos técnicas de minería de datos que soporten el tratamiento de LQD. Usaremos las técnicas presentadas en esta PARTE II para llevar a cabo la experimentación en la PARTE III.

El esquema general del desarrollo de esta parte, se muestra en la Figura 5.1.

Figura 5.1: Esquema general de la parte

CAPÍTULO

6

Una extensión de un Árbol de Decisión Fuzzy

6.1 Introducción

Los árboles de decisión son una de las elecciónes más populares para el aprendizaje y el razonamiento basado en las características de los ejemplos, [17, 106, 154, 157, 168, 169, 189]. La popularidad de esta técnica se basa en la gran facilidad para comprender y analizar los resultados, su eficiencia, la independencia del problema y la capacidad para tratar con aplicaciones a gran escala. Los árboles de decisión se engloban dentro del aprendizaje supervisado, en donde los conjuntos de ejemplos están representados mediante un conjunto de atributos o características. Algunos de estos ejemplos ya tienen asignada una clase, es decir, ya están clasificados y son ejemplos que se usan en la fase de aprendizaje del árbol. El objetivo es conseguir una técnica con la cual discriminar, describir o taxonomizar la tendencia de aquellos ejemplos que no tienen una clase asignada. Dos de las técnicas más importantes son ID3 y CART, [168, 169].

No todo son ventajas en los árboles de decisión ya que se conocen como clasificadores inestables con respecto a las perturbaciones en los datos de entrenamiento, en otras palabras, los árboles de decisión pueden presentar una alta varianza. Sin embargo esta desventaja se ve paliada en parte al hacer uso de la lógica fuzzy en lugar de la lógica clásica, lógica que utilizan las técnicas ID3 y CART. Esto sucede porque la lógica fuzzy aporta una mejora en estos aspectos debido a la elasticidad en el formalismo de los conjuntos fuzzy, [106, 154, 157, 189].

Utilizaremos el árbol de decisión fuzzy propuesto en [106] como procedimiento base para la construción de un árbol de decisión fuzzy extendido para la manipulación de distintos LQD.

En la siguiente sección presentamos los conceptos básicos del árbol fuzzy propuesto en [106] y a continuación presentaremos la extensión propuesta.

6.2 Un árbol de decisión fuzzy

La principal diferencia entre un árbol de decisión crisp y un árbol de decisión fuzzy, además de que trabaja con lógicas diferentes, es que en un árbol de decisión fuzzy existe la posibilidad de que un ejemplo descienda por más de una rama del árbol y alcance más de una hoja. Esta posibilidad radica en el hecho de que la lógica fuzzy permite manejar un grado de imprecisión asociado a cada ejemplo. Sin embargo, en un árbol de decisión crisp, el ejemplo solamente desciende por una rama y alcanza una única hoja.

El árbol de decisión fuzzy ("fuzzy decision tree"), que denotaremos por FDT, propuesto en [106] está basado en el árbol fuzzy ID3, [189], donde todos los atributos numéricos tienen que ser discretizados mediante un conjunto de particiones fuzzy. Antes de entrar en detalle en el funcionamiento tanto de la fase de aprendizaje como de inferencia del árbol de decisión fuzzy, vamos a introducir la nomenclatura que utilizaremos.

6.2.1 Nomenclatura

- N: Nodo que está siendo explorando en un momento dado.
- C: Conjunto de clases o posibles valores del atributo de decisión. $|C|$ denota la cardinalidad de C.
- E: Conjunto de ejemplos. $|E|$ indica el número de ejemplos, es decir, la cardinalidad de E.
- E_N: Conjunto de ejemplos en el nodo N; $E_N \subseteq E$.
- e_j: j-ésimo ejemplo del conjunto de datos.
- A: Conjunto de atributos que describen un ejemplo. $|A|$ denota el número de atributos que describen un ejemplo.
- G_i^N: La ganancia de información cuando el nodo N is dividido por el atributo i.

$$G_i^N = I^N - I^{S_{V_i}^N} \tag{6.1}$$

donde:

  - I^N: La información estándar asociada con el nodo N. Esta información se calcula como sigue:

1. Para cada clase $k = 1, ..., |C|$, el valor de P_k^N, el cual es el número de ejemplos en el nodo N que pertenecen a la clase k, se calcula como:

$$P_k^N = \sum_{j=1}^{|E|} \chi_N(e_j) \cdot \mu_k(e_j)$$

donde:

· $\chi_N(e_j)$ es el grado de pertenencia del ejemplo e_j al nodo N.

· $\mu_k(e_j)$ es el grado de pertenencia del ejemplo e_j a la clase k.

2. P^N, que es el número total de ejemplos en el nodo$N$, es calculado como:

$$P^N = \sum_{k=1}^{|C|} P_k^N$$

3. La información estándar se calcula como:

$$I^N = -\sum_{k=1}^{|C|} \frac{P_k^N}{P^N} \cdot \log \frac{P_k^N}{P^N}$$

– $I^{S_{V_i}^N}$ es el producto de tres factores y representa la información estándar obtenida al dividir el nodo N usando el atributo i ajustado a la existencia de valores missing en este atributo.

$$I^{S_{V_i}^N} = I_1^{S_{V_i}^N} \cdot I_2^{S_{V_i}^N} \cdot I_3^{S_{V_i}^N}$$

donde:

∗ $I_1^{S_{V_i}^N}$ = 1 - $\frac{P^{N_{m_i}}}{P^N}$, donde $P^{N_{m_i}}$ es el peso de los ejemplos en el nodo N con valores missing en el atributo i.

∗ $I_2^{S_{V_i}^N}$ = $\frac{1}{\sum_{h=1}^{H_i} P^{N_h}}$, siendo H_i el número de descendientes asociado con el nodo N cuando este nodo es dividido por el atributo i and P^{N_h} el peso de los ejemplos asociados con cada uno de los descendientes.

∗ $I_3^{S_{V_i}^N}$ = $\sum_{h=1}^{H_i} P^{N_h} \cdot I^{N_h}$, siendo I^{N_h} la información estándar de cada descendiente h del nodo N.

6.2.2 **Aprendizaje del árbol de decisión fuzzy**

El FDT se construye mediante un algoritmo recursivo en el cual se van explorando las diferentes ramas del mismo en base a los mejores atributos del conjunto de ejemplos de entrenamiento, descritos por atributos donde uno de ellos es nominal y actúa de atributo clase. Los atributos que describen un ejemplo pueden tomar valores numéricos, nominales, missing y etiquetas lingüísticas. Los atributos numéricos necesitan de una partición para que el árbol de decisión

pueda trabajar con este tipo de datos. Estas particiones se deben de proporcionar como información de entrada al FDT. Para ello se dispone de un particionamiento fuzzy de los atributos, donde para cada atributo se indican los distintos valores que se puedan presentar. Este particionamiento muestra para los atributos numéricos los correspondientes conjuntos fuzzy asociados a cada etiqueta lingüística de dicho atributo. Las particiones para todos los atributos numéricos tienen las siguientes características:

- Todos los puntos del dominio son discretizados y pertenecen a una partición (completitud), y
- Las particiones son fuertes, es decir, se verifica que $\forall x \in \Omega_i$, $\sum_{f=1}^{F_i} \mu_{B_f}(x) = 1$, donde $B_1, \ldots, B_{F_i}$ son los F_i conjuntos fuzzy para la partición correspondiente al i-ésimo atributo continuo con dominio Ω_i.

Además el dominio de cada atributo numérico i se particiona en conjuntos fuzzy trapezoidales, B_1, B_2, ..., B_{F_i}, donde:

$$\mu_{B_1}(x) = \begin{cases} 1 & b_{11} \leq x \leq b_{12} \\ \frac{(b_{13}-x)}{(b_{13}-b_{12})} & b_{12} \leq x \leq b_{13} \\ 0 & b_{13} \leq x \end{cases} \quad ; \quad \mu_{B_2}(x) = \begin{cases} 0 & x \leq b_{12} \\ \frac{(x-b_{12})}{(b_{13}-b_{12})} & b_{12} \leq x \leq b_{13} \\ 1 & b_{13} \leq x \leq b_{23} \\ \frac{(b_{24}-x)}{(b_{24}-b_{23})} & b_{23} \leq x \leq b_{24} \\ 0 & b_{24} \leq x \end{cases} ;$$

$$\ldots\ldots\ldots ; \quad \mu_{B_{F_i}}(x) = \begin{cases} 0 & x \leq b_{(F_i-1)3} \\ \frac{(x-b_{(F_i-1)3})}{(b_{(F_i-1)4}-b_{(F_i-1)3})} & b_{(F_i-1)3} \leq x \leq b_{(F_i-1)4} \\ 1 & b_{F_i 3} \leq x \end{cases}$$

Un aspecto a tener en cuenta es que todos los atributos numéricos están normalizados, es decir, $\forall i, \quad \Omega_i = [0, 1]$. Por lo tanto, $b_{11} = 0$ y $b_{F_i 3} = 1$. Como se puede observar en las particiones, los dos primeros puntos de cada conjunto fuzzy (desde el 2 hasta el F_i son los dos últimos puntos del conjunto fuzzy anterior).

Una vez descritas las particiones que se utilizan en este árbol de decisión, en el Algoritmo 1, se detalla el proceso de construcción.

Si analizamos en detalle algunos aspectos del Algoritmo 1, es necesario comentar que en el paso 2 a cada e_j se le asigna un peso igual a 1 ($\chi_{root}(e_j) = 1$) en el nodo raíz, indicando que el ejemplo inicialmente solamente se encuentra en el nodo raíz del FDT. Este valor podrá continuar siendo 1 siempre y cuando el ejemplo e_j no pertenezca a más de un nodo durante el proceso de construcción del árbol. En un árbol de decisión clásico, un ejemplo solamente puede pertenecer a un nodo en cada momento, por lo que este peso inicial no es necesario, ya

Algoritmo 1 Aprendizaje del Árbol de decisión Fuzzy

AprendizajeFDT(in: E, $Particiones\text{-}Fuzzy$; **out:** $Arbol\text{-}Decision\text{-}Fuzzy$)
begin

1. Inicializar:
   1.1 $M_N = A$ con los atributos numéricos discretizados de acuerdo a las $Particiones\text{-}Fuzzy$
   1.2 $E_N = E$
2. **for** cada ejemplo e_j **do**
   $\chi_{root}(e_j) = 1$ con $j = 1, \ldots, |E|$
   end for
3. Seleccionar un atributo de M_N como test en el nodo N:
   3.1 **for** cada atributo i **do**
   calcular la ganancia de información $G_i^N, i = 1, \ldots, |M_N|$
   end for
   3.2 Se elige el atributo i_{best} cuya G_i^N es máxima.
   3.3 Dividir N en H_i nodos hijos de acuerdo al atributo i_{best}
4. **for** cada nodo hijo N_h con $h = 1, \ldots, H_i$: **do**
   4.1 $M_{N_h} = M_N - i_{best}$
   4.2 Obtener E_{N_h} de E_N
   4.3 Ir al paso 3 si no se cumple la condición de parada en el nodo N_h haciendo $M_N = M_{N_h}$ y $E_N = E_{N_h}$

   end for

end

que no se modifica a lo largo de la construcción. Por el contrario para el FDT que estamos describiendo este valor puede modificarse con los valores missing y con los valores numéricos.

- **Missing**: Cuando el ejemplo e_j tiene un valor missing en el atributo i el cual es usado como atributo test en el nodo N. En este caso, el ejemplo desciende por cada nodo hijo $N_h, h = 1, \ldots, H_i$ con el valor modificado proporcionalmente al peso de cada nodo hijo. El valor modificado para cada N_h se calcula como:

$$\chi_{N_h}(e_j) = \chi_N(e_j) \cdot \frac{P^{N_{h,known}}}{P^{N_{known}}} \tag{6.2}$$

  donde $P^{N_{known}}$ es la suma de los pesos de los ejemplos que tienen un valor conocido en el atributo i en el nodo N y $P^{N_{h,known}}$ es la suma de los pesos de los ejemplos con valor conocido i que descienden al nodo hijo N_h.

- **Numérico**: Según el grado de pertenencia de e_j a los diferentes conjuntos de la partición cuando el test del nodo N está basado en el atributo i el cual es numérico. En este caso,

el ejemplo desciende a los nodos hijos a los cuales el ejemplo pertenece con grado mayor que cero ($\mu_{B_f}(e_j) > 0;\ f = 1, \ldots, F_i$).

Debido a las características de las particiones que se utilizan, un ejemplo puede descender como máximo a dos nodos hijos. En esta situación,

$$\chi_{N_h}(e_j) = \chi_N(e_j) \cdot \mu_{B_h}(e_j) \tag{6.3}$$

donde $\mu_{B_h}(e_j) > 0, \forall h$.

- **Etiqueta y nominal**: Cuando el ejemplo e_j tiene un valor nominal o un valor que coincide con una etiqueta de la partición en el atributo i usado como test en el nodo N, el ejemplo e_j desciende al nodo que se corresponda con el valor nominal o con la partición a la cual corresponda la etiqueta. En este caso el peso del ejemplo no varía, ya que el ejemplo pertenece a un único hijo descendiente del nodo N.

En el paso 3 del Algoritmo 1, se calcula la ganancia de información, G_i^N, para cada atributo i con el fin de seleccionar el mejor atributo para dividir un nodo N. Es importante aclarar que en el paso 3.3, si el atributo i_{best}, es un atributo numérico, entonces $H_i = F_i$ (número de conjuntos fuzzy de la partición del atributo i_{best}), si es nominal el valor de H_i es el número de posibles valores del atributo i_{best}.

La condición de parada indicada en el paso 4.3, hace referencia a la primera condición alcanzada en el nodo N de entre las siguientes:

(a) El nodo N es puro, es decir, todos los ejemplos son de la misma clase.

(b) M_N es un conjunto vacío.

(c) E_N alcanza el mínimo número de ejemplos permitidos en un nodo.

Una vez que todos los nodos construidos en el árbol alcanzan una de las condiciones de parada, el árbol de decisión fuzzy queda construido.

6.3 Clasificación en el árbol de decisión fuzzy

Cuando queramos clasificar un nuevo ejemplo usando el FDT utilizaremos el algoritmo de clasificación del árbol que se detalla en el Algoritmo 2.

Algunos detalles a tener en cuenta respecto al Algoritmo 2 son los siguientes. En el paso 2 cuando el ejemplo ec desciende a través del árbol de decisión desde el nodo raíz, hay que tener en cuenta que este ejemplo puede ramificarse y alcanzar más de una rama del árbol de decisión. Por un lado, si el atributo test de algún nodo es un atributo numérico y el valor de ec para ese atributo pertenece a más de un elemento de la partición con grado mayor que cero. Por

Algoritmo 2 Clasificación del Árbol de decisión Fuzzy

ClasificacionFDT(in: $Arbol\text{-}Decision\text{-}Fuzzy$, $Particiones\text{-}Fuzzy$, Ejemplo ec; **out:** Clase c)

begin

1. Inicializar $\chi_{root}(ec) = 1$
2. El ejemplo ec desciende a través del árbol desde N_{root}
3. L= el conjunto de hojas alcanzadas por ec
4. Se calcula el soporte para cada una de las clases k, $P_k = \sum_{l=1}^{|L|} \chi_l(ec) \cdot P_k^{N_l}$, $k = 1, \ldots, |C|$
5. Se devuelve la clase c obtenida mediante algún critério de decisión determinado.

end

otro lado, el ejemplo también puede ramificarse por más de una rama si al menos hay un valor missing en alguno de los atributos usados por el árbol de decisión como test. Debido a que un ejemplo puede descender por más de una rama del árbol, en los pasos 3 y 4 del Algoritmo 2, se calcula el soporte o peso que cada hoja alcanzada por ejemplo ec tiene para cada clase, considerando así la información de todas las hojas alcanzadas.

El paso 5 del Algoritmo 2 puede ser modificado dependiendo de la forma en la cual queramos inferir, por ejemplo, la clase mayoritaria considerando todas las hojas alcanzadas por ec en el árbol de decisión o la clase mayoritaria de la hoja alcanzada con mayor grado por el ejemplo ec. En este último caso no es necesario calcular el peso para todas las hojas alcanzadas por el ejemplo ec sino que simplemente elegimos la hoja alcanzada con mayor peso y la clase mayoritaria de esa hoja será la decisión tomada por el árbol. Otra opción posible es ponderar la clase mayoritaria de cada hoja alcanzada, utilizando el peso con el cual el ejemplo alcanza la hoja. Una vez ponderadas todas las clases mayoritarias, elegimos como decisión final del árbol de decisión la clase con mayor valor.

6.4 Mostrando el proceso del árbol de decisión fuzzy

Tras detallar la fases de aprendizaje y clasificación del FDT, veamos un ejemplo para clarificar algunos de los pasos del proceso. Para ello, vamos a suponer un conjunto de datos compuesto de 72 ejemplos divididos en tres clases, c1, c2 y c3. Cada ejemplo se describe por tres atributos numéricos (At1, At2 y At3) y un atributo nominal que actúa como valor de clase. Para cada atributo númerico Ati, $i = 1, \ldots, 3$, disponemos de una partición fuzzy de su dominio.

Durante el proceso de aprendizaje (construcción) del FDT se tiene lo siguiente: Para cada nodo, el número de ejemplos obtenido P_k^N para cada clase k teniendo en cuenta si el valor de los atributos son missing o númericos (ecuaciones 6.2 y 6.3); y además, para cada nodo, se van calculando los valores de G_i^N para cada atributo Ati y se elige para dividir el nodo el atributo con mayor G_i^N.

Este proceso se repite para cada nodo construido. Una vez terminado el proceso de construcción, el árbol obtenido está compuesto de once nodos, donde seis de ellos son nodos hojas (Figura 6.1).

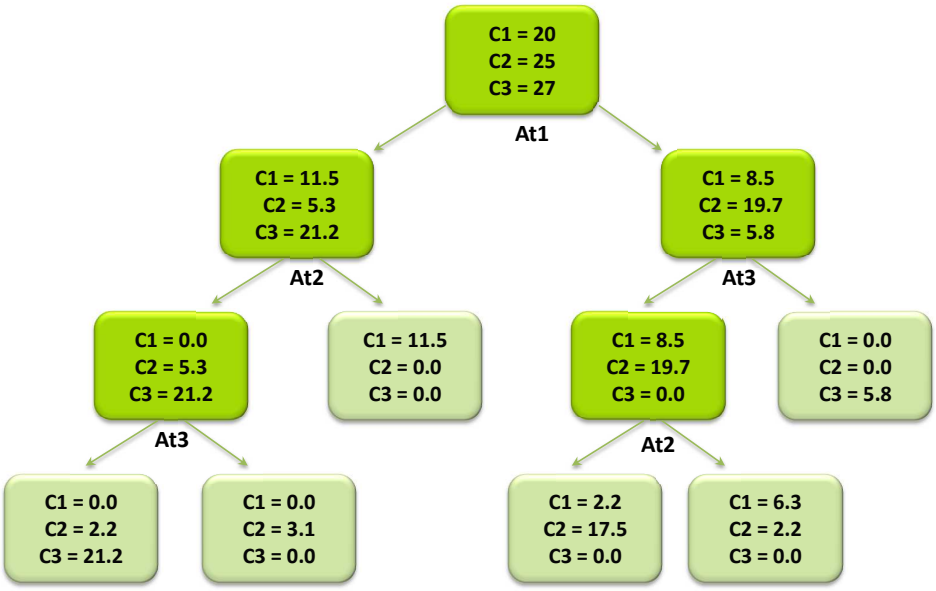


Figura 6.1: FDT aprendido (construido)

Los nodos hojas se forman por un lado porque son puros solamente contienen una clase y por otro lado porque los nodos no tienen más atributos para seguir dividiendo. En cada nodo se indica el atributo por el cual se divide el nodo (At1, At2 o At3) y para cada clase se muestra los pesos de los ejemplos que se encuentran en ese nodo. Hay que tener presente que al tratarse de un FDT un ejemplo descenderá por más de un nodo y por lo tanto no podemos hablar de número de ejemplos en los nodos sino del peso de los mismos.

Una vez aprendido el FDT, vamos a clasificar un ejemplo ec y durante este proceso mostraremos como el ejemplo activa los nodos del árbol.

Para ello, vamos a suponer que los atributos At1 y At3 son númericos (particionados en sus correspondientes particiones fuzzy) y el atributo At2 es nominal. El ejemplo $ec = (?, a, 1{,}7)$

Este ejemplo ec comenzará en el nodo raíz. Dicho nodo se divide mediante el atributo At1, y como el ejemplo ec tiene un valor de missing, se propaga el ejemplo por las dos ramas del árbol. En este caso, suponemos que de los 72 ejemplos en el nodo, 70 tienen valores conocidos en este atributo y 42 se propagarían por la rama izquierda y 28 por la derecha, proporcionando los valores de 0.6 y 0.4 de propagación del ejemplo ec por las dos ramas. Por la rama izquierda, el nodo siguiente se divide mediante el atributo At2, y como el ejemplo ec tiene un valor de $\{a\}$, se propaga íntegramente (0.6) por la rama izquierda del nodo. Si continuamos por este nuevo nodo activado, este se divide mediante el atributo At3. El valor de este atributo en el

ejemplo es 1.7 y activa la partición del atributo en con dos valores de pertenencia 0.4 y 0.6 (Figura 6.2).

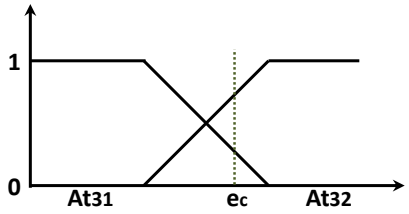


Figura 6.2: Activación del nodo con respecto al ejemplo ec

Calculando el valor de activación del ejemplo ec a cada una de las ramas, obtenemos los valores 0.24 y 0.36. Así, sucesivamente irá descendiendo por las ramas del árbol. En la Figura 6.3 mostramos en rojo las ramas por las cuales a descendido el ejemplo y también en rojo se encuentran las hojas que finalmente ha alcanzando indicando para cada una de ellas su peso.

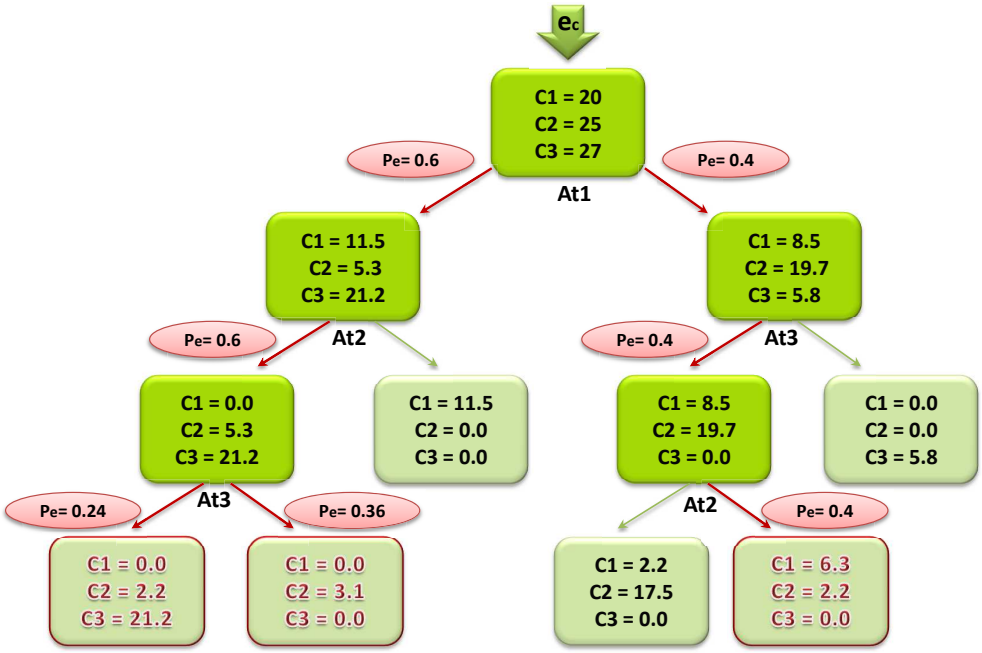


Figura 6.3: Clasificación de un ejemplo en el FDT

Como podemos apreciar, en la Figura 6.3 el ejemplo ec alcanza cuatro hojas con diferentes pesos.

Vamos a utilizar dos métodos diferentes para la toma de decisión final. Primero vamos a tomar la clase mayoritaria de la hoja que se ha alcanzado con mayor peso y después vamos a ponderar las clases con el peso con el cual el ejemplo alcanza cada hoja y combinando la información para decidir finalmente la clase con mayor valor obtenido.

Analizando las hojas alcanzadas y tomando como método de decisión la elección de la clase mayoritaria en la hoja alcanzada con mayor peso, el árbol decide que la clase del ejemplo ec es c1, ya que la hoja que se alcanza con mayor peso (0.4) tiene clase c1 es la clase mayoritaria de esa hoja.

Sin embargo si tomamos como método de decisión la clase mayoritaria ponderada por los pesos con los cuales se alcanza cada hoja, el árbol toma como decisión que el ejemplo ec pertenece a la clase c3 debido a lo siguiente: Si ponderamos la clase c1 de cada hoja alcanzada tenemos que el peso de la clase es $0.0\times0.24+0.0\times0.36+6.3\times0.4=2.52$; para la clase c2 obtenemos un peso de $2.2\times0.24+3.1\times0.36+2.2\times0.4=2.524$; y para la clase c3 de $21.2\times0.24+0.0\times0.36+0.0\times0.4=5.088$. Por lo tanto analizando los resultados vemos que la clase mayoritaria es la clase c3.

6.5 Una extensión del árbol de decisión fuzzy

En esta sección vamos a presentar una extesión del FDT para que pueda aprender y clasificar a partir de nuevos tipos de LQD. Para ello, primero vamos a describir estos nuevos tipos de datos con los que trabajaremos de forma explícita, y después describiremos los nuevos elementos y cambios necesarios para esta extensión.

6.5.1 Nuevos tipos de datos de baja calidad

En la Subsección 6.2.2 hemos comentado los tipos de datos soportados por el FDT, tanto en la fase de aprendizaje como en la de clasificación, tal y como originalmente fue propuesto en [106]. En resumen, soporta atributos numéricos, nominales, missing y etiquetas lingüisticas. Los atributos numéricos deben estar particionados mediante una partición fuzzy y las etiquetas deben ser algunas de las definidas en las correspondientes particiones. Ahora, queremos extender el aprendizaje y clasificación del FDT para incorporar nuevos tipos de datos (denotaremos a esta extensión como FDT_{LQD}). Estos nuevos tipos de datos serán valores intervalares, valores fuzzy (o etiquetas) que puedan ser diferentes de los valores fuzzy que constituyen la partición fuzzy del atributo y, por lo tanto, el grado de similitud de estos valores fuzzy a cada elemento de la partición fuzzy del atributo puede ser menor que 1, y datos expresados mediante subconjuntos crisp/fuzzy. Además queremos permitir que el atributo clase pueda estar expresado mediante valores imprecisos a través de un subconjunto crisp/fuzzy de valores del dominio. En la Figura 6.4 se muestran estos tipos de valores.

Para incorporar estos nuevos tipos de datos, necesitamos modificar los siguientes elementos:

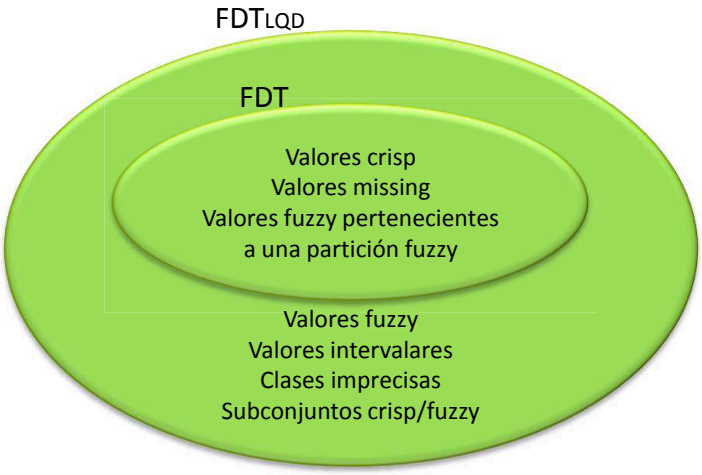


Figura 6.4: Extendiendo la información procesada por el árbol de decisión fuzzy

- Definir una función extendida para medir el grado de pertenencia de estos nuevos tipos de datos a las particiones fuzzy de los atributos numéricos utilizados por el FDT.
- La definición de esta función requiere la modificación de la función de *ganancia de información* tanto en la fase de aprendizaje como en la fase de clasificación.
- Para incorporar las clases imprecisas, debemos de modificar también las fases de aprendizaje y clasificación.

Vamos a analizar más en detalle estos nuevos elementos y modificaciones.

6.5.1.1 Valores fuzzy e intervalares

Para incorporar atributos con valores fuzzy (conjuntos fuzzy) diferentes de los conjuntos de las particiones del atributo, o atributos con valores intervalares, debemos de extender la función que mide el grado de pertenencia de estos tipos de valores a los diferentes conjuntos fuzzy que forman la partición del atributo numérico.

Hasta ahora en el árbol de decisión fuzzy, cuando un atributo numérico está discretizado mediante una partición fuzzy, los únicos valores fuzzy permitidos en los datos de entrada para este atributo son los valores de esa partición. Por ejemplo, si un atributo X está discretizado de acuerdo a la partición mostrada en la Figura 6.5.a), los únicos valores permitidos en los datos de entrada son B_1, B_2 ó B_3. En la extensión del FDT, los valores fuzzy pueden ser diferentes de B_1, B_2 y B_3, por lo que se permiten datos de entrada como los mostrados en la Figura 6.5.b). El grado de similaridad de los valores fuzzy con cada uno de los valores B_1, B_2 y B_3 serán utilizados para modificar el valor χ_{node} del ejemplo tanto en la fase de aprendizaje como en la de inferencia del FDT.

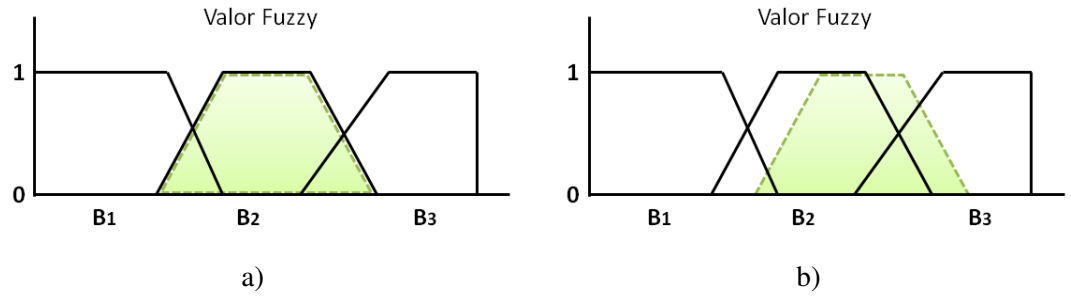


Figura 6.5: Manejo de los valores fuzzy en el FDT a) y en el ensamble FDT_{LQD} b)

Esta nueva función (denotada por $\mu_{simil}(\cdot)$) captura los cambios en el valor $\chi_N(e)$ del ejemplo e cuando éste desciende en el árbol de decisión fuzzy y tiene algún atributo cuyos valores están expresados mediante intervalos o valores fuzzy que no coinciden con los valores fuzzy de la partición. Esta función será utilizada tanto en la fase de aprendizaje como de clasificación.

En estas situaciones, el valor de $\chi(\cdot)$ se calculará como $\chi_{childnode}(e) = \chi_N \times \mu_{simil}(e)$.

Usando la función $\mu_{simil}(\cdot)$, permitimos un nuevo tratamiento de los valores missing. En la Subsección 6.2.2, indicamos que un valor missing desciende a través de un árbol distribuyendo su peso proporcionalmente entre los hijos de un nodo. Mediante la incorporación de los valores intervalares, podemos expresar un valor missing como un intervalo que incluye todo el dominio de un atributo. Así utilizando la función $\mu_{simil}(\cdot)$, el ejemplo desciende a todos los hijos de un nodo distribuyendo su peso proporcionalmente en ellos, de acuerdo al tamaño de cada conjunto fuzzy de la partición. En un problema dado, la función $\mu_{simil}(\cdot)$ será definida por una medida particular de similitud.

6.5.1.2 **Subconjuntos crisp/fuzzy**

Vamos a extender el FDT para que pueda trabajar con subconjuntos crisp/fuzzy. Los atributos nominales podrán estar formados por valores del dominio de dicho atributo que no tienen asignado un grado de pertenencia, en el caso de los subconjuntos crisp, o lo tienen asignado en el caso de los subconjuntos fuzzy. En el caso de los atributos numéricos, éstos podrán estar formados por subconjuntos fuzzy cuyos elementos son etiquetas lingüísticas de la partición fuzzy asociada a dicho atributo. Para que el FDT pueda trabajar con este tipo de valores de baja calidad, no necesitamos modificar de forma explícita los algoritmos de aprendizaje y clasificación, sino que internamente el FDT va a tratar estos valores tal y como explicamos a continuación, al igual que hace con los valores missing.

En el caso de atributos numéricos expresados mediante subconjuntos fuzzy, un ejemplo e con el atributo i-ésimo expresado con un subconjunto fuzzy y que se encuentra en un nodo N

cuyo test está basado en el atributo i, descenderá a cada nodo hijo de N que se corresponda con algún valor del subconjunto fuzzy, modificando su peso actual con el grado de pertenencia que dicho valor tiene asignado en el subconjunto:

$\chi_{childnode}(e) = \chi_N(e) \times \mu(e, B_j)$, donde B_j es la j-ésima etiqueta lingüística del subconjunto fuzzy de e y además es la etiqueta que corresponde al nodo hijo $childnode$. $\mu(e, B_j)$ es el grado de pertenencia que la etiqueta B_j tiene asignada en el subconjunto fuzzy.

Por ejemplo, si el valor del atributo i del ejemplo e es $\{0.2/templado, 0.8/caliente\}$, el ejemplo desciende al nodo hijo correspondiente a la etiqueta templado con un peso $\chi_N \times$ 0.2 y desciende al nodo hijo caliente con un peso $\chi_N \times$ 0.8, (Figura 6.6).

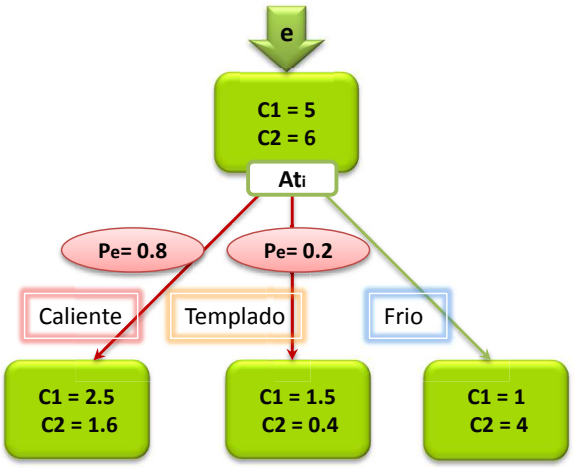


Figura 6.6: Ejemplo de subconjunto fuzzy en un atributo numérico

En el caso de atributos nominales expresados mediante subconjuntos crisps, un ejemplo e con el atributo i-ésimo expresado con un subconjunto crisp y que se encuentra en un nodo N cuyo test está basado en el atributo i, descenderá a cada nodo hijo de N que se corresponda con algún valor del subconjunto crisp, manteniendo su peso actual, es decir, $\chi_{childnode}(e) = \chi_N(e)$.

En el caso de atributos nominales expresados mediante subconjuntos fuzzy, un ejemplo e con el atributo i-ésimo expresado con un subconjunto fuzzy y que se encuentra en un nodo N cuyo test está basado en el atributo i, descenderá a cada nodo hijo de N que se corresponda con algún valor del subconjunto fuzzy, modificando su peso actual con el grado de pertenencia que dicho valor tiene asignado en el subconjunto:

$\chi_{childnode}(e) = \chi_N(e) \times \mu(e, V_j)$, donde V_j es el j-ésimo valor del dominio de i que aparece en el subconjunto fuzzy y además es el valor que tiene asociado el nodo hijo $childnode$. $\mu(e, V_j)$ es el grado de pertenencia que el valor V_j tiene asignado en el subconjunto fuzzy.

Por ejemplo, si el valor del atributo i del ejemplo e es $\{0.5/rojo, 0.5/azul\}$, el ejemplo desciende al nodo hijo correspondiente al valor rojo con un peso $\chi_N \times$ 0.5 y desciende al nodo

hijo azul con un peso $\chi_N \times 0.5$, (Figura 6.7).

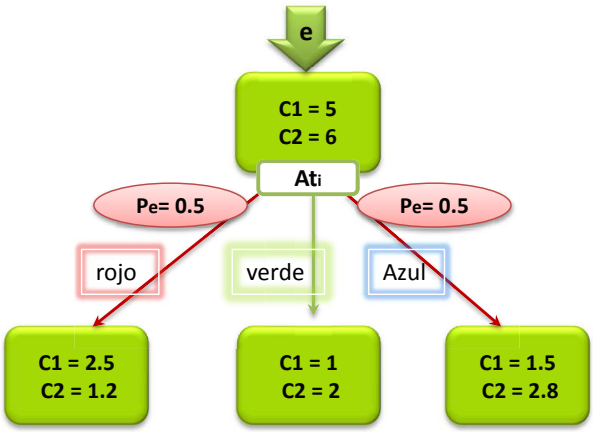


Figura 6.7: Ejemplo de subconjunto fuzzy en un atributo nominal

6.5.1.3 Clases imprecisas

Como se ha descrito en la Subssección 6.2.2, los ejemplos en el conjunto de datos son ejemplos pesados, inicialmente con peso 1. Cuando incorporamos en el árbol ejemplos cuya clase puede ser expresada por un conjunto de valores, es necesario modificar la forma en la cual tratamos este peso en la fase de aprendizaje. Así en el paso 3.1 del Algoritmo 1 deberemos calcular la ganancia de información considerando al ejemplo como perteneciente a cada clase potencial. Por esto, el ejemplo en el paso 2 del algoritmo, debe de ser replicado para cada valor de clase que contenga su atributo clase, utilizando un peso que expresa la incertidumbre sobre la clase del ejemplo. En la fase de clasificación, el ejemplo con clase imprecisa no es replicado y así su peso inicial es 1. El ejemplo desciende a través de varias ramas del árbol y activa diferentes hojas. Cada una de las hojas activadas asigna un peso a cada posible clase.

6.5.2 Cambios en los algoritmos de aprendizaje y clasificación

Los Algoritmos 1 y 2 deben de ser modificados para permitir el aprendizaje y la clasificación desde un ejemplo expresado con los tipos de valores presentados en la Sección 6.5.1. En esta subsección vamos a mostrar la modificación de estos algoritmos.

El Algoritmo 1 se modifica en los pasos 2 y 3.1 tal y como se muestra en el Algoritmo 3.

El primer cambio se debe a la posible aparición de subconjuntos crisp/fuzzy en el atributo clase. Para aquellos ejemplos con clase imprecisa, debemos de replicar los ejemplos, donde a

Algoritmo 3 Aprendizaje del Árbol de Decisión Fuzzy extendido

AprendizajeFDT$_{\text{LQD}}$(in: E, $Particiones\text{-}Fuzzy$; **out:** $Arbol\text{-}Decision\text{-}Fuzzy$)
begin

.....

2. **for** cada ejemplo e_j con clase única **do**
   $\chi_{root}(e_j) = 1$
   end for
   for cada ejemplo e_j con clase imprecisa **do**
   Replicar cada ejemplo j tantas veces como valores de clase tenga.
   Inicializar sus valores $\chi_{root}(e_j)$ de acuerdo a la información disponible sobre los valores de clase.
   end for

.....

3.1 **for** cada atributo i **do**
Calcular la ganancia de información $G_i^N, i = 1, \ldots, |M_N|$ utilizando los valores $\chi_N(e)$ teniendo en cuenta la función $\mu_{simil}(e)$ para los casos que lo requieran.
end for
.....

end

cada ejemplo le corresponde uno de los valores de clase del subconjunto crisp/fuzzy y el peso asociado al ejemplo dependerá de la información disponible acerca de las clases y de los ejemplos. Con respecto al segundo cambio, la modificación se produce debido a los posibles valores de baja calidad que pueden aparecer en los distintos atributos. Ahora para calcular el mejor atributo para dividir un nodo, seguimos utilizando la ganancia de información, sin embargo cuando nos encontramos con atributos que cuentan con valores de baja calidad debemos de utilizar la función $\mu_{simil}(e)$ para calcular los grados de pertenencia correspondientes.

El Algoritmo 2 se modifica de acuerdo al Algoritmo 4.

Algoritmo 4 Clasificación del Árbol de decisión fuzzy extendido

ClasificacionFDT$_{\text{LQD}}$(in: $Arbol\text{-}Decision\text{-}Fuzzy$, $Particiones\text{-}Fuzzy$, Ejemplo ec; **out:** Clase c)
begin

.....

2. El ejemplo ec desciende a través del árbol desde N_{root} teniendo en cuenta la función $\mu_{simil}(ec)$ para los casos que lo requieran.
   .....
5. Devuelve el conjunto de posibles clases obtenidas de las distintas hojas alcanzadas por el ejemplo ec

end

En este caso, tenemos en cuenta dos cambios, el primero implica hacer uso de la función $\mu_{simil}(ec)$ para ir calculando el grado de pertenencia en cada nodo por el cual descienda ec.

La otra modificación es respecto a la salida obtenida. Ahora, el árbol no predice una sola clase sino que predice un conjunto de posibles valores para la clase, con un peso asociado.

CAPÍTULO

7

Una extensión de un Fuzzy Random Forest

7.1 Introducción

En los últimos años, se ha observado un incremento en el desarrollo de sistemas de clasificadores múltiples, los cuales normalmente suelen obtener mejores resultados que los clasificadores individuales, [3, 21, 26, 158]. Sin embargo, estas técnicas y algoritmos deben trabajar con información que no siempre es tan precisa como se desea.

En esta sección nos centramos en el ensamble Fuzzy Random Forest (que denotaremos por FRF) propuesto en [21] y vamos a ilustrar su poder para manejar LQD. A continuación proponemos una extensión de este ensamble para tratar nueva información imprecisa. FRF utiliza el árbol de decisión fuzzy, presentado en la Sección 6.2, como clasificador base y por lo tanto, FRF aprovecha la robustez de los árboles de decisión fuzzy y de los conjuntos de árboles. También aprovecha el poder de la aleatoriedad para aumentar la diversidad, y la flexibilidad de la lógica fuzzy para gestionar los LQD.

En la siguiente sección vamos a revisar los principales elementos que componen el ensamble FRF. Después vamos a describir los tipos de LQD que soporta FRF y vamos a extenderlo para que pueda trabajar con otros tipos de LQD. Finalmente vamos a mostrar una serie de experimentos para validar el comportamiento de este ensamble. Por un lado vamos a utilizar conjuntos de datos con valores de baja calidad, para analizar el comportamiento cuando el ensamble trabaja con la verdadera naturaleza de los datos y cuando esta naturaleza se ve alterada

mediante la modificación de los datos para eliminar la imperfección. Además llevaremos a cabo otros experimentos mediante conjuntos de datos reales, que también contienen LQD, para evaluar el comportamiento del ensamble al enfrentarse a datos reales con imperfección.

7.2 Fuzzy Random Forest: Un ensamble basado en árboles de decisión fuzzy

El ensamble FRF fue originalmente presentado en [21]. Este ensamble utiliza como base un árbol de decisión fuzzy, concretamente el árbol de decisión fuzzy detallado en la Sección 6.2 pero con selección de atributos aleatoria en los nodos. El objetivo que nos planteamos de nuevo, es extender el ensamble para añadirle la funcionalidad de poder trabajar con nuevos tipos de LQD. Para llevar a cabo este objetivo, primero vamos a presentar las fases de aprendizaje y clasificación del ensamble FRF, para posteriormente mostrar las modificaciones necesarias para poder extender la funcionalidad del mismo.

7.2.1 Aprendizaje de Fuzzy Random Forest

En esta subsección vamos a detallar la fase de aprendizaje del ensamble FRF. El Algoritmo 5 expone la fase de aprendizaje de FRF de forma resumida.

Algoritmo 5 Aprendizaje del Ensamble FRF

AprendizajeFRF(**in:** E, $Particiones\text{-}Fuzzy$; **out:** $Fuzzy\text{-}Random\text{-}Forest$)
begin
Repeat
1. Extraer una muestra aleatoria MAE_i de $|E|$ ejemplos con reemplazamiento desde el conjunto de datos E.
2. Aplicar el Algoritmo 1 como **AprendizajeFDT**(MAE_i, $Particiones\text{-}Fuzzy$).

until que T árboles de decisión fuzzy estén construidos.
end

Como hemos comentado, el algoritmo base del ensamble es el árbol de decisión fuzzy presentado en el Sección 6.2. Sin embargo, para aplicar el árbol de decisión como algoritmo base del ensamble FRF, se introducen pequeños cambios para que cada árbol de decisión creado por FRF añada al modelo diversidad y al final entre todos los árboles de decisión creados obtengamos un modelo general y robusto.

7.2.2 Clasificación en Fuzzy Random Forest

Ahora vamos a describir cómo se lleva a cabo la clasificación con este ensamble. Primero vamos a introducir la notación utilizada y después vamos a presentar las estrategias alternativas

para obtener la decisión final para un ejemplo. Además vamos a presentar instancias concretas de estas estrategias (métodos de combinación) que se utilizan no sólo en este capítulo sino también a lo largo de los diferentes experimentos realizados para evaluar las diferentes técnicas propuestas.

7.2.2.1 **Notación**

La notación utilizada para el algoritmo de clasificación del ensamble FRF es la siguiente:

- T es el número de árboles de decisión fuzzy en el ensamble FRF. Utilizaremos el índice t para referirnos a un árbol particular.
- N_t es el número de hojas alcanzadas por un ejemplo en el árbol t. Utilizaremos el índice n para referirnos a una hoja particular de las alcanzadas por un árbol.
- C es el conjunto de clases. $|C|$ denota la cardinalidad de C. Utilizaremos el índice i para referirnos a una clase particular.
- e es un ejemplo que podrá ser utilizado como un ejemplo de aprendizaje o de test.
- $\chi_{t,n}(e)$ es el grado de satisfacción con el cual un ejemplo e alcanza la hoja n del árbol t.
- $Support$ para la clase i se obtiene en cada hoja como $\frac{E_i}{E_n}$ donde E_i es la suma de los grados de satisfacción de los ejemplos con clase i en la hoja n y E_n es la suma de los grados de satisfacción de todos los ejemplos en esta hoja.
- L_FRF es una matriz de tamaño $(T \times MAX_{N_t})$ con $MAX_{N_t} = max\{N_1, N_2, \ldots, N_T\}$, donde cada elemento de la matriz es un vector de tamaño $|C|$ que contiene el respaldo para cada clase provisto por cada hoja n activada sobre cada árbol t. Por lo tanto, la matriz L_FRF contiene toda la información generada por el ensamble FRF cuando se utiliza para clasificar un ejemplo e y desde la cual se va a tomar la decisión (decisión por medio de ciertos métodos de combinación). $L\_FRF_{t,n,i}$ se refiere a un elemento de la matriz que indica el respaldo a la clase i por la hoja n del árbol t.
- T_FRF es una matriz de tamaño $(T \times |C|)$ que contiene la confianza asignada por cada árbol t a cada clase i. Los elementos de la matriz se obtienen desde el respaldo para cada clase en las hojas alcanzadas cuando aplicamos algunos de los métodos de combinación. Un elemento de esta matriz se denota como $T\_FRF_{t,i}$.
- D_FRF es un vector de tamaño $|C|$ que indica la confianza asignada por el ensamble FRF a cada clase i. Los elementos de este vector se obtienen del respaldo para cada clase en las hojas alcanzadas cuando aplicamos algún método de combinación. Los elementos de este vector se denotan como D_FRF_i.

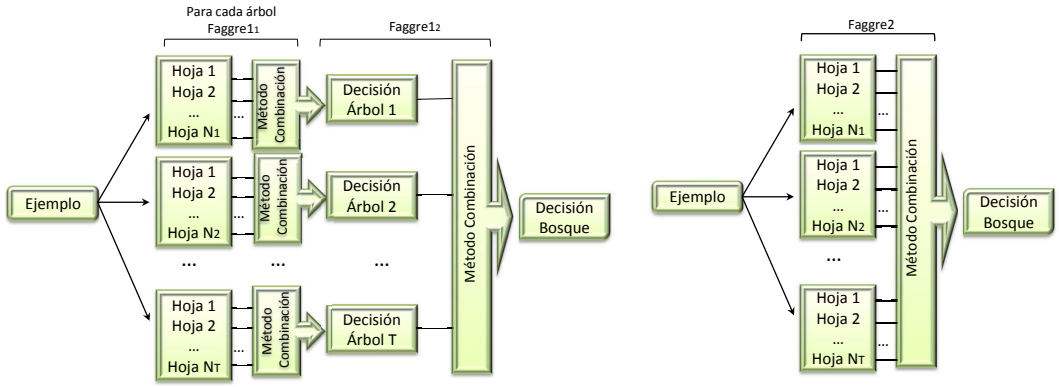
7.2.2.2 **Estrategias para el módulo clasificador en el ensamble FRF**

El módulo clasificador opera sobre los árboles del ensamble FRF utilizando una de las dos siguientes posibles estrategias:

Estrategia 1: Esta estrategia consiste en combinar la información de las diferentes hojas alcanzadas en cada árbol para obtener la decisión de cada árbol individual y luego aplicar el mismo u otro método de combinación para generar la decisión global del ensamble FRF. Para combinar la información de las hojas alcanzadas en cada árbol, se utiliza la función $Faggre1_1$ y luego para combinar las salidas obtenidas con $Faggre1_1$ utilizamos la función $Faggre1_2$. La Figura 7.1 muestra gráficamente esta estrategia, que se describe más esquemáticamente en el Algoritmo 6.

Estrategia 2: Esta segunda estrategia combina la información de todas las hojas alcanzadas por todos los árboles para generar la decisión global del ensamble FRF. Utilizamos la función $Faggre2$ para combinar la información generada por todas las hojas. La Figura 7.1 muestra gráficamente esta estrategia, que también se describe esquemáticamente en el Algoritmo 7.

Como podemos deducir de las estrategias, los árboles utilizados para constituir el ensamble FRF no clasifican los ejemplos sino que devuelven la información de las distintas hojas que activan el ejemplo. Es decir, el paso 5 del Algoritmo 2 de clasificación del FDT no se ejecuta debido a que la decisión final sobre el ejemplo la debe tomar el ensamble FRF.



Estrategia 1 Estrategia 2

Figura 7.1: Estrategias para el módulo clasificador de FRF

La función $Faggre1_1$ se utiliza en el Algoritmo 6 para obtener la matriz T_FRF. En este caso, la función $Faggre1_1$ agrega la información proporcionada por cada hoja alcanzada en el

Algoritmo 6 Clasificación FRF (Estrategia 1)

ClasificacionFRF-S1(in: ec, $Fuzzy\text{-}Random\text{-}Forest$, $Particiones\text{-}Fuzzy$; **out:** c)
begin
  DecisiondeFDT(in: ec, $Fuzzy\text{-}Random\text{-}Forest$, $Particiones\text{-}Fuzzy$; **out:** T_FRF)
  DecisionFRF(in: T_FRF; **out:** c)
end
DecisiondeFDT(in: ec, $Fuzzy\text{-}Random\text{-}Forest$, $Particiones\text{-}Fuzzy$; **out:** T_FRF)
begin

1. Ejecutar **ClasificacionFDT**(t, $Particiones\text{-}Fuzzy$, ec) para el ejemplo ec a través de cada árbol t para obtener la matriz L_FRF de información generada por el ensamble FRF.
2. **for** cada árbol t **do**
   for cada clase i **do**
   $T\_FRF_{t,i} = Faggre1_1(t, i, L\_FRF)$
   end for
   end for

end
DecisiondeFRF(in: T_FRF; **out:** c)
begin

1. **for** cada clase i **do**
   $D\_FRF_i = Faggre1_2(i, T\_FRF)$
   end for
2. El ensamble FRF asigna la clase c al ejemplo ec siendo $c = \arg \max_{i,\ i=1,\ldots,|C|} \{D\_FRF_i\}$

end

árbol. Después, los valores obtenidos en cada árbol t, serán agregados por la función $Faggre1_2$ para obtener el vector D_FRF. Este algoritmo toma como ejemplo objetivo, el ejemplo ec y el ensamble FRF, y genera el valor de clase c como la decisión del ensamble FRF.

Para implementar la estrategia 2, se simplifica el Algoritmo 6 de forma que ya no se añada la información de cada árbol, sino que se utiliza la información de todas las hojas alcanzadas por el ejemplo ec en los diferentes árboles del ensamble FRF. El Algoritmo 7 implementa la estrategia 2 y clasifica el ejemplo ec, con la intención de proporcionar el valor de clase c de tal ejemplo, este valor de clase será la decisión tomada por el ensamble FRF. La función $Faggre2$ agrega la información proporcionada por todas las hojas alcanzadas en los distintos árboles del ensamble con el fin de obtener el vector D_FRF.

Como puede observarse, independientemente de la estrategia utilizada, la salida del ensamble FRF se obtiene del vector D_FRF, el cual contiene el grado de pertenencia del ejemplo objetivo ec a cada clase. Finalmente, se asigna al ejemplo ec la clase c con máximo valor.

En el siguiente apartado presentamos algunos de los métodos de combinación. En [21] y [29] se encuentran descritos otros métodos de combinación que también están disponibles en el ensamble FRF.

Algoritmo 7 Clasificación FRF (Estrategia 2)

ClasificacionFRF-S2(in: ec, $Fuzzy\text{-}Random\text{-}Forest$, $Particiones\text{-}Fuzzy$; **out:** c)

begin

1. Ejecutar **ClasificacionFDT**(t, $Particiones\text{-}Fuzzy$, ec) para el ejemplo ec a través de cada árbol t para obtener la matriz L_FRF de información generada por el ensamble FRF.
2. **for** cada clase i **do**
   $D\_FRF_i = Faggre2(i, L\_FRF)$
   end of
3. El ensamble FRF asigna la clase c al ejemplo ec siendo $c = \arg \underset{i,\ i=1,\ldots,|C|}{\text{máx}} \{D\_FRF_i\}$

end

7.2.2.3 Métodos de combinación

Presentamos, para cada estrategia, algunas funciones de agregación $Faggre1_1$, $Faggre1_2$ y $Faggre2$. Dividimos estos métodos en métodos no entrenables, métodos entrenables explícitamente dependientes y métodos entrenables implícitamente dependientes.

Métodos no-entrenables. - En estos métodos de combinación, se aplica una transformación a la matrix L_FRF en el paso 2 de los Algoritmos 6 y 7, para que cada hoja alcanzada asigne un voto simple a cada clase mayoritaria.

Para $t = 1, \ldots, T$
  Para $n = 1, \ldots, N$
    Para $i = 1, \ldots, |C|$

$$L\_FRF - mod_{t,n,i} = \begin{cases} 1 & \text{si } i = \arg \underset{j,\ j=1,\ldots,|C|}{\text{máx}} \{L\_FRF_{t,n,j}\} \\ 0 & \text{otro caso} \end{cases}$$

Dentro de este grupo de métodos, definen los siguientes:

- *Voto Mayoritario Simple:*

  Estrategia 1 → método SM1

  La función de agregación $Faggre1_1$ del Algoritmo 6 se define como :

$$Faggre1_1(t, i, L\_FRF) = \begin{cases} 1 & \text{si } i = \arg \underset{j,\ j=1,\ldots,|C|}{\text{máx}} \left\{ \sum_{n=1}^{N_t} L\_FRF - mod_{t,n,j} \right\} \\ 0 & \text{en otro caso} \end{cases}$$

  En este método, cada árbol t asigna un voto simple a la clase más votada entre las N_t hojas alcanzadas por el ejemplo ec en el árbol.

  La función de agregación $Faggre1_2$ del Algoritmo 6 queda definida para este método como:

$$Faggre1_2(i, T\_FRF) = \sum_{t=1}^{T} T\_FRF_{t,i}$$

Estrategia 2 → método SM2

Para la estrategia 2 se define la función $Faggre2$ combinando la información de todas las hojas alcanzadas en el ensamble por el ejemplo ec. Así, la función $Faggre2$ en el Algoritmo 7 se define para este método como:

$$Faggre2(i, L\_FRF) = \sum_{t=1}^{T} \sum_{n=1}^{N_t} L\_FRF_{mod_{t,n,i}}$$

Métodos entrenables explícitamente dependientes.

- *Voto Mayoritario ponderados por Hoja:*

  En este método de combinación, se aplica una transformación a la matriz L_FRF en el paso 2 de los Algoritmos 6 y 7 para que cada hoja alcanzada asigne un voto ponderado a la clase mayoritaria. Este voto es ponderado por el grado de satisfacción que el ejemplo ec alcanza en la hoja.

  Para $t = 1, \ldots, T$
  Para $n = 1, \ldots, N$
  Para $i = 1, \ldots, |C|$

$$L\_FRF - mod_{t,n,i} = \begin{cases} \chi_{t,n}(e) & \text{si } i = \arg \underset{j,\ j=1,\ldots,|C|}{\text{máx}} \{L\_FRF_{t,n,j}\} \\ 0 & \text{en otro caso} \end{cases}$$

  Dentro de este grupo de métodos, definen los siguientes:

  Estrategia 1 → método MWL1

  Las funciones de agregación $Faggre1_1$ y $Faggre1_2$ se definen como:

$$Faggre1_1(t, i, L\_FRF) = \begin{cases} 1 & \text{si } i = \arg \underset{j,\ j=1,\ldots,|C|}{\text{máx}} \left\{ \sum_{n=1}^{N_t} L\_FRF - mod_{t,n,j} \right\} \\ 0 & \text{en otro caso} \end{cases}$$

$$Faggre1_2(i, T\_FRF) = \sum_{t=1}^{T} T\_FRF_{t,i}$$

  Estrategia 2 → método MWL2

  La función $Faggre2$ queda definida como:

$$Faggre2(i, L\_FRF) = \sum_{t=1}^{T} \sum_{n=1}^{N_t} L\_FRF_{t,n,i}$$

- *Voto Mayoritario ponderado por la Hoja y por el Árbol:*

  De nuevo, este método de combinación, aplica una transformación en la matriz L_FRF en el paso 2 de los Algoritmos 6 y 7 para que cada hoja alcanzada asigne un voto ponderado a la clase mayoritaria. El voto es ponderado por el grado se satisfacción con el cual el ejemplo ec alcanza la hoja.

  Para $t = 1, \ldots, T$
  Para $n = 1, \ldots, N$
  Para $i = 1, \ldots, |C|$

$$L\_FRF - mod_{t,n,i} = \begin{cases} \chi_{t,n}(e) & \text{si } i = \arg \max\limits_{j,\ j=1,\ldots,|C|} \{L\_FRF_{t,n,j}\} \\ 0 & \text{en otro caso} \end{cases}$$

  Además en este método se introduce una ponderación por cada árbol, testeando cada árbol individual con el conjunto de datos OOB. Sea $\overline{p} = (p_1, p_2, \ldots, p_T)$ el vector con los pesos asignados a cada árbol. Cada p_t se obtiene como $\frac{N\_success\_OOB_t}{size\_OOB_t}$ donde $N\_success\_OOB_t$ es el número de ejemplos clasificados correctamente del conjunto de datos OOB utilizando como test el árbol t-ésimo y $size\_OOB_t$ es el total de ejemplo de este conjunto de datos.

  Estrategia 1 → método MWLT1

  La función $Faggre1_1$ se define como:

$$Faggre1_1(t, i, L\_FRF) = \begin{cases} 1 & \text{si } i = \arg \max\limits_{j,\ j=1,\ldots,|C|} \left\{ \sum\limits_{n=1}^{N_t} L\_FRF - mod_{t,n,j} \right\} \\ 0 & \text{en otro caso} \end{cases}$$

  El vector $\overline{p}$ es utilizado en la definición de la función $Faggre1_2$:

$$Faggr1_2(i, T\_FRF) = \sum_{t=1}^{T} p_t \cdot T\_FRF_{t,i}$$

  Estrategia 2 → método MWLT2

  Se aplican los pesos del vector $\overline{p}$ a la estrategia 2 como sigue:

$$Faggre2(i, L\_FRF) = \sum_{t=1}^{T} p_t \sum_{n=1}^{N_t} L\_FRF_{t,n,i}$$

- *Mínimo Ponderado por Hoja y por una función de pertenencia:*

  En este método se aplica una transformación a la matriz L_FRF en el paso 2 del Algoritmo 6.

  Para $t = 1, \ldots, T$

Para $n = 1, \ldots, N$
  Para $i = 1, \ldots, |C|$
    $L\_FRF - mod_{t,n,i} = \chi_{t,n}(e) \times \frac{E_i}{E_n}$

Estrategia 1 $\rightarrow$ método MIWLF1

Definimos la función $Faggre1_1$ como:

$$Faggre1_1(t, i, L\_FRF) = \begin{cases} 1 & \text{si } i = \arg \underset{j,\ j=1,\ldots,|C|}{\text{máx}} \{min(L\_FRF - mod_{t,1,j}, \\ & \qquad \ldots, L\_FRF - mod_{t,N_t,j})\} \\ 0 & \text{en otro caso} \end{cases}$$

La función $Faggre1_2$ incorpora la ponderación definida por una función de pertenencia para cada árbol de decisión:

$$Faggre1_2(i, T\_FRF) = \sum_{t=1}^{T} \mu_{pond}\left(\frac{errors_{(OOB_t)}}{size_{(OOB_t)}}\right) \cdot T\_FRF_{t,i}$$

La función de pertenencia queda definida por $\mu_{pond}(x)$:

$$\mu_{pond}(x) = \begin{cases} 1 & 0 \leq x \leq (pmin + marg) \\ \frac{(pmax+marg)-x}{(pmax-pmin)} & (pmin + marg) \leq x \leq (pmax + marg) \\ 0 & (pmax + marg) \leq x \end{cases}$$

donde

- $pmax$ es la máxima tasa de errores en los árboles del ensamble FRF ($pmax = \text{máx}_{t=1,\ldots,T}\left\{\frac{errors_{(OOB_t)}}{size_{(OOB_t)}}\right\}$. donde $errors_{(OOB_t)}$ es el número de errores en clasificación del árbol t (usando el conjunto de datos OOB_t como conjunto test), y $size_{(OOB_t)}$ es el cardinal del conjunto de datos OOB_t. Como hemos mencionado anteriormente, los ejemplos OOB_t no se utilizan en la construcción del árbol t y por ello constituyen una muestra independiente para poder testear el árbol t. Así podemos medir la bondad de un árbol t como el número de errores cuando clasificamos usando el conjunto de ejemplos OOB_t;
- $pmin$ es la tasa mínima de errores en los árboles del ensamble FRF; y
- $marg = \frac{pmax-pmin}{4}$

Métodos entrenables implícitamente dependientes - Dentro de este grupo definimos los siguientes métodos:

- *Mínimo Ponderado por la función de pertenencia:* En este método de combinación no se aplica ninguna transformación sobre la matriz L_FRF en el paso 2 del Algoritmo 6.

Estrategia 1 → método MIWF1

La función $Faggre1_1$ se define como:

$$Faggre1_1(t,i,L\_FRF) = \begin{cases} 1 & \text{si } i = \arg \max\limits_{j,\ j=1,\ldots,|C|} \{min(L\_FRF_{t,1,j}, L\_FRF_{t,2,j}, \\ & \qquad \ldots, L\_FRF_{t,N_t,j})\} \\ 0 & \text{en otro caso} \end{cases}$$

Por su parte, la función $Faggre1_2$ incorpora la ponderación definida por la función de pertenencia anterior:

$$Faggre1_2(i,T\_FRF) = \sum_{t=1}^{T} \mu_{pond}\left(\frac{errors_{(OOB_t)}}{size_{(OOB_t)}}\right) \cdot T\_FRF_{t,i}$$

7.3 Extendiendo el ensamble FRF

El ensamble FRF es una técnica muy versátil que es capaz de adaptarse fácilmente a nuevos elementos. Una vez la hemos introducido en las secciones anteriores, vamos a describir una extensión de esta técnica para permitir la incorporación de nuevos LQD (que denotaremos por FRF_{LQD}), [42],[44],[41], [31]. Esta extensión involucra cambios en las fases de aprendizaje y clasificación de FRF. La extensión es una extensión natural ya que utilizaremos el árbol de decisión fuzzy extendido que hemos propuesto en la Sección 6.5. Por tanto, el ensamble FRF_{LQD} soportará los LQD que hemos comentado en la Subsección 6.5.1. Vamos describir los nuevos elementos y cambios necesarios para esta extensión del ensamble.

7.3.1 Cambios en los algoritmos de aprendizaje y clasificación

Los Algoritmos 5, 6 y 7 deben modificarse para permitir al ensamble FRF_{LQD} el aprendizaje y la clasificación desde un ejemplo expresado con los tipos de valores presentados en la Sección 6.5.1. En esta subsección vamos a mostrar la modificación de estos algoritmos.

El Algoritmo 5 se modifica en el paso 2 para permitir que el árbol de decisión extendido se utilice como árbol base del ensamble, como se muestra en el Algoritmo 8.

El Algoritmo 6 “Clasificación FRF (Estrategia 1)” se modifica de acuerdo al Algoritmo 9.

Finalmente, el Algoritmo 7 “Clasificación FRF_{LQD} (Estrategia 2)” se modifica en el paso 1 como se muestra en el Algoritmo 10. Los cambios son similares a los producidos en la clasificación utilizando la estrategia 1.

Algoritmo 8 Aprendizaje del Ensamble FRF_{LQD}

AprendizajeFRF$_{\text{LQD}}$(in: E, $Particiones\text{-}Fuzzy$; **out:** $Fuzzy\text{-}Random\text{-}Forest$)
begin
Repeat

1. Extraer una muestra aleatoria MAE_i de $|E|$ ejemplos con reemplazamiento desde el conjunto de datos E.
2. Aplicar el Algoritmo 3 como **AprendizajeFDT$_{\text{LQD}}$**(MAE_i, $Particion\text{-}Fuzzy$).

until que T árboles fuzzy estén construidos.
end

Algoritmo 9 Clasificación FRF_{LQD} (Estrategia 1)

ClasificacionFRF$_{\text{LQD}}$-S1(in: ec, $Fuzzy\text{-}Random\text{-}Forest$; **out:** c)
begin
DecisiondeFDT$_{\text{LQD}}$(in: ec, $Fuzzy\text{-}Random\text{-}Forest$; **out:** T_FRF)
DecisionFRF$_{\text{LQD}}$(in: T_FRF; **out:** c)
end
DecisiondeFDT$_{\text{LQD}}$(in: ec, $Fuzzy\text{-}Random\text{-}Forest$; **out:** T_FRF)
begin

1. Ejecutar **ClasificaciondeFDT$_{\text{LQD}}$**(t, $Particiones\text{-}Fuzzy$, ec) para el ejemplo ec a través de cada árbol t para obtener la matriz L_FRF de información generada por el ensamble FRF_{LQD}.

.....

end
DecisionFRF$_{\text{LQD}}$(in: T_FRF; **out:** c)
begin

.....

end

Algoritmo 10 Clasificación FRF_{LQD} (Estrategia 2)

ClasificacionFRF$_{\text{LQD}}$-S2(in: ec, $Fuzzy\text{-}Random\text{-}Forest$; **out:** c)
begin

1. Ejecutar **ClasificacionFDT$_{\text{LQD}}$**(t, $Particiones\text{-}Fuzzy$, ec) para el ejemplo ec a través de cada árbol t para obtener la matriz L_FRF de información generada por el ensamble FRF_{LQD}.

.....

end

7.4 Resultados experimentales

En esta sección vamos a describir varios resultados experimentales que muestran el comportamiento del ensamble FRF_{LQD} ante diferentes tipos de LQD. Además vamos a mostrar cómo la subtitución de los LQD por valores crisp produce diferentes resultados (diferente precisión del ensamble) a los obtenidos utilizando los LQD originales. Basándonos en estos resultados, vamos a destacar la necesidad de mantener los datos originales, expresando la verdadera naturaleza del proceso de medición o de la fuente de información.

7.4.1 **Marco experimental**

Para ello, vamos a definir un conjunto de experimentos que serán aplicados a un conjunto de datos de distinta naturaleza y definiremos los parámetros utilizamos para esos distintos experimentos.

7.4.1.1 **Experimentos**

a) Los experimentos de la subsección 7.4.2 son diseñados para medir el comportamiento del ensamble FRF_{LQD} con valores missing y datos fuzzy. Además, estos valores los vamos a sustituir por valores crisp para analizar cuál es el comportamiento y cómo afecta a los resultados, la acción de transformar los LQD en datos crisp. Esta acción es llevada a cabo en algunas ocasiones por las técnicas que no son capaces de enfrentarse de forma explícita con los LQD.

   - Los valores missing pueden afectar tanto a los atributos numéricos como a los nominales,
     - Los valores missing se reemplazan por el valor medio del atributo para los atributos numéricos y por el valor predominante para los atributos nominales.
   - Valores fuzzy,
     - los valores fuzzy se reemplazan por los intervalos que recogen su soporte,
     - los valores fuzzy se reemplazan por la media del $\alpha - corte = 1$,
     - los valores fuzzy se reemplazan por la media del $\alpha - corte = 0$,
     - los valores fuzzy se reemplazan por el centro de gravedad del conjunto fuzzy.

b) Los experimentos de la subsección 7.4.3 los hemos diseñado para medir el comportamiento del ensamble FRF_{LQD} utilizando los conjuntos de datos y los resultados propuestos en [160, 161].

Todos los experimentos realizados son validados estadísticamente mediante test estadísticos. Más concretamente seguimos la metodología propuesta por García et al. en [79], utilizando tests no paramétricos. Cuando comparamos multiples técnicas, nosotros utilizamos el test de Friedman y el procedimiento de Holm como test post-hoc. El test de Friedman es un test no paramétrico equivalente al ANOVA de medidas repetidas. Bajo la hipótesis nula, se establece que las técnicas son equivalentes, por lo que un rechazo de esta hipótesis supone la existencia de diferencias en el rendimiento de todas las técnicas estudiadas. Si se detectan diferencias, a continuación se lleva a cabo el procedimiento de Holm usado como un test post-hoc para encontrar si el control o la técnica propuesta muestra diferencias estadísticas con respecto a las

otras técnicas en comparación. Para llevar a cabo este análisis estadístico, utilizamos el paquete R, [101].

7.4.1.2 Conjuntos de datos y parámetros para el ensamble FRF_{LQD}

Para obtener estos resultados, hemos utilizados varios conjuntos de datos del repositorio de la UCI [77] y algunos conjuntos de datos reales sobre diagnóstico médico y alto rendimiento en atletas [160, 161], cuyas características se muestran en la Tabla 7.1. Esta tabla muestra el número de ejemplos ($|E|$), el número de atributos ($|A|$) y el número de clases ($|C|$) para cada conjunto de datos. "Abbr" indica la abreviación del conjunto de datos utilizado en los experimentos.

Tabla 7.1: Conjuntos de datos

Conjuntos de datos	Abbr	$\|E\|$	$\|A\|$	$\|C\|$	Conjuntos de datos	$\|E\|$	$\|A\|$	$\|C\|$
Appendicitis	APE	106	7	2	Dyslexic-12	65	12	4
Wisconsin Breast Cancer	BCW	683	9	2	Dyslexic-12-01	65	12	3
Statlog Heart Disease	HEA	270	13	2	Dyslexic-12-12	65	12	3
Iris Plants	IRP	150	4	3	Long-4	25	4	2
Pima Indian Diabetes	PIM	768	8	2	100ml-4-I	25	4	2
Attitude Smoking	SMO	2855	8	3	100ml-4-P	25	4	2

Todos los ensambles FRF_{LQD} utilizados tienen un tamaño de 100 árboles. El número de atributos elegidos aleatoriamente en un nodo dado es $\log_2(|\cdot|+1)$, donde $|\cdot|$ es el número de atributos disponibles en ese nodo, y cada árbol del ensamble FRF_{LQD} es construido a tamaño máximo (nodo puro o conjunto de atributos disponibles vacío) y sin podar. Las particiones fuzzy utilizadas en los experimentos del ensamble FRF_{LQD} son particiones creadas mediante el algoritmo de discretización $\text{OFP\_CLASS}_{\text{LQD}}$, [33, 43], que detallaremos en los siguientes capítulos. Las particiones utilizadas en los trabajos [160, 161] son uniformes.

La función $\mu_{simil}(e)$ se define para $f = 1, \ldots, F_i$, como:

$$\mu_{simil}(e) = \frac{\int(\text{mín}\{\mu_e(x), \mu_f(x)\})dx}{\sum_{f=1}^{F_i}\int(\text{mín}\{\mu_e(x), \mu_f(x)\})dx}$$

donde

- $\mu_e(x)$ representa la función de pertenencia de un valor fuzzy o intervalar del ejemplo e en el atributo i.
- $\mu_f(x)$ representa la función de pertenencia de un conjunto fuzzy de la partición del atributo i.

- F_i es la cardinalidad de la partición del atributo i.

Los últimos seis conjuntos de datos de la Tabla 7.1 contienen salidas imprecisas y hemos considerado una distribución uniforme sobre los posibles valores para la clase de cada ejemplo. Al considerar una distribuación uniforme estamos indicando que el peso asignado a cada ejemplo replicado será el mismo. Es decir, los ejemplos con clases imprecisas se replican tantas veces como valores diferentes tengan en su atributo clase y el peso inicial de cada ejemplo repetido depende del número de valores que tiene la clase imprecisa, puesto que el peso asignado al ejemplo será de $\frac{1}{n\_valores\_clases\_del\_ejemplo}$. Por ejemplo, si un ejemplo tiene un valor de clase de {rojo,amarillo}, el ejemplo se replica dos veces y el peso asignado a cada ejemplo es de 0.5.

Como ya hemos comentado en alguna ocasión en este documento, es complicado obtener conjuntos de datos del mundo real, con o sin LQD. Debido a la falta de conjuntos de datos con LQD, hemos tenido que añadir la imprecisión explícitamente en los datos. Para realizar esto hemos utilizado la herramienta NIP, [39], que presentaremos en detalle en capítulos posteriores. Esta herramienta nos permite, entre otras cosas, añadir de forma artificial LQD en los conjuntos de datos con el fin de poder evaluar las técnicas que trabajan con este tipo de información.

7.4.2 **Experimentos con datos missing y fuzzy**

Primero vamos a experimentar con conjuntos de datos crisp, a las cuales le hemos añadido un porcentaje de valores missing y valores fuzzy. Es importante destacar que este porcentaje no afecta en ninguna ocasión al atributo clase. Los experimentos con estos conjuntos de datos han sido realizados para comprobar el efecto en precisión del clasificador cuando incluimos en un conjunto de datos, LQD expresados como tales y cuando los LQD incluidos son reemplazados por otros datos. Es decir, el objetivo de estos experimentos es comprobar si podemos reemplazar con seguridad LQD por otros valores imputados o calculados a partir de los datos, permitiendo así la aplicación de técnicas más tradicionales que no pueden manejar LQD. Cada conjunto de datos ha sido modificado añadiéndole un 5 % de valores missing y un 5 % de valores fuzzy. Los conjuntos de datos con valores missing (MIS) han sido transformados en otros conjuntos de datos donde los valores missing han sido reemplazados por la media o por el valor predominante (PMV) del correspondiente atributo.

Los conjuntos de datos con valores fuzzy (FUZ) han sido transformados en varios conjuntos de datos, reemplazando los valores fuzzy por: a) Un valor intervalar (INT) para el $\alpha - corte = 0$, b) el valor del centro de gravedad (CG), c) el valor medio para el $\alpha - corte = 0$ (Mad), y d) el valor medio para el $\alpha - corte = 1$ (Mbc).

7.4.2.1 **Resultados**

Los resultados obtenidos de estos experimentos se muestran en la Tabla 7.2. La tabla indica el porcentaje medio de la precisión en clasificación del ensamble FRF_{LQD} (media y desviación estándar) con una validación cruzada de tamaño 10 repetida 5 veces.

Tabla 7.2: Resultados del ensamble FRF_{LQD} para 5% de valores missing y 5% de valores fuzzy

	Valores missing		Valores Fuzzy				
Conjuntos de datos	MIS	PMV	FUZ	INT	CG	Mad	Mbc
APE	$92.830_{8.09}$	$91.509_{8.21}$	$92.453_{8.79}$	$91.321_{9.53}$	$93.019_{8.63}$	$91.887_{9.46}$	$93.019_{8.63}$
BCW	$97.218_{1.56}$	$96.428_{2.05}$	$97.189_{2.00}$	$96.984_{1.94}$	$96.925_{2.03}$	$96.925_{2.03}$	$96.925_{2.03}$
HEA	$80.889_{7.41}$	$79.556_{7.73}$	$80.296_{7.54}$	$79.259_{7.13}$	$79.259_{7.15}$	$79.556_{6.97}$	$79.259_{7.18}$
IRP	$96.267_{4.20}$	$95.333_{5.25}$	$96.000_{4.56}$	$96.000_{4.56}$	$96.800_{4.19}$	$96.800_{4.19}$	$96.800_{4.19}$
PIM	$76.771_{4.21}$	$76.562_{5.61}$	$76.615_{4.37}$	$75.417_{5.10}$	$76.458_{5.28}$	$76.328_{4.99}$	$76.719_{5.08}$
SMO	$63.636_{2.75}$	$60.616_{2.67}$	$61.037_{2.57}$	$60.518_{2.40}$	$61.058_{2.36}$	$60.904_{2.61}$	$60.764_{2.54}$

Observando la Tabla 7.2 podemos apreciar cómo los resultados parecen cambiar considerablemente cuando transformamos el conjunto de datos original con imperfección en uno con datos imputados/calculados. Para alcanzar una conclusión más formal, vamos a llevar a cabo un estudio estadístico de las diferencias significativas siguiendo las recomendaciones de [79].

Para los conjuntos de datos con valores missing, hay diferencias significativas entre tener un 5 % de valores missing (MIS) y reemplazarlos por el valor imputado de la media o valor predominante (PMV), en todos los casos excepto para el conjunto de datos PIM. En este caso, el mejor resultado se obtiene con los conjuntos de datos que contienen un 5 % de valores missing.

Para los conjunto de datos con 5 % de valores fuzzy, el análisis es el siguiente:

- Conjunto de datos APE: FUZ, Mbc y CG tienen diferencias significativas con respecto a INT, siendo todas mejores que INT.
- Conjunto de datos BRE: no hay diferencias significativas.
- Conjunto de datos HEA: FUZ tiene diferencias significativas con respecto a INT, Mbc, Mad y CG, siendo FUZ el mejor.
- Conjunto de datos PIM: FUZ, Mbc, Mad y CG tienen diferencia significativas con respecto a INT, siendo todas mejor que INT.
- Conjunto de datos IRP: no hay diferencias significativas
- Conjunto de datos SMO: FUZ tiene diferencia significativas con respecto a INT y Mbc, siendo FUZ el mejor.

Con estos experimentos, intentamos estudiar cómo de fiable es reemplazar valores imperfectos por otros imputados, para poder aplicar técnicas tradicionales. La conclusión que extraemos tras el análisis de los datos es que cuando modificamos la naturaleza original de los datos, la calidad de los resultados tiende a decrecer, ya que durante la transformación de los LQD a datos crisp es posible que hayamos perdido información valiosa y esta pérdida termina por afectar a los resultados.

7.4.3 Experimentos con conjuntos de datos reales con datos de baja calidad

Estos experimentos van dirigidos a comprobar la precisión del ensamble FRF_{LQD} cuando trabajamos con conjuntos de datos del mundo real con valores de baja calidad. Además comparamos estos resultados con los obtenidos por el clasificador GFS propuesto en [160].

En estos experimentos hemos utilizado los conjuntos de datos que se encuentran disponibles en "http://sci2s.ugr.es/keel/" y los resultados obtenidos en los trabajos [160, 161]. Estos conjuntos de datos provienen de dos problemas diferentes del mundo real. El primero está relacionado con la composición de los equipos de atletismo de alto rendimiento y el segundo es un problema de diagnóstico médico.

7.4.3.1 Atletismo de alto rendimiento

La puntuación de un equipo de atletismo es la suma de las puntuaciones individuales de los atletas en los diferentes eventos. Es responsabilidad del entrenador equilibrar las capacidades de los diferentes atletas con el fin de maximizar la puntuación de un equipo de acuerdo a las regulaciones. Las variables que definen cada problema son las siguientes:

- Hay cuatro indicadores para el salto de longitud que se utilizan para predecir si un atleta va a pasar de un determinado umbral: la relación entre el peso y la estatura, la velocidad máxima en la carrera de 40 metros, y las pruebas (abdominales), los músculos centrales y extremidades inferiores.
- También hay cuatro indicadores para la carrera de 100 metros: la relación entre el peso y la estatura, el tiempo de reacción, el comienzo ó 20 metros de velocidad, y el máximo ó 40 metros de velocidad.

Una descripción más detallada de este problema puede encontrarse en [160]. Los conjuntos de datos utilizados en estos experimentos son los siguientes:

- Conjunto de datos "Long-4": Conjunto de datos utilizado para predecir si un atleta mejorará cierto umbral en el salto de longitud, dado los indicadores mencionados anteriormente. El conjunto cuenta con 25 ejemplos (25 atletas), 4 atributos, 2 clases, y no tiene

valores missing. Todos los atributos, incluyendo el atributo de salida, son valores intervalares.

- Conjunto de datos "100ml-4-I": Utilizado para predecir si se está logrando una marca en la carrera de 100 metros sprint. Las mediciones reales se toman por tres observadores, y se combinan en el intervalo más pequeño que los contiene. El conjunto cuenta con 25 ejemplos, 4 atributos, 2 clases, y no tiene valores missing. Todos los atributos, incluyendo el atributo de salida, son valores intervalares.
- Conjunto de datos "100ml-4-P": El mismo conjunto de datos que " 100ml-4-I", pero las mediciones han sido sustituidas por el grado subjetivo que el entrenador ha asignado a cada indicador, es decir, " el tiempo de reacción es bajo" en lugar de " el tiempo de reacción es 0.1 segundos".

Como en [160], hemos utilizado una validación cruzada de tamaño 10 para todos los conjuntos de datos. La Tabla 7.3 muestra los resultados obtenidos en [160] y los obtenidos por el ensamble FRF_{LQD}, mostrando los resultados de los 8 métodos de combinación que hemos detallado anteriormente en la Subsección 7.2.2.3. Excepto por el algoritmo crisp propuesto en [160], en esta tabla se muestra el intervalo $[media\_min\_error, media\_max\_error]$, donde min_error se calcula considerando solamente el error indicado en $errores$ y max_error se calcula considerando como error $errores + acierto\_error$, obtenido para cada conjunto de datos de acuerdo al siguiente proceso de decisión (Algoritmo 11).

Algoritmo 11 Decisión en la clasificación

for all ejemplo $e \in E$ **do**
  if $clase(e) == clase_{\text{FRF}_{\text{LQD}}}(e)$ **then** acierto++;
  else if $clase(e) \bigcap clase_{\text{FRF}_{\text{LQD}}}(e) \neq \emptyset$ **then** acierto_error++;
    else error++;
  end if
end for

Los datos de la última fila son obtenidos de [162] repitiendo el experimento 100 veces con un muestreo bootstrap del conjunto de entrenamiento y donde cada partición del test contiene 1000 tests. Esta forma de obtener una estimación del error es más costosa, pero obtiene una mejor estimación y una evaluación más exhaustiva.

En las Figuras 7.2 y 7.3 se muestran los boxplots con los mejores resultados del ensamble FRF_{LQD}. Observar que los boxplots de los experimentos con LQD no son estándares. Hemos utilizado los boxplots extendidos propuestos en [160]. En estos boxplots se utiliza una caja mostrando el 75 % percentil del error máximo y el 25 % percentil del error mínimo, así la caja

Tabla 7.3: Comparativa de los resultados de FRF_{LQD} con los obtenidos en [160] para los conjuntos de datos 100ml-4-I, 100ml-4-P y Long-4 con cuatro etiquetas/atributo

	Conjuntos de datos					
	100ml-4-I		100ml-4-P		Long-4	
Técnica	Train	Test	Train	Test	Train	Test
$FRF_{LQD\text{-}SM1}$	[0.229,0.421]	[0.250,0.443]	[0.075,0.269]	[0.120,0.333]	[0.156,0.436]	[0.217,0.483]
$FRF_{LQD\text{-}SM2}$	[0.226,0.419]	[0.233,0.427]	[0.081,0.274]	[0.100,0.297]	[0.151,0.427]	[0.217,0.483]
$FRF_{LQD\text{-}MWL1}$	[0.209,0.402]	[0.233,0.427]	[0.092,0.287]	[0.120,0.317]	[0.196,0.467]	[0.217,0.483]
$FRF_{\text{-}MWL2}$	[0.207,0.399]	[0.250,0.443]	[0.094,0.286]	[0.120,0.317]	[0.187,0.462]	[0.217,0.483]
$FRF_{LQD\text{-}MWLT1}$	[0.175,0.363]	[0.197,0.483]	[0.094,0.286]	[0.120,0.317]	[0.196,0.475]	[0.217,0.483]
$FRF_{LQD\text{-}MWLT2}$	[0.199,0.391]	[0.250,0.443]	[0.094,0.286]	[0.120,0.317]	[0.191,0.471]	[0,217,0.483]
$FRF_{LQD\text{-}MIWF1}$	[0.212,0.404]	[0.250,0.443]	[0.075,0.267]	[0.120,0.317]	[0.156,0.436]	[0.183,0.450]
$FRF_{LQD\text{-}MIWLF1}$	[0.267,0.460]	[0.273,0.467]	[0.075,0.267]	[0.100,0.297]	[0.156,0.436]	[0.183,0.450]
Crisp [160]	0.259	0.384	0.288	0.419	0.327	0.544
GGFS [160]	[0.089,0.346]	[0.189,0.476]	[0.076,0.320]	[0.170,0.406]	[0.000,0.279]	[0.349,0.616]
GGFS [162]	–	[0.176,0.378]	–	[0.176,0,355]	–	[0.321,0.590]

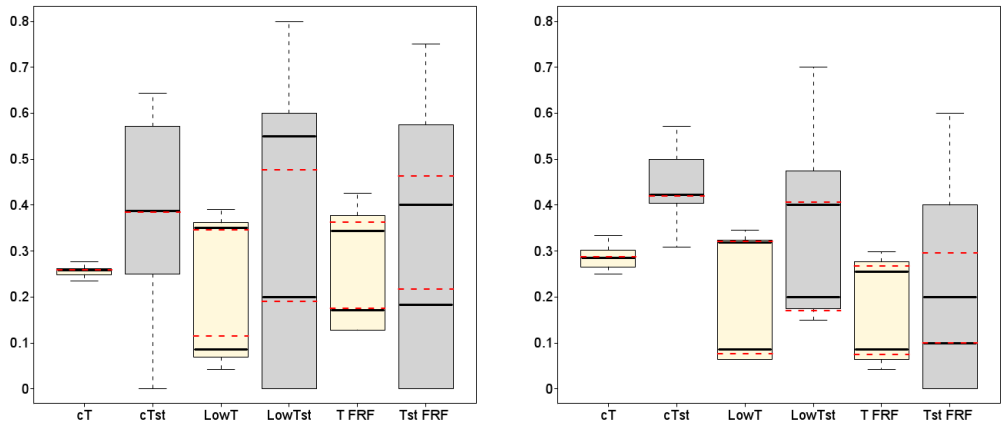


Figura 7.2: Boxplots de los conjuntos de datos 100ml-4-I y 100ml-4-P con cinco etiquetas/atributo. cT y cTst son los resultados para el train y test respectivamente del algoritmo crisp propuesto en [160]. LowT y LowTst son los resultados para el train y test del algoritmo extendido propuesto en [160].T-FRF y Tst-FRF son los resultados del train y test $FRF_{LQD\text{-}MWLT1}$ para 100ml-4-I y $FRF_{LQD\text{-}MIWF1}$ para 100ml-4-P.

muestra al menos el 50 % de los datos. Además, en ellos se representa el valor intervalar medio del resultado máximo y mínimo. Por esta razón las cajas tienen dos marcas en su interior.

Los resultados obtenidos por el algoritmo GGFS extendido en [160] y por el ensamble FRF_{LQD}, son bastante prometedores porque ellos representan la información de una forma más natural y apropiada y en este problema, permitimos la obtención de conocimiento de los

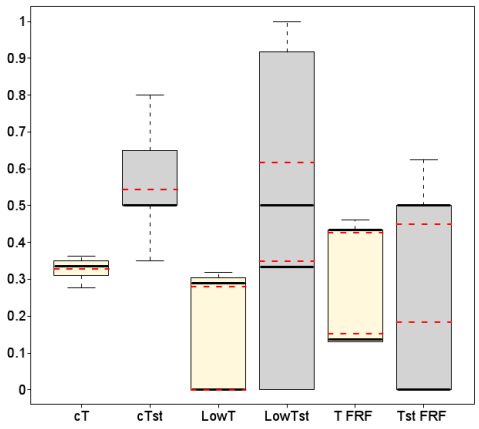


Figura 7.3: Boxplots del conjunto de datos Long-4 con cinco etiquetas/atributo. ccT y cTst son los resultados para el train y test respectivamente del algoritmo crisp propuesto en [160]. LowT y LowTst son los resultados para el train y test del algoritmo extendido propuesto en [160]. T-FRF y Tst-FRF son los resultados del train y test para $FRF_{LQD\text{-}SM2}$.

entrenadores mediante rango de valores y términos lingüísticos. Los resultados del ensamble FRF_{LQD} son muy competitivos.

7.4.3.2 **Diagnóstico de la dislexia**

La dislexia es una discapacidad de aprendizaje en las personas con un coeficiente intelectual normal, y sin más problemas físicos o psicológicos que puedan explicar dicha discapacidad. La dislexia puede hacerse evidente al principio de la infancia, con dificultad para formar oraciones y mediante los antecedentes familiares. El reconocimiento del problema es muy importante con el fin de dar al bebé una enseñanza apropiada. Una descripción más detallada de este problema se puede encontrar en [160, 161].

En estos experimentos, hemos utilizado tres conjuntos de datos diferentes. Sus nombres son "Dyslexic-12', "Dyslexic-12-01" y "Dyslexic-12-12". Cada conjunto de datos está formado por 65 ejemplos, con 4 clases y 12 atributos. La variable de salida para cada un de estos conjuntos de datos es un subconjunto de las siguientes etiquetas:

- No disléxico.
- Control y revisión.
- Disléxico.
- Inatención, hiperactividad u otros problemas

Tanto la entrada como la salida tienen imprecisión y valores missing. Estos tres conjuntos de datos se diferencian solamente en sus salidas:

- “Dyslexic-12” contiene las cuatro clases mencionadas.
- “Dyslexic-12-01” no utiliza la clase “control y revision”, cuyos miembros son incluidos en la clase “no disléxico”.
- “Dyslexic-12-12” no utiliza la clase “control y revision”, cuyos miembros son incluidos en la clase “disléxico”.

Todos los experimentos son repetidos 100 veces para un bootstrap con un muestreo con reemplazamiento del conjunto de entrenamiento. El conjunto test se compone de los ejemplos del conjunto de datos OOB. De nuevo utilizamos los boxplots no estándares extendidos propuestos en [160].

Los resultados de estos experimentos se muestran en la Tabla 7.4. Comparamos los resultados obtenidos por el ensamble FRF_{LQD} con los mejores obtenidos en [161]. De nuevo se muestra en la tabla, el intervalo $[media\_min\_error, media\_max\_error]$ obtenido para cada conjunto de datos de acuerdo al proceso de decisión descrito en el apartado 6.5.1.3. Las diferencias significativas estadísticas se muestran en las Figuras 7.4 y 7.5, donde el ensamble FRF_{LQD} tiene un mejora significativa sobre el algoritmo crisp GGFS.

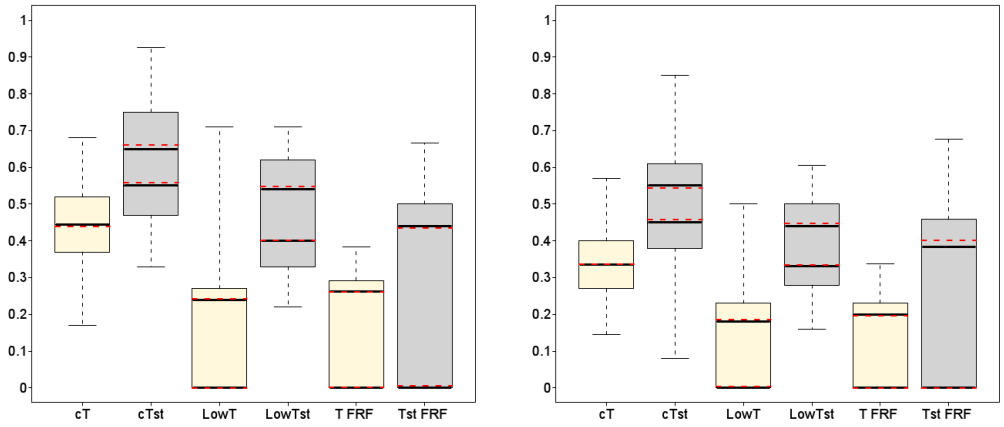


Figura 7.4: Boxplots Dyslexic-12 y Dyslexic-12-01 con cuatro etiquetas/atributo. cT y cTst son los resultados para el train y test, respectivamente, del algoritmo crisp CF_0 propuesto en [161]. LowT y LowTst son los resultados del algoritmo extendido CF_0 propuesto en [161]. T-FRF y Tst-FRF son los resultados para el train y test de $\text{FRF}_{\text{LQD-SM2}}$.

Tabla 7.4: Comparativa de los resultados del ensamble FRF_{LQD} y los obtenidos en [161] para los conjuntos de datos Dyslexic-12, Dyslexic-12-01 y Dyslexic-12-12.

	Conjuntos de datos					
	Dyslexic-12		Dyslexic-12-01		Dyslexic-12-12	
Técnica	Train	Test	Train	Test	Train	Test
$FRF_{LQD\text{-}SM1}$ (4 etiquetas)	[0.000,0.235]	[0.011,0.441]	[0.000,0.288]	[0.000,0.443]	[0.000,0.265]	[0.000,0.434]
$FRF_{LQD\text{-}SM2}$ (4 etiquetas)	[0.000,0.231]	[0.014,0.447]	[0.000,0.106]	[0.000,0.402]	[0.000,0.247]	[0.000,0.415]
$FRF_{LQD\text{-}MWL1}$ (4 etiquetas)	[0.002,0.262]	[0.004,0.436]	[0.000,0.308]	[0.000,0.436]	[0.000,0.290]	[0.000,0.435]
$FRF_{LQD\text{-}MWL2}$ (4 etiquetas)	[0.001,0.260]	[0.006,0.443]	[0.000,0.312]	[0.000,0.445]	[0.000,0.292]	[0.000,0.436]
$FRF_{LQD\text{-}MWLT1}$ (4 etiquetas)	[0.002,0.260]	[0.005,0.435]	[0.000,0.279]	[0.000,0.425]	[0.000,0.286]	[0.000,0.432]
$FRF_{LQD\text{-}MWLT2}$ (4 etiquetas)	[0.002,0.261]	[0.005,0.434]	[0.000,0.286]	[0.000,0.430]	[0.000,0.285]	[0.000,0.432]
$FRF_{LQD\text{-}MIWF1}$ (4 etiquetas)	[0.000,0.234]	[0.010,0.461]	[0.000,0.250]	[0.000,0.443]	[0.000,0.250]	[0.000,0.439]
$FRF_{LQD\text{-}MIWLF1}$ (4 etiquetas)	[0.000,0.235]	[0.011,0.470]	[0.000,0.253]	[0.000,0.447]	[0.000,0.254]	[0.000,0.447]
Crisp CF^0 (4 etiquetas) [161]	0.444	[0.572,0.694]	0.336	[0.452,0.533]	0.390	[0.511,0.664]
GGFS (4 etiquetas) [161]	–	[0.421,0.558]	–	[0.219,0.759]	–	[0.199,0.757]
GGFS CF^0 (4 etiquetas) [161]	[0.003,0.237]	[0.405,0.548]	[0.005,0.193]	[0.330,0.440]	[0.003,0.243]	[0.325,0.509]
$FRF_{LQD\text{-}SM1}$ (5 etiquetas)	[0.000,0.250]	[0.004,0.465]	[0.000,0.307]	[0.000,0.459]	[0.000,0.275]	[0.000,0.443]
$FRF_{LQD\text{-}SM2}$ (5 etiquetas)	[0.000,0.251]	[0.001,0.515]	[0.000,0.205]	[0.000,0.421]	[0.000,0.264]	[0.000,0.473]
$FRF_{LQD\text{-}MWL1}$ (5 etiquetas)	[0.003,0.284]	[0.008,0.448]	[0.000,0.339]	[0.000,0.448]	[0.000,0.308]	[0.000,0.451]
$FRF_{LQD\text{-}MWL2}$ (5 etiquetas)	[0.001,0.278]	[0.004,0.443]	[0.000,0.335]	[0.000,0.449]	[0.000,0.305]	[0.000,0.447]
$FRF_{LQD\text{-}MWLT1}$ (5 etiquetas)	[0.002,0.280]	[0.007,0.442]	[0.000,0.309]	[0.000,0.431]	[0.000,0.307]	[0.000,0.435]
$FRF_{LQD\text{-}MWLT2}$ (5 etiquetas)	[0.001,0.277]	[0.004,0.440]	[0.000,0.309]	[0.000,0.435]	[0.000,0.303]	[0.000,0.445]
$FRF_{LQD\text{-}MIWF1}$ (5 etiquetas)	[0.001,0.229]	[0.007,0.480]	[0.000,0.250]	[0.000,0.446]	[0.000,0.254]	[0.000,0.446]
$FRF_{LQD\text{-}MIWLF1}$ (5 etiquetas)	[0.001,0.236]	[0.007,0.505]	[0.000,0.269]	[0.000,0.470]	[0.000,0.259]	[0.000,0.459]
Crisp CF^0 (5 etiquetas) [161]	0.556	[0.614,0.731]	0.460	[0.508,0.605]	0.485	[0.539,0.692]
GGFS (5 etiquetas) [161]	–	[0.490,0.609]	–	[0.323,0.797]	–	[0.211,0.700]
GGFS CF^0 (5 etiquetas) [161]	[0.038,0.233]	[0.480,0.621]	[0.000,0.187]	[0.394,0.522]	[0.000,0.239]	[0.393,0.591]

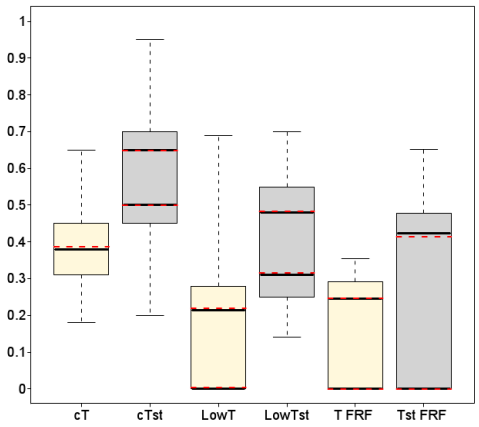


Figura 7.5: Boxplots Dyslexic-12-12 con cuatro etiquetas/atributo. cT y cTst son los resultados para el train y test respectivamente del algoritmo crisp CF_0 propuesto en [161]. LowT y LowTst son los resultados del algoritmo extendido CF_0 propuesto en [161]. T-FRF y Tst-FRF son los resultados para el train y test de $FRF_{LQD\text{-}SM2}$.

CAPÍTULO

8

K-vecinos más cercanos desde datos de baja calidad

8.1 Introducción

Dentro de la fase de minería de datos del proceso del AID, uno de las técnicas para clasificación más conocida es k-vecinos más cercanos, (KNN), donde k es el número de vecinos considerados [70]. Esta técnica presenta las siguientes ventajas:

- La técnica KNN es una técnica predictiva que puede inferir tanto atributos nominales (el valor de atributo más frecuente entre los k vecinos más cercanos) como atributos numéricos (la media entre los valores de los k vecinos más cercanos).
- KNN permite la existencia de valores missing en los datos.
- Carece de fase de aprendizaje. KNN no crea modelos explícitos como un árbol de decisión o un conjunto de reglas, sino que es el propio conjunto de datos el que se utiliza como un modelo "vago". Así, la técnica puede ser adaptada fácilmente para predecir cualquier atributo.

Sin embargo, KNN sufre de algunos inconvenientes como la necesidad de un elevado requerimiento de memoria para almacenar todos los ejemplos que constituyen el conjunto de aprendizaje, la baja eficiencia durante el funcionamiento de la regla de decisión debido al elevado cálculo de similaridades entre los ejemplos test y los ejemplos de aprendizaje, y la poca tolerancia al ruido ya que usa todos los ejemplos como relevantes. Sin embargo, la limitación

más importante de las tres mencionadas se produce al intentar buscar los ejemplos más cercanos al ejemplo que intentamos inferir. En este caso el tema es crítico porque la técnica debe de recorrer todos los ejemplos del conjunto de datos y dependiendo del número de ejemplos de éste y del número de atributos de cada ejemplo, el tiempo empleado podría ser excesivo. Por lo tanto, el tamaño del conjunto de datos es un problema. Sin embargo, en la literatura podemos encontrar técnicas que intentan resolver esta limitación, creando conjuntos reducidos de aprendizaje compuestos solamente por ejemplos prototipo. Una revisión de estas técnicas la podemos encontrar en [78].

En este capítulo nos vamos a centrar en la técnica KNN para clasificar, donde el objetivo por tanto es predecir el atributo clase. En las siguientes secciones, comenzaremos detallando la técnica KNN para clasificar desde un conjunto de datos que carece de valores de baja calidad. A continuación, vamos a extender esta técnica para que pueda trabajar con LQD que aparece explícitamente en los datos. La técnica extendida la denominamos KNN_{LQD}

8.2 K-vecinos más cercanos desde datos crisp

La técnica k-vecinos más cercanos para clasificar es un procedimiento no paramétrico que obtiene el valor de la clase de un ejemplo e basándose en la información proporcionada por los k vecinos más cercanos a e.

Sea E un conjunto de ejemplos descritos por un conjunto A de atributos y C el conjunto de posibles clases de los ejemplos. Los atributos pueden ser nominales o numéricos y cada ejemplo debe de contener al menos dos atributos, siendo uno de ellos nominal y considerado como la clase del ejemplo. Sin perder generalidad vamos a suponer que la clase de un ejemplo es el último atributo.

Como ya hemos mencionado, la técnica KNN no tiene fase de aprendizaje ya que el modelo se compone del conjunto de ejemplos iniciales E. Por lo tanto, el Algoritmo 12 muestra solamente la fase de inferencia, es decir, de clasificación donde se obtiene el valor de la clase de un ejemplo dado ec desde otros ejemplos con todos sus valores crisp. Así, si ec es un ejemplo con su valor de clase desconocida o missing, la técnica obtiene dicha clase a partir de la información proporcionada por los k vecinos más cercanos a ec del conjunto E.

Algoritmo 12 Clasificación mediante KNN

KNN(in: ec, E, k ($1 \leq k \leq |E|$); **out:** $Valor\_clase$ para ec)
begin

1. Calcular la distancia $d(e_j, ec)$ entre ec y cada $e_j \in E$
2. Seleccionar $K \subseteq E$, el conjunto de los k ejemplos de E más cercanos a ec
3. Asignar al ejemplo ec la clase más frecuente en K

end

Para calcular la distancia entre ejemplos es necesario definir una medida que nos indique cómo de próximos están. Una de las medidas de distancia más usadas es la distancia Euclídea normalizada, la cual calcula la distancia entre dos ejemplos e_i, e_j de la siguiente manera:

$$d(e_i, e_j) = \sqrt{\sum_h \frac{(e_i^h - e_j^h)^2}{(\sigma^h)^2}}$$

donde h identifica a los atributos continuos que describen el ejemplo y $(\sigma^h)^2$ la varianza muestral de los valores del atributo e^h. También podemos utilizar cualquier otra medida que nos permita trabajar con atributo heterogéneos (nominales y numéricos).

Com podemos ver en el Algoritmo 12, el valor de k es un parámetro externo y el valor más frecuentemente utilizado es $k = \sqrt{|E|}$, [70].

8.3 K-vecinos más cercanos desde datos de baja calidad

Una vez que hemos presentado la técnica de k-vecinos más cercanos que trabaja con datos crisp, vamos a extender la técnica con el fin de que pueda trabajar con LQD. Esta extensión la llamaremos KNN_{LQD}. Para unificar el tratamiento de los diferentes tipos de LQD permitidos por la nueva extensión propuesta, vamos a representar los valores que aparecen en el conjunto de datos de entrada en un formato común:

a) Para los atributos numéricos, los diferentes tipos de valores se expresan como una cuádrupla representando la función de pertenencia asociada (Figura 8.1). En el caso de los valores missing, min y max se corresponden con los valores globales mínimo y máximo para ese atributo en el conjunto de datos de entrada.

b) Para el caso de los atributos nominales, representamos un valor nominal crisp como un subconjunto del dominio de dicho atributo compuesto por un único valor (por ejemplo, {rojo}). Los valores imprecisos podrán ser un subconjunto crisp (por ejemplo, {rojo, azul}), un subconjunto fuzzy (por ejemplo, {0.2/rojo,0.8/azul}). Por último, un valor nominal missing es representado como un subconjunto crisp que contiene todos los posibles valores del atributo.

Como parte de la técnica KNN_{LQD}, es necesario definir una métrica que sirva como medida de distancia. Vamos a definir, de forma general, una medida que sea capaz de trabajar con LQD. La medida $d_{LQD}(\cdot, \cdot)$, entre dos ejemplos e_j y e_h, la definimos como:

$$d_{LQD}(e_j, e_h) = \sqrt{\frac{\sum_{i=1,\ldots,|A|; i \neq n} f(e_j^i, e_h^i)^2}{|A| - 1}}$$

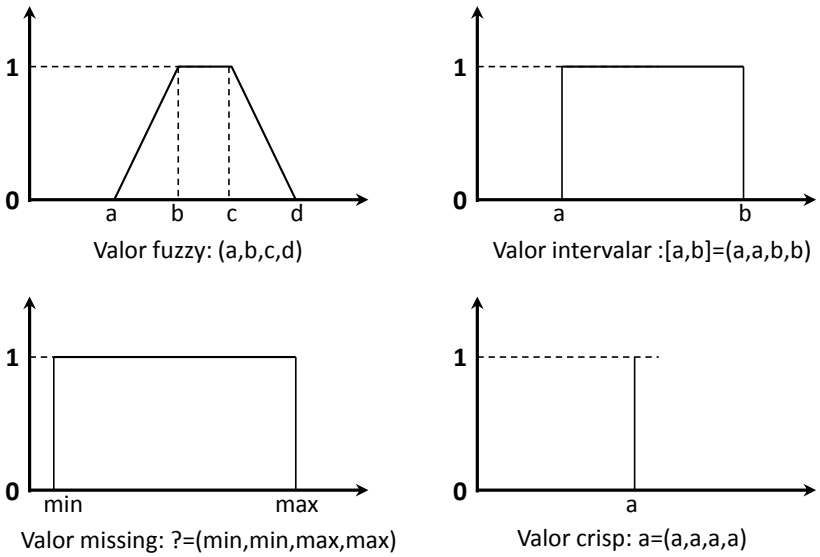


Figura 8.1: Atributos numéricos como cuádruplas

donde la función $f(\cdot,\cdot)$ es una medida heterogénea de la distancia entre dos atributos cuyos valores pueden ser de baja calidad expresados tal y como hemos comentado anteriormente.

Por lo tanto, la función $f(\cdot,\cdot)$ la definimos como:

$$f(e_j^i, e_h^i) = \begin{cases} f_1(e_j^i, e_h^i) & \text{si i es numérico} \\ \\ f_2(e_j^i, e_h^i) & \text{si i es nominal} \end{cases}$$

Utilizando la función $d_{LQD}(\cdot,\cdot)$ obtenemos el conjunto de k vecinos más cercanos al un ejemplo dado ec. A este conjunto lo llamamos K_{LQD} y será a partir del cual obtengamos la clase del ejemplo ec. Ahora el problema con el cual nos encontramos es que los valores de clase pueden contener valores de baja calidad por lo que la clase más frecuente entre los vecinos podrá ser una clase imprecisa. En el Algoritmo 13 se muestra el proceso para obtener el valor de clase de un ejemplo dado ec.

Algoritmo 13 Clasificación mediante KNN_{LQD}

KNN_{LQD}(in: ec, E, k ($1 \leq k \leq |E|$); **out:** $Valor\_clase$ para ec)
begin

1. Calcular la distancia $d_{LQD}(e_j, ec)$ entre ec y cada $e_j \in E$
2. Seleccionar $K_{LQD} \subseteq E$, el conjunto de los k ejemplos de E más cercanos a ec
3. $clase(ec) = \{\frac{\sum_{e_j \in K_{LQD}} \mu(e_j, c_i)}{|K_{LQD}|} / c_i\}$; $c_i \in C$ donde $\mu(e_j, c_i)$ es el grado de creencia del valor de clase c_i en el j-ésimo ejemplo

end

Por lo tanto, la salida del Algoritmo 13 puede ser un valor de clase impreciso expresado mediante un subconjunto crisp/fuzzy.

CAPÍTULO

9

Conclusiones parciales y aportaciones

9.1 Conclusiones

A lo largo de esta PARTE II hemos llevado a cabo la descripción de tres técnicas para llevar a cabo la fase minería de datos del AID.

En el Capítulo 6, hemos presentado la extensión de un árbol de decisión fuzzy (FDT) para permitir el tratamiento de LQD. A esta extensión la hemos denominado FDT_{LQD}. Ambas técnicas (tanto FDT como FDT_{LQD}) requieren una discretización fuzzy de los atributos numéricos. La extensión FDT_{LQD} es capaz de tratar de forma explícita con los mismos tipos de LQD que FDT (valores fuzzy de la partición de entrada y valores missing) pero además es capaz de tratar con valores fuzzy diferentes a los de la partición fuzzy de entrada, valores intervalares, subconjuntos crisp/fuzzy y clases imprecisas.

Una vez presentada la técnica FDT_{LQD}, la hemos utilizado como clasificador base del ensamble de árboles de decisión fuzzy FRF. De esta forma, en el Caítulo 7 hemos presentado la extensión del ensamble FRF para el tratamiento de LQD. Esta extensión la hemos denominado FRF_{LQD}. FRF_{LQD} es capaz de trabajar con los mismos tipos de LQD que FDT_{LQD} combinando por lo tanto la capacidad de tratar con LQD de FDT_{LQD} y la robustez de las técnicas basadas en clasificadores múltiples.

Para evaluar el comportamiento y la robustez tanto de FDT_{LQD} como de FRF_{LQD} hemos llevado a cabo una serie de experimentos con diversos conjuntos de datos. Por un lado hemos utilizado conjuntos de datos del repositorio de la UCI, [77], los cuales han sido modificados

para añadirles LQD. Con esta experimentación hemos mostrado la ventaja de trabajar con la verdadera naturaleza de los datos frente a llevar a cabo transformaciones de los mismos para adaptarlos a las técnicas con las que debemos trabajar. Por otro lado, hemos utilizado conjuntos de datos reales con LQD sobre atletismo de alto rendimiento y sobre la dyslexia. Con estos conjuntos de datos hemos analizado la precisión en clasificación de las técnicas propuestas comparándolas con otras propuestas en la literatura. Tras esta experimentación podemos concluir que los resultados obtenidos con ambas técnicas son competitivos y que el comportamiento es estable y robusto.

Por último, en el Capítulo 8, hemos presentado una extensión de la regla de vecinos más cercanos para clasificar (KNN) para que pueda trabajar con LQD. A esta extensión la hemos denominado KNN_{LQD}. La extensión es capaz de trabajar con ejemplos con valores crisp, valores fuzzy, valores intervalares, subconjuntos crisp/fuzzy y valores missing donde todos deben ser expresados mediante los cuatro valores que definen una función de pertenencia trapezoidal. Para ello es necesario disponer de una función distancia heterogénea que pueda trabajar con este tipo de valores. Además, la salida proporcionada por esta técnica puede ser imprecisa.

9.2 Aportaciones más relevantes

[32] J.M. Cadenas, M.C. Garrido, R. Martínez. Metodologías basadas en SoftComputing para el diseño de técnicas de Clasificación Automática. Proceedings of CAEPIA 2011 (Doctoral Consortium), Tenerife (España) 2011.

[41] J.M. Cadenas, M.C. Garrido, R. Martínez, P.P. Bonissone. Towards the Learning from Low Quality Data in a Fuzzy Random Forest ensemble. IEEE International Conference Fuzzy System (FUZZIEEE 2011), 2897–2904, Taipei, Taiwan, 2011.

Congreso CORE A

[29] J.M. Cadenas, M.C. Garrido, R. Martínez, A. Martínez. Consensus Operators for Decision Making in Fuzzy Random Forest Ensemble. International Conference on Intelligen Systems Design and Applications (ISDA 2011), 1377–1382, Cordoba, España, 2011.

[31] J.M. Cadenas, M.C. Garrido, R. Martínez. Learning in a Fuzzy Random Forest Ensemble from Imperfect Data. International Conference on Systems, Man, and Cybernetics (IEEE SMC 2011), 277–282, Anchorage, Alaska, 2011.

[42] J.M. Cadenas, M.C. Garrido, R. Martínez, P.P. Bonissone. Extending of information processing in a Fuzzy Random Forest ensemble. Soft Computing 16(5), 845–861, 2012.

JCR 1.124

[44] J.M. Cadenas, M.C. Garrido, R. Martínez, R.A. Díaz-Valladares. “FRF Software”. Registro de la propiedad intelectual. 07-03-2014. nº asiento registral 08/2014/0198.

Patente

[45] J.M. Cadenas, M.C. Garrido, R. Martínez, A. Martínez. Regla K_M-vecinos más cercanos en bases de datos de baja calidad. Actas del VI Simposio de Teoría y Aplicaciones de Minería de Datos (TAMIDA 2013) - Multiconference CAEPIA, 1303–1312, Madrid, Spain, 2013.

Parte III

PREPROCESAMIENTO DE DATOS DE BAJA CALIDAD

En esta PARTE III vamos a presentar las técnicas de preprocesamiento de datos que hemos desarrollado para que puedan tratar con LQD. Entre la variedad de procedimientos que existen en la literatura para llevar a cabo la limpieza y mejora de la calidad de los datos antes de abordar la fase de minería de datos, nos vamos a centrar en la discretización de atributos numéricos, en la selección de atributos, en la imputación de valores missing y en la selección de ejemplos.

En el Capítulo 10 vamos a presentar nuestra propuesta de una técnica de discretización fuzzy (OFP_CLASS) para particionar el dominio de atributos numéricos en un conjunto de particiones fuzzy. Esta técnica es categorizada como híbrida y está compuesta por un árbol de decisión fuzzy y un algoritmo genético. Sobre esta técnica llevamos a cabo dos mejoras. En la primera mejora, OFP_CLASS es extendido para que sea capaz de tratar con LQD de forma explícita. La técnica resultante de esta primera mejora es denominada OFP_CLASS$_{\text{LQD}}$. En la segunda mejora, añadimos un proceso de bagging en OFP_CLASS$_{\text{LQD}}$ para hacerlo más robusto y estable cuando trate con conjuntos de datos donde el número de ejemplos es inferior al producto del número de atributos y número de clases que los componen. La técnica resultante es denominada BAGOFP_CLASS. Para cada una de las versiones de las técnicas de discretización fuzzy vamos a realizar una evaluación de sus resultados utilizando para ello tanto conjuntos de datos reales como conjuntos de datos del repositorio de la UCI, [77]. También realizaremos una comparación entre las distintas versiones propuestas para validar que BAGOFP_CLASS es robusta y obtiene resultados más satisfactorios que sus versiones anteriores.

En el segundo capítulo de esta parte, Capítulo 11, proponemos una técnica para llevar a cabo la selección de atributos que permite trabajar de forma directa con LQD en los conjuntos de datos. La técnica la hemos denominado FRF-fs. Nuestro objetivo con esta técnica es conseguir un resultado igual de satisfactorio y aceptable tanto al trabajar con conjuntos de datos con/sin LQD. Además también pretendemos proporcionar al usuario la máxima información aportándole no solo un subconjunto de atributos óptimo, sino también un ranking de atributos. Para conseguir toda esta información hemos utilizado como elemento base en FRF-fs el ensamble FRF$_{\text{LQD}}$ que hemos presentado en el Capítulo 7. Para validar la técnica de selección de atributos propuesta vamos a tratar con conjuntos de datos de microarrays y conjuntos de datos del repositorio de la UCI, a los cuales les vamos a añadir ciertos porcentajes de LQD para evaluar la robustez de la técnica frente a estos tipos de datos. Además, llevaremos a cabo

diversas comparativas; por un lado, realizaremos el ajuste de los parámetros de FRF-fs y por otro lado, llevaremos a cabo una comparación de la técnica con otras técnicas existentes en la literatura.

Por último y como un trabajo que se encuentra en su fase inicial, en el Capítulo 12 proponemos una técnica de imputación de valores missing que es capaz de trabajar con LQD y una técnica para llevar a cabo la selección de ejemplos a partir de LQD (CNN_{LQD}). Para diseñar ambas técnicas nos hemos basado en la técnica de minería de datos KNN_{LQD} presentada en el Capítulo 8. Uno de los elementos fundamentales en la definición de estas técnicas es la medida de distancia para calcular los vecinos más cercanos a un ejemplo dado. En este capítulo proponemos una medida de distancia heterogénea capaz de tratar con LQD. Llvaremos a cabo la evaluación de ambas técnicas usando conjuntos de datos del repositorio de la UCI a los que hemos añadido ciertos porcentajes de LQD. Dado que uno de los problemas principales en las técnicas basadas en vecinos más cercanos es el coste computacional en el cálculo de las distancias entre ejemplos, reducimos los conjuntos de datos usando la técnica CNN_{LQD} y llevamos a cabo la imputación de valores missing usando los conjuntos de datos originales y sus versiones reducidas para realizar una comparación de los resultados.

El esquema general de desarrollo de esta parte, se muestra en la Figura 9.1.

Figura 9.1: Esquema general de la parte

CAPÍTULO

10

Discretización de atributos numéricos

10.1 Introducción

Tal y como se ha comentado en capítulos anteriores, la discretización de valores numéricos es una etapa crucial en los clasificadores que por su naturaleza no pueden trabajar con datos numéricos, puesto que el resultado en clasificación depende de la calidad de ella. Además, hay técnicas que aunque pueden tratar con datos numéricos, obtienen mejor rendimiento cuando estos valores están discretizados, debido a que dicha discretización reduce el número de valores numéricos y facilita un aprendizaje más rápido y con mayor precisión. Por estas razones y otras expuestas anteriormente, hemos diseñado una técnica, que hemos ido mejorando mediante diferentes propuestas, para llevar a cabo el proceso de discretización de atributos con valores numéricos mediante la creación de particiones fuzzy. Todas las técnicas propuestas se componen de dos etapas: En la primera etapa se utiliza un árbol de decisión para determinar un conjunto de posibles puntos de corte que dividen el dominio; y en la segunda etapa un algoritmo genético se encarga de formar y optimizar las particiones fuzzy a partir de los puntos de corte generados en la primera etapa.

Las particiones creadas por la técnica de discretización propuesta y las diferentes mejoras propuestas de la misma a lo largo de este capítulo son fuertes y completas, es decir, son particiones válidas para el árbol de decisión fuzzy que se ha detallado en el Capítulo 6.

En las siguientes secciones vamos a explicar en detalle las técnicas de discretización fuzzy propuestas, justificando en cada una de ellas las modificaciones llevadas a cabo y las diferencias

que presentan entre ellas.

10.2 OFP_CLASS: Una técnica de discretización fuzzy

En esta sección introducimos la técnica OFP_CLASS, (Optimización de Particiones Fuzzy para CLASSificación) para discretizar atributos numéricos a través de particiones fuzzy, [33, 43]. La técnica consiste en una combinación de algoritmos y de acuerdo con las definiciones de la Sección 4.2.2, podemos taxonomizar a la técnica propuesta como supervisada, local, top down e incremental, además de usar la entropía como medida para obtener y evaluar los intervalos.

La técnica OFP_CLASS está compuesta de dos etapas como puede observarse gráficamente en la Figura 10.1. En la primera etapa se definen un conjunto de intervalos crisp mediante la búsqueda de puntos de corte que dividen el dominio. En la segunda, y usando los intervalos definidos en la primera etapa, se forman y optimizan las particiones fuzzy. Para la primera etapa se hace uso de un árbol de decisión, mientras que para la segunda se utiliza un algoritmo genético con el cual se determina la cardinalidad y la calidad de las particiones. A continuación vamos a presentar en detalle cada una de estas dos etapas.

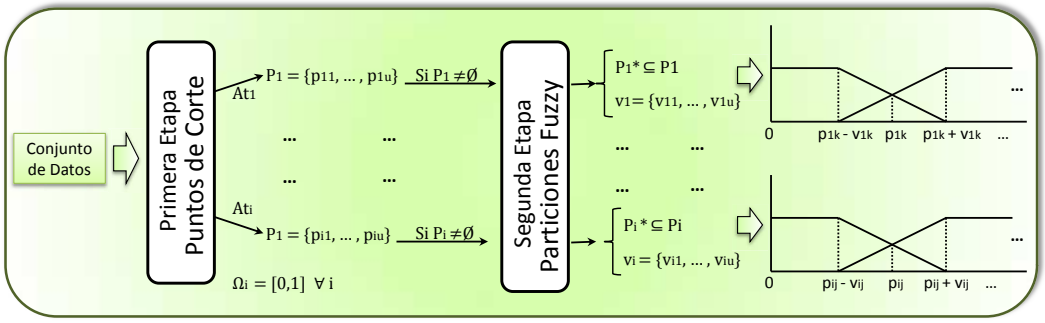


Figura 10.1: Esquema general del algoritmo OFP_CLASS

10.2.1 Buscando los puntos de corte de la partición

En la primera etapa de la técnica de discretización se construye un árbol de decisión fuzzy siguiendo como base el proceso descrito en el Capítulo 6, con dos modificaciones: a) la primera de ellas consiste en añadir una cola de prioridad en el proceso de aprendizaje del árbol y b) la segunda es que el árbol de decisión fuzzy puede tratar atributos que no están discretizados. La discretización de estos atributos es precisamente el objetivo de esta primera etapa.

Respecto a la primera modificación cabe destacar que al introducir la cola de prioridad, los nodos creados serán examinados en función del peso de los ejemplos que contengan, estudiando primero aquellos con mayor número, ya que contienen una mayor cantidad de información. En referencia a la segunda modificación, para tratar con atributos numéricos no discretizados,

utilizamos el proceso básico del algoritmo C4.5. El umbral seleccionado en cada nodo del árbol de decisión para estos atributos será el punto de corte que delimitará los posibles intervalos para llevar a cabo su discretización. De esta forma, el algoritmo que constituye esta primera etapa está basado en un árbol de decisión fuzzy que permite trabajar con atributos nominales, numéricos discretizados por medio de particiones fuzzy, numéricos no discretizados y además permite la existencia de valores missing en todos ellos. En el Algoritmo 14 se describe el proceso completo de una forma esquemática.

Algoritmo 14 Búsqueda de puntos de corte para la técnica OFP_CLASS

BuscandoPuntosCorte(in: E, $Particiones\text{-}Fuzzy$ (opcional); **out:** $Puntos\text{-}Corte$)
begin

1. Q=cola vacía, $E_N = E$, $M_N = A$ (conjunto de atributos), CPS vector de tamaño $|A|$ inicialmente vacio
2. **for** cada ejemplo e_j, $j = 1, \ldots, |E|$, **do**
   $\chi(e_j) = 1$
   end for
3. Crear un nodo N conteniendo todos los ejemplos de E con su peso asociado. Insertarlo en Q.
4. **Repeat**
   4.1. N=Obtener_Primero(Q)
   4.2. **for** cada atributo i **do**
   calcular la ganancia de información G_i^N, $i = 1, \ldots, |M_N|$
   end for
   Elegir el atributo i_{best} cuya G_i^N es máxima
   if i_{best} es un atributo numérico no-discretizado **then**
   el umbral seleccionado es un punto de corte para este atributo: $CPS_i = CPS_i \bigcup \{\text{Punto}-\text{Corte}\}$
   end if
   4.3. Dividir N en H_i nodos hijos de acuerdo al atributo i_{best}
   4.4. **for** cada nodo hijo N_h con $h = 1, \ldots, H_i$ **do**
   Obtener E_{N_h} desde E_N
   if ($|E_{N_h}|$ >mim_num_ejemplos) y (E_{N_h} no tienen la misma clase) **then**
   insertar N_h en Q
   end if
   end for

   until (Q vacía)
5. $Puntos\text{-}Corte$=CPS

end

Detallando el algoritmo nos encontramos que todo comienza en el nodo raíz del árbol donde tiene lugar la inicialización y la cola de prioridad se encuentra vacía. Los ejemplos del nodo raíz tiene un peso inicial igual a 1. La cola de prioridad es ordenada de mayor a menor de acuerdo con el peso total de los ejemplos de los nodos que forman tal cola. De esta forma, los nodos con más peso, son tratados primeros y así garantizamos que el dominio se particione de acuerdo a los atributos más relevantes, ya que los nodos con más peso contienen más información. Tras

inicializar se extrae el primer nodo de la cola de prioridad y se selecciona el mejor atributo para dividir este nodo usando la ganancia de información definida en la ecuación 6.1 como criterio. Se pueden dar dos casos:

Caso 1. La mayor ganancia de información corresponde a un atributo que ya está discretizado, bien porque es nominal o porque ya tiene definida una partición fuzzy (bien mediante un experto o bien mediante otras fuentes de información). En este caso se expande el nodo N con tantos hijos como valores posible tenga el atributo seleccionado. Este comportamiento es similar al descrito en el Capítulo 6 cuando se describe el árbol de decisión.

Caso 2. El atributo con mayor ganancia de información no esta discretizado y por lo tanto es necesario obtener sus puntos de corte. Para ello hay que obtener sus posibles descendientes. Esto se realiza como en un árbol de decisión C4.5, donde los ejemplos son ordenados de acuerdo con el valor del atributo en cuestión y se obtienen los valores intermedios entre los valores e_j y e_{j+1} del atributo una vez ordenados. El valor intermedio obtenido es el que proporciona dos descendientes del nodo y el usado para obtener la ganancia de información. Este proceso es repetido para cada par de valores consecutivos del atributo, con el fin de buscar el valor que proporciona una mayor ganancia de información. Este valor es utilizado para dividir al nodo y es considerado como un posible punto de corte para discretizar al atributo.

Tras seleccionar un atributo en el nodo N, los descendientes generados se introducen en la cola de prioridad de acuerdo al orden establecido y de nuevo se comienza el proceso de extraer el primer nodo de la cola. Este proceso se repite hasta que se llega a la condición de parada de que no quedan nodos en la cola de prioridad. Esto ocurre cuando al generar los nodos descendientes, estos son considerados nodos hojas, por no contener un número mínimo de ejemplos o porque el valor de clase de todos los ejemplos del nodo es igual.

10.2.2 **Construyendo y optimizando las particiones fuzzy**

En esta segunda etapa de la técnica OFP_CLASS, se utiliza un algoritmo genético para obtener los conjuntos fuzzy que definen el particionamiento de los atributos numéricos. Los algoritmos genéticos son muy robustos y tienen mucho poder para tratar infinidad de problemas de diversas áreas, incluida la minería de datos, [50], donde no existe una técnica especializada o directamente aunque existe la técnica, ésta puede ser combinada con un algoritmo genético para mejorar los resultados a través de un algoritmo híbrido, [59].

Esta segunda etapa comienza tomando como entrada los puntos de corte obtenidos en la primera etapa. Si la primera etapa proporciona $F_i - 1$ puntos de corte para el atributo i, como

máximo pueden obtenerse F_i conjuntos fuzzy para dicho atributo (pueden obtenerse menos conjuntos fuzzy debido a que el algoritmo genético selecciona los mejores puntos de corte desactivando los menos relevantes).

Los elementos que componen el algoritmo genético se definen como sigue:

Codificación

Un individuo tiene codificación real y su tamaño es la suma del número de puntos de corte que el árbol de decisión fuzzy proveerá para cada atributo en la primera etapa. Cada gen representa la cantidad a sumar y restar a cada punto de corte de cada atributo que forman la partición fuzzy (la cantidad a restar y sumar a cada punto de corte es la misma). Además cada gen tiene asociado un valor booleano que indica si el gen, punto de corte, debe ser tenido en cuenta o no, en otras palabras si el punto de corte está activo o no. Debemos de considerar que si un gen no está activo, el dominio de los genes adyacentes debe ser recalculado. El dominio de cada gen es definido por un intervalo definido como $[0, min(\frac{p_r - p_{r-1}}{2}, \frac{p_{r+1} - p_r}{2})]$ donde p_r es el r-th punto de corte del atributo i representado por este gen excepto en el primero (p_1) y el último (p_u) punto de corte de cada atributo cuyos dominios son, respectivamente: $[0, min(p_1, \frac{p_2 - p_1}{2}]$ y $[0, min(\frac{p_u - p_{u-1}}{2}, 1 - p_u]$. Cuando $F_i = 2$, el dominio es un único punto de corte definido por $[0, min(p_1, 1 - p_1]$. Hay que tener en cuenta que el dominio de los valores de los atributos numéricos es $[0, 1]$, es decir, están normalizados.

La Figura 10.2 muestra la información de un individuo, que tiene algunos puntos de corte $(p_1, \ldots, p_u)$ activos y otros inactivos en el atributo i.

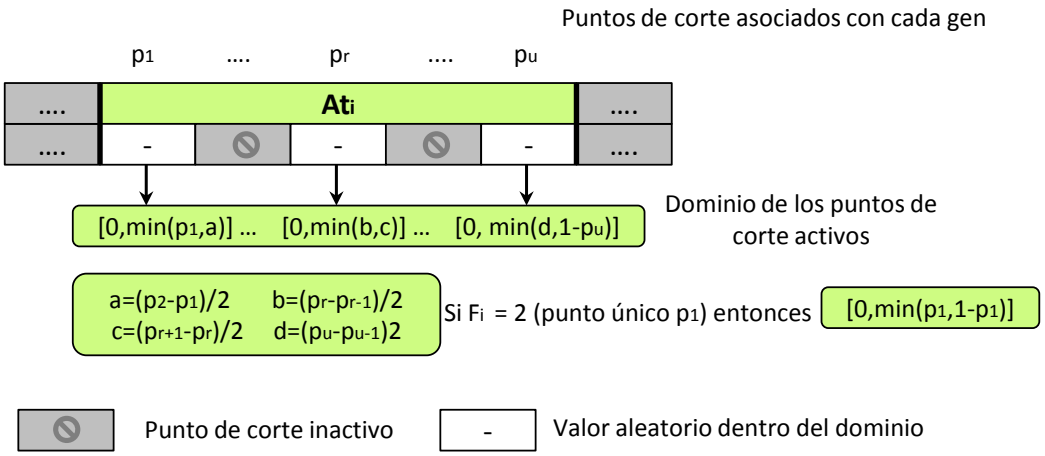


Figura 10.2: Codificación de un individuo

Es importante resaltar que la cantidad a sumar y restar a un punto de corte puede ser cero,

por lo que podría darse que algunas particiones fueran crisp en lugar de fuzzy y la partición para un atributo estuviera compuesto por particiones fuzzy y por intervalos. Además también podríamos considerar la opción de que la cantidad a sumar y restar al punto de corte siempre fuera cero, con lo que al final tendríamos unas particiones crisp optimizadas. Con esta última opción hay que tener en cuenta que las restricciones del resto de operadores referentes a dicha cantidad desaparecían y solamente habría que comprobar que al menos un gen del individuo se encontrase activo.

Inicialización

Se lleva acabo de manera aleatoria teniendo siempre en cuenta el dominio de cada gen. Primero se inicializa para cada gen el valor booleano que indica si el gen está activo o no y una vez que se inicializa el valor booleano de cada gen, se calcula el dominio para cada uno de los genes teniendo en cuenta si los genes adyacentes están o no activos. Tras calcular el dominio de cada gen, aleatoriamente se inicializa el valor de cada gen que indica la cantidad a sumar y restar al punto de corte. Hay que tener en cuenta que al menos debe de existir un gen activo para cada individuo, ya que si ningún gen estuviera activo significaría que no se discretizará ningún atributo, y el atributo quedaría particionado en un único intervalo que cubriría todo el dominio.

Función Fitness

La función fitness de cada individuo es definida acorde a la ganancia de información definida en [9]. Esta función de fitness indica cómo de dependiente son los atributos con respecto a la clase, en otras palabras, cómo de discriminatorias son las particiones de cada atributo. Si el valor de fitness para cada individuo es cercano a cero, esto indica que los atributos son totalmente independientes de las clases, lo cual significa que las particiones obtenidas no discriminan la clase. Por el contrario, cuando más alejado sea este valor de cero, las particiones obtenidas serán más aceptables y discriminarán más las clases proporcionando un buena precisión en clasificación. El Algoritmo 15 muestra en detalle el cálculo de esta función donde:

- μ_{if} es el grado de pertenencia correspondiente al conjunto fuzzy f del atributo i.
- E_k es el subconjunto de ejemplos de E que pertenecen a la clase k.

Selección

El proceso de selección se lleva a cabo mediante torneo, tomando subconjuntos de tamaño dos con reemplazamiento. Se seleccionan aleatoriamente dos individuos de la población y se compara el valor de fitness entre ellos, se selecciona aquel individuo con mayor fitness. En el caso de tener igual valor de fitness, aleatoriamente se selecciona uno de ellos para pasar a la población seleccionada y a la cual se le aplicarán los operadores de cruce y mutación.

Algoritmo 15 Función Fitness para la técnica OFP_CLASS

Fitness(in: E, **out:** $ValorFitness$)

begin

1. **for** cada atributo, $i = 1, \ldots, |A|$, **do**

   1.1 **for** cada conjunto, $f = 1, \ldots, F_i$, del atributo i **do**
   for cada clase $k = 1, \ldots, |C|$
   calcular la probabilidad $P_{ifk} = \frac{\Sigma_{e \epsilon E_k} \mu_{if}(e)}{\Sigma_{e \epsilon E} \mu_{if}(e)}$
   end for
   end for

   1.2 **for** cada clase $k = 1, \ldots, |C|$ **do**
   calcular la probabilidad $P_{ik} = \Sigma_{f=1}^{F_i} P_{ifk}$
   end for

   1.3 **for** cada $f = 1, \ldots, F_i$ **do**
   calcular la probabilidad $P_{if} = \Sigma_{k=1}^{|C|} P_{ifk}$
   end for

   1.4 **for** cada $f = 1, \ldots, F_i$ **do**
   calcular la ganancia de información del atributo i y conjunto f: $I_{if} = \Sigma_{k=1}^{|C|} P_{ifk} \cdot \log_2 \frac{P_{ifk}}{P_{ik} \cdot P_{if}}$
   end for

   1.5 **for** cada $f = 1, \ldots, F_i$ **do**
   calcular la entropía $H_{if} = -\Sigma_{k=1}^{|C|} P_{ifk} \cdot \log_2 P_{ifk}$
   end for

   1.6 Calcular la I y H total del atributo i, $I_i = \sum_{f=1}^{F_i} I_{if} y H_i = \sum_{f=1}^{F_i} H_{if}$

   end for

2. Calcular el fitness como $ValueFitness = \frac{\sum_{i=1}^{|A|} I_i}{\sum_{i=1}^{|A|} H_i}$

end

Cruce

El operador de cruce es por un único punto y se aplica con cierta probabilidad. No todas las operaciones de cruce son válidas, ya que debemos de tener en cuenta las restricciones de que en el cruce, los individuos descendientes no tengan todos sus genes inactivos, ya que si se diera este caso el cruce no sería válido. Otro aspecto a tener en cuenta es que al crear los dos individuos nuevos en el cruce es posible que existan conflictos con los dominios de los genes adyacentes si el punto de corte se encuentra entre los puntos de corte de un mismo atributo.En este caso los dominios deben de ser actualizados y en caso de que el valor que indica la cantidad a sumar y restar al punto de corte se encuentre fuera de tal dominio, éste es actualizado al valor máximo del nuevo dominio.

Mutación

La mutación consiste en una mutación híbrida que según cierta probabilidad es aplicada

una u otra. La mutación se lleva a cabo gen a gen y cuando a un gen le toca mutarse, según la probabilidad puede:

a) Modificar el valor booleano, activando el gen si este está inactivo o viceversa. En este caso hay que tener en cuenta que si el gen es el único activo en todo el individuo y la mutación indica que hay que desactivarlo, esta mutación no se llevaría a cabo. Por otro lado, si el gen está desactivado y la mutación lo activa, habrá que recalcular el dominio para ese gen y los adyacentes y actualizar los valores en caso de sobrepasar los límites del nuevo dominio, al valor máximo del mismo.
b) Modificar el valor que indica la cantidad a sumar y restar al punto de corte. En este caso solamente se modifica el valor actual por otro valor aleatoriamente dentro del dominio del gen.

Condición de Parada

EL AG para cuando se alcanzan un determinado número de generaciones.

Al final el algoritmo genético devuelve el individuo con mejor fitness el cual indica los conjuntos fuzzy para cada atributo i.

Para mostrar cómo trabajan los operadores genéticos vamos a describir un ejemplo.

Supongamos un conjunto de datos que solamente consta de tres atributos, para los cuales el Algoritmo 14 ha obtenido 2, 3 y 1 puntos de corte para cada atributo respectivamente (Tabla 10.1).

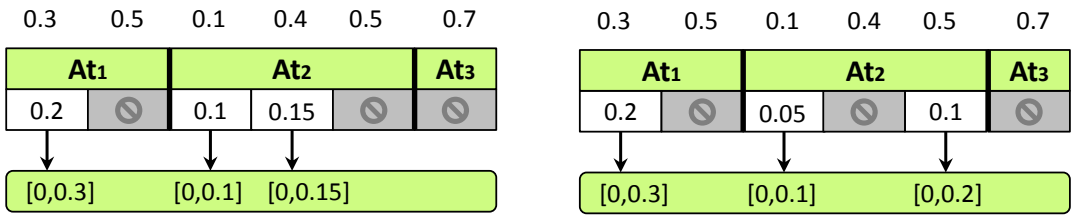
Tabla 10.1: Salida de la etapa 1 de la técnica OFP_CLASS

Atributo 1	0.3	0.5	Atributo 2	0.1	0.4	0.5	Atributo 3	0.7

Basándose en los puntos de corte, en la segunda etapa, el algoritmo genético determina cuáles de ellos formarán la partición fuzzy de cada atributo. Así, la codificación de dos posibles individuos se muestra en la Figura 10.3. En cada individuo se muestra la información indicada en la Figura 10.2.

La Figura 10.4 muestra un posible cruce válido por el punto A. Como se puede observar en la figura, la parte izquierda del hijo debe adaptar sus dominios (genes 2 y 3 del atributo 2) y además adaptar los valores de ambos genes.

La Figura 10.5 muestra un ejemplo de una mutación válida. Como podemos observar el gen 2 del atributo 1 es un gen inactivo antes de la mutación por lo que si este gen es mutado, se



a) Individuo 1 b) Individuo 2

Figura 10.3: Codificación de los individuos

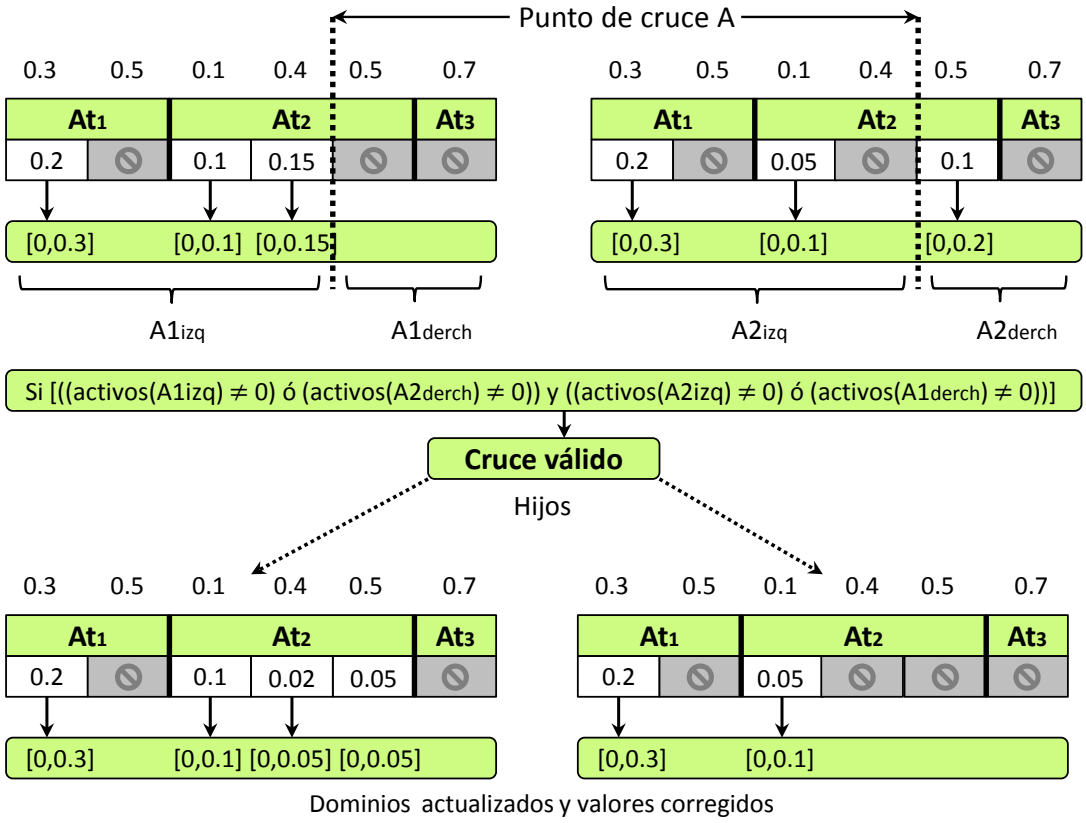


Figura 10.4: Ejemplo de un cruce válido

convierte en un gen activo y hay que recalcular los dominios de los genes adyacentes y ajustar los valores pertinentes.

Si asumimos que el individuo no cambiará después de la mutación mostrada en la Figura 10.5 y se produce la condición de parada, solamente quedarían discretizados los atributos At_1 y At_2. La discretización fuzzy de estos dos atributos quedaría como se muestra en la Figura 10.6.

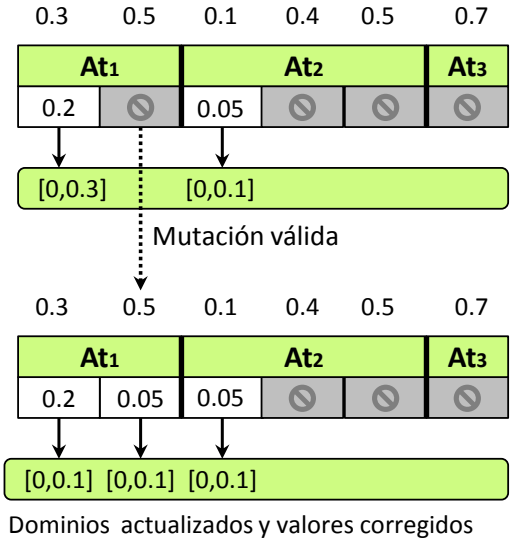
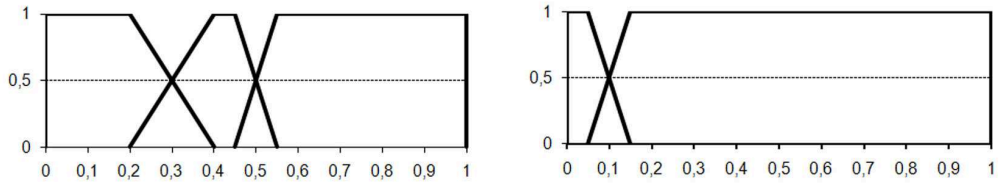


Figura 10.5: Ejemplo de mutación permitida



a) Discretización del atributo At_1 b) Discretización del atributo At_2

Figura 10.6: Particiones Fuzzy obtenidas

10.3 OFP_CLASS$_{LQD}$: Extendiendo OFP_CLASS para trabajar con datos de baja calidad

En la sección anterior hemos descrito la técnica de particionamiento OFP_CLASS que discretiza los atributos continuos mediante conjuntos fuzzy. Como ya hemos comentado, esta técnica permite trabajar con atributos nominales, numéricos discretizados por medio de particiones fuzzy, numéricos no discretizados y además permite la existencia de valores missing en todos ellos. Sin embargo, y como hemos comentado en el Capítulo 3, los datos LQD aparecen en muchos dominios y situaciones reales. Por esta razón extendemos la técnica OFP_CLASS para que pueda trabajar con LQD. La extensión de la técnica, que denominaremos OFP_CLASS$_{LQD}$, va a incorporar el manejo de valores de baja calidad (valores intervalares y fuzzy) en atributos numéricos o continuos y valores imprecisos en el atributo nominal de clase, [30, 37]. La técnica OFP_CLASS$_{LQD}$ sigue la misma filosofía que la técnica OFP_CLASS, es decir, está compuesta por las mismas dos etapas con sus algoritmos correspondientes. Estos algoritmos serán los

mismos incluyendo las modificaciones oportunas para hacer posible el manejo de datos LQD.

Para extender el algoritmo **BusquedaPuntosCorte**, en la primera etapa debemos añadir modificaciones en el árbol de decisión fuzzy que sirve como base para la búsqueda de los puntos de corte. La principal modificación en el árbol es en el punto donde se selecciona el mejor atributo para dividir un nodo usando la ganancia de información, puesto que en este caso debemos de establecer un orden de los valores los cuales ahora pueden contener valores fuzzy e intervalares en los atributos numéricos. Lo mismo sucede en la segunda etapa cuando la función fitness utiliza la ganancia de información para calcular la bondad de un individuo.

Veamos en detalle dichas modificaciones.

10.3.1 Extendiendo la búsqueda de puntos de corte desde LQD

Al igual que para el algoritmo OFP_CLASS, en esta primera etapa se utiliza el árbol de decisión fuzzy expuesto en el Capítulo 6 con algunas modificaciones. En este caso algunas de las modificaciones las hemos comentado en las subsecciones anteriores donde se ha detallado como trabaja el árbol al encontrarse con uno de los nuevos tipos de datos LQD. En este caso, la extensión del árbol de decisión fuzzy permite que los atributos continuos puedan contener valores fuzzy e intervalares. Las modificaciones que realizamos en el Algoritmo 14 se muestran en el Algoritmo 16.

Algoritmo 16 Extensión de la búsqueda de puntos de corte al manejo de LQD

BuscandoPuntosCorte$_{LQD}$(**in:** E, $Particiones\text{-}Fuzzy$ (opcional); **out:** $Puntos\text{-}Corte$)
begin

.....

2. **for** cada ejemplo e_j con clase única **do**
   $\chi_{root}(e_j) = 1$
   end for
   for cada ejemplo e_j con clase imprecisa **do**
   Replicar cada ejemplo j tantas veces como valores de clase tenga.
   Inicializar sus valores $\chi_{root}(e_j)$ de acuerdo a la información disponible sobre los valores de clase.
   end for
   .....
   .....
   4.2. .....
   calcular la ganancia de información $G_i^N, i = 1, \ldots, |M_N|$
   .....
   4.3. Dividir N en H_i nodos hijos de acuerdo al atributo i_{best}
   .....

end

La diferencia estriba en los siguientes elementos: En el paso 3, todos los ejemplos en el nodo raíz tienen un peso inicial igual a 1, excepto los ejemplos que tienen el atributo clase

impreciso cuyo peso es inicializado según la información disponible.

La cola de prioridad es ordenada de mayor a menor teniendo en cuenta el peso total de los ejemplos de los nodos que la forman. Así garantizamos que el dominio de la partición sea creado respecto a los atributos más relevantes.

En el caso del paso 3, cuando se expande un nodo de acuerdo a un atributo, se pueden dar dos situaciones:

1. Si el atributo ya está discretizado, el nodo es expandido en tantos nodos hijos como posibles valores tiene el atributo seleccionado. En este caso el comportamiento del árbol es similar al descrito en el Capítulo 6.

2. Si el atributo no está discretizado previamente, debemos de obtener sus posibles descendientes. Para realizar esto, como en un árbol C4.5, los ejemplos son ordenados de acuerdo al valor del atributo en cuestión. Así que debemos de ordenar los datos con valores crisp, fuzzy e intervalos y para esto utilizamos un índice de ordenación, [197].

   Todo lo demás en esta situación es igual que lo comentado en la técnica OFP_CLASS añadiendo que cuando los descendientes son generados y un ejemplo e desciende por ambos descendientes, el proceso que se lleva a cabo es el mismo que para el árbol de decisión fuzzy expuesto en el Capítulo 6. Si el valor del atributo es fuzzy o intervalar entonces aplicamos la función $\mu_{simil}(\cdot))$ para determinar el grado de pertenencia de este ejemplo E a los nodos descendientes. Es importante tener en cuenta que el valor representativo para los valores fuzzy e intervalares solamente se utiliza para ordenar los valores y poder obtener los puntos de cortes, pero cuando es necesario utilizar estos valores para hacer alguna estimación, se utiliza el valor original y no el valor representativo.

10.3.2 Construyendo y optimizando las particiones fuzzy desde LQD

En esta segunda etapa del algoritmo extendido OFP_CLASS$_{\text{LQD}}$ se utiliza, al igual que en la técnica OFP_CLASS, un algoritmo genético para obtener los conjuntos fuzzy que discretizarán los atributos numéricos no discretizados. Las características y los elementos que componen este algoritmo son muy similares y en algunos casos exactamente iguales a los elementos que componen el algoritmo genético de la técnica OFP_CLASS. Para evitar repetir la explicación de los elementos solamente vamos a hacer referencia a las diferencias entre el algoritmo genético de OFP_CLASS$_{\text{LQD}}$ y el algoritmo genético de OFP_CLASS.

La codificación de un individuo y los operadores de selección cruce y mutación, funcionan de la misma forma en el algoritmo genético de OFP_CLASS$_{\text{LQD}}$ que en el algoritmo genético de OFP_CLASS. Ya que en estos operadores se trabaja directamente con el individuo, y no se

trata directamente con los valores de los atributos, es por eso que el comportamiento es similar. Puesto que hay que recordar que OFP_CLASS$_{\text{LQD}}$ es una extensión de OFP_CLASS para poder trabajar con atributos que contengan información de baja calidad.

La única función que muestra diferencias entre los dos algoritmos es la función de fitness que sí trabaja directamente con los valores de los atributos, para obtener su valor. Concretamente cuando se calcula el valor de μ_{if}, que es la función de pertenencia correspondiente al conjunto fuzzy f del atributo i. El valor μ_{if} depende del tipo del atributo i. Si el tipo es fuzzy o intervalar la función de pertenencia se calcula acorde a la función $\mu_{simil}(\cdot)$, sino el cálculo se realiza tal y como se ha indicado al explicar la técnica OFP_CLASS.

Algoritmo 17 Función Fitness extendida al manejo de LQD

Fitness$_{LQD}$(**in:** E, **out:** $ValorFitness$)
begin

.....

1.1 **for** cada conjunto, $f = 1, \ldots, F_i$, del atributo i **do**
for cada clase $k = 1, \ldots, |C|$
calcular la probabilidad $P_{ifk} = \frac{\Sigma_{e \epsilon E_k} \mu_{if}(e)}{\Sigma_{e \epsilon E} \mu_{if}(e)}$
end for
end for

.....

end

10.4 BAGOFP_CLASS: Usando bagging en la discretización de atributos numéricos

En esta sección vamos a presentar una mejora para las técnicas de discretización que hemos presentado en las secciones anteriores. En este caso la nueva propuesta se denomina BAGOFP_CLASS, [38], [46], [40], y al igual que sus dos técnicas predecesoras también se compone de dos etapas, estando la primera etapa formada por un árbol de decisión fuzzy y la segunda etapa la forma un algoritmo genético. En este caso el propósito es mejorar el algoritmo de discretización para cuando nos enfrentamos a conjuntos de datos que tienen pocos ejemplos con respecto al número de atributos. Establecer una relación exacta entre la probabilidad de los errores de clasificación, el número de ejemplos de entrenamiento, el número de atributos y los verdaderos parámetros de las densidades condicionales para la clase es muy difícil, pero se han sugerido algunas directrices. Como regla general, se requiere un número mínimo de $10 \cdot |A| \cdot |C|$ ejemplos de entrenamiento para un problema de clasificación $|A|$-dimensional de $|C|$ clases, [102]. A los conjuntos de datos que no cumplan con esta regla general, las llamaremos a partir

de ahora conjuntos de datos de tamaño pequeño. Para poder trabajar con conjuntos de datos de tamaño pequeño, proponemos añadir al algoritmo de discretización la técnica bagging en ambas etapas.

La técnica bagging, [24], consiste en producir k conjuntos de entrenamiento con tamaño igual al conjunto de entrenamiento inicial, mediante la selección aleatoria de ejemplos con reemplazamiento del conjunto original. El uso de bagging en OFP_CLASS, y su extensión, está justificado por el hecho de que bagging puede proporcionar ganancias substanciales en precisión, cuando la técnica de clasificación utilizada es inestable (si al perturbar el conjunto de aprendizaje puede causar cambios significativos en el clasificador construido, con bagging se puede mejorar su precisión, [24]). Además la inestabilidad fue estudiada en [25] donde se señala que las redes neuronales y los árboles de clasificación y regresión (entre otras) son técnicas que pueden ser inestables. Los resultados de los experimentos tanto teóricos como experimentales, muestran que bagging puede proporcionar a un procedimiento que es bueno pero inestable a la vez, un paso significativo hacia la optimalidad. Resumidamente las ventajas generales que ofrece la técnica bagging son: 1) mayor tolerancia al ruido ya que una combinación múltiple de clasificadores produce un rendimiento superior que un clasificador único; 2) la distribución de clases se mantiene con respecto al conjunto de datos original y 3) además aporta cierta estabilidad cuando se complementa con técnicas de clasificación que pueden ser inestables en ciertas situaciones, [24].

Centrándonos en los árboles de decisión, ya que es la técnica utilizada en la primera etapa del algoritmo, la inestabilidad de esta técnica puede verse incrementada por el uso de conjuntos de datos de tamaño pequeño. Cuando el número de ejemplos es pequeño comparado con el número de atributos, la probabilidad de que existan atributos redundantes se incrementa. De esta forma, pueden haber varios atributos con la misma ganancia de información y solamente uno de ellos será seleccionado. Con cierta probabilidad, los atributos no seleccionados, no serán particionados y no formarán parte de la partición generada por el árbol de decisión. Así, cuando se introduce el proceso de bagging se incrementa la probabilidad de que estos atributos formen parte de la partición final.

Por lo tanto, con esta primera fase vamos a mejorar el particionamiento al habilitar un mayor número de atributos que formarán parte de la partición cuando la ganancia de información de varios atributos sea la misma. Será la segunda etapa del algoritmo la que determine de todos los atributos los más relevantes para la clasificación y por lo tanto para ser discretizados.

Por otra parte, si analizamos la segunda etapa, nos podemos centrar en la función de fitness que es la más susceptible a los cambios en la diferentes versiones, ya que es la que trabaja directamente con los valores de los atributos, tratando de encontrar las mejores particiones para dividir los ejemplos con respecto al atributo clase. Por ello, vamos a introducir el proceso

de bagging en esta segunda fase para intentar reducir la variabilidad en los resultados de las particiones. Modificamos por tanto la función de fitness para obtener un mejor valor de estas particiones (intentando mejorar los individuos) con el rendimiento medio utilizando bagging. Además con bagging el algoritmo genético puede estudiar diferentes regiones del espacio de búsqueda y puede mejorar la visión global de las particiones fuzzy.

En resumen, vamos a introducir el proceso de bagging en la técnica BAGOFP_CLASS en: 1) la fase de construcción del árbol de decisión, para generar los diferentes puntos de corte, 2) en la función de fitness que se utiliza en el algoritmo genético de la segunda fase. Además, de forma resumida, las ventajas que nos aporta el proceso de bagging en esta técnica de discretización son:

- Añadir estabilidad al árbol de decisión, ya que los árboles suelen presentar inestabilidad cuando trabajan con conjuntos de datos de tamaño pequeño, [25].
- Durante la construcción del árbol puede producirse la situación de que en un nodo varios atributos tengan la misma ganancia de información y si no se aplica bagging solamente uno de ellos sería elegido. Por el contrario al aplicarse bagging estos atributos podrán ser seleccionados en una de las repeticiones.
- Con cierta probabilidad, los atributos no seleccionados no son particionados, por lo que estos atributos podrían no formar parte de las particiones finales generadas por el árbol de decisión. Al aplicarse bagging una mayor cantidad de atributos son discretizados y es el algoritmo genético el encargado de decidir cuáles de ellos son más importantes.
- Por último, aplicando bagging en la función fitness del algoritmo genético, reducimos la variabilidad de este valor, obteniendo una medida más robusta y fiable para cada individuo.

Antes de detallar las características de las dos etapas que componen la técnica BAGOFP_CLASS, en la Figura 10.7 se muestra el proceso de las dos etapas que sigue este algoritmo para llevar a cabo la discretización de los atributos numéricos. En las siguientes secciones vamos a presentar las novedades introducidas en estas etapas que diferencian a la técnica BAGOFP_CLASS de las dos presentadas anteriormente.

10.4.1 **Bagging en la búsqueda de puntos de corte**

La primera fase de BAGOFP_CLASS utiliza un árbol de decisión fuzzy como técnica base para la búsqueda de los posibles puntos de corte para construir las particiones, pero el conjunto de datos de entrada a este árbol de decisión fuzzy no es el conjunto de datos original sino que es

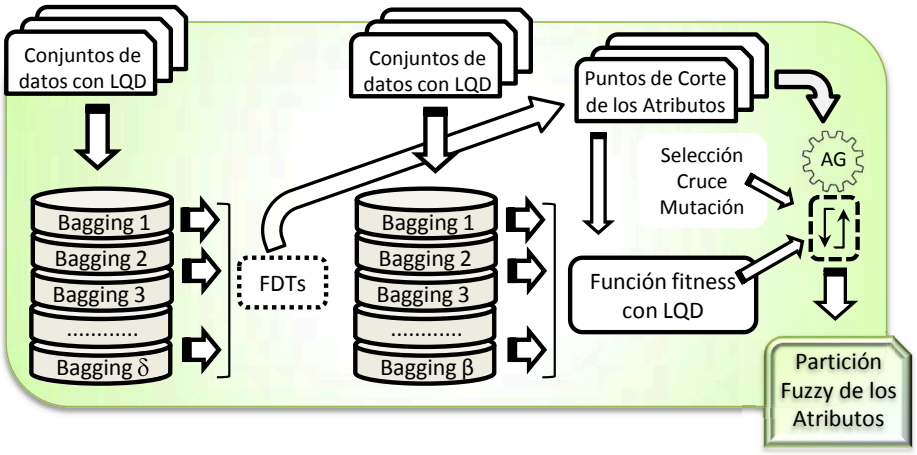


Figura 10.7: Técnica BAGOFP_CLASS

cada uno de los conjuntos de datos obtenidos mediante el proceso de bagging que se lleva a cabo sobre el conjunto de datos original.

Los tipos de valores con los cuales este árbol de decisión puede trabajar son valores nominales, numéricos discretizados mediante una partición fuzzy, numéricos no discretizados con valores crisp, fuzzy e intervalares, permitiendo además la existencia de valores missing en cada uno de ellos, en definitiva, son los mismos tipos de valores que permite el algoritmo OFP_CLASS$_{\text{LQD}}$ presentado en la sección anterior. El comportamiento del árbol depende del tipo de atributo que es analizado con el fin de obtener el mejor atributo para dividir un nodo. Por un lado, cuando el atributo es numérico y no tiene particiones disponibles, el proceso seguido es similar al proceso en un árbol de decisión C4.5. El umbral seleccionado en cada nodo del árbol para estos atributos serán los puntos de corte que delimiten los intervalos. Por otra parte, para los otros tipos de atributos, la metodología seguida es la misma descrita para el árbol de decisión fuzzy presentado en el Capítulo 6. Durante el resto del proceso, el comportamiento de la técnica es similar tanto para atributos discretizados como no discretizados. Pero hay que tener en cuenta que los nodos del árbol no se estudian en orden de aparición, sino que se analizan primero aquellos nodos donde la suma del peso de los ejemplos que contienen es mayor. Ya que conforme los nodos se van creando se introducen en una cola de prioridad ordenados por la suma de los pesos de los ejemplos.

Si comparamos la técnica que presentamos en está sección con las técnicas que hemos presentado en la secciones anteriores, la principal diferencia entre ellas es para los conjuntos de datos que no tienen $10 \cdot |A| \cdot |C|$ ejemplos de entrenamiento como mínimo. Para estos conjuntos de datos los puntos de corte obtenidos por las técnicas anteriores no son lo suficientemente ricos en información para crear buenas particiones porque estos puntos de corte son demasiados

específicos y las particiones obtenidas no son globales. Para evitar obtener puntos de corte tan específicos y además conseguir que más atributos sean discretizados, la entrada al árbol de decisión fuzzy en la técnica BAGOFP_CLASS es el bagging obtenido del conjunto de datos original. Es importante recordar que durante el proceso de bagging pueden aparecer puntos de cortes repetidos, pero estos solamente serán incluidos una vez. El Algoritmo 18 describe todo el proceso para obtener todos los puntos de corte posibles.

Algoritmo 18 Obteniendo puntos de corte

BuscandoPuntosCorte$_{BAG}$(**in:** E, $Particion\text{-}Fuzzy$ (opcional); **out:** $Puntos\text{-}Corte$)
begin

1. Inicializar el conjunto de puntos de corte de los atributos numéricos: $CPS = \emptyset$
2. Inicializar el tamaño del bagging δ.
3. **for** j=1 hasta δ **do**
   - 3.1 Obtener el conjunto de datos BA_j que consiste en $|E|$ ejemplos seleccionados desde E aleatoriamente y con remplazamiento.
   - 3.2 CPS=$CPS \bigcup$ **BuscandoPuntosCorte**$_{LQD}$(BA_j,$Particion\text{-}Fuzzy$).
4. **end for**
5. $Puntos\text{-}corte$=CPS

end

Todos los puntos de corte obtenidos son introducidos, como entrada, en la segunda etapa para poder construir y optimizar las particiones fuzzy.

10.4.2 Bagging en la construcción y optimización de las particiones fuzzy

En esta segunda etapa de la técnica se utiliza igual que en las versiones anteriores un algoritmo genético para obtener los conjuntos fuzzy que constituyen las particiones de los atributos no discretizados. Recordemos que el algoritmo genético toma como entrada los puntos de corte obtenidos en la primera etapa y es este algoritmo quien decide qué puntos de corte son importantes y cuáles no para terminar formando las particiones finales. La modificación llevada a cabo para mejorar esta etapa es la introducción de un proceso de bagging en la función de fitness, por lo tanto el resto de elementos del algoritmo genético, como la codificación, la inicialización, la selección, el cruce y la mutación no se modifican y trabajan de la misma forma que lo hacen OFP_CLASS y su extensión. Por lo tanto vamos a detallar solamente el funcionamiento de la función de fitness con el fin de evitar repetir la explicación del resto de elementos cuyo funcionamiento no se modifica.

10.4.2.1 **Función de Fitness**

La función de fitness para cada individuo es definida de acuerdo a la ganancia de información definida en [9]. En este caso, de la misma forma que en la primera etapa, para calcular el fitness de cada individuo aplicamos un proceso de bagging. Así, el valor de fitness de cada individuo es calculado como la media de los valores de fitness obtenidos tras aplicar los diferentes conjuntos de datos obtenidos con el proceso de bagging. De esta forma el algoritmo obtiene un valor de fitness más robusto y menos variable para cada individuo porque al utilizar el valor medio, se obtiene una visión general sobre los diferentes subconjuntos y al mismo tiempo una medida más fiable. El Algoritmo 19 describe la función de fitness.

En el Algoritmo 19, $\mu_{if}(\cdot)$ es la función de pertenencia correspondiente al conjunto fuzzy f del atributo i. La función de fitness está basada en la ganancia de información e indica cómo de dependientes son los atributos con respecto a la clase, es decir, cómo de discriminatorias son las particiones de cada atributo.

Algoritmo 19 Función Fitness para la técnica BAGOFP_CLASS

Fitness$_{BAG}$(**in:** E, **out:** $FitnessFinal$)
begin

1. Inicializar el tamaño de bagging β
2. Inicializar $ValorFitness = 0$
3. **for** j=1 to β **do**
   - 3.1 Obtener el conjunto de datos BA_j que consiste en $|E|$ ejemplos seleccionados de E aleatoriamente y con remplazamiento.
   - 3.2 $ValorFitness_j$=**Fitness**$_{LQD}(BA_j)$
   - 3.3 Calcular el fitness como: $ValorFitness = ValorFitness + ValorFitness_j$

   end for
4. $FitnessFinal = ValorFitness/\beta$
5. Devolver $FitnessFinal$

end

10.5 Resultados experimentales

En esta sección vamos a presentar tres experimentos: en el primero de ellos trataremos de mostrar el rendimiento de la técnica OFP_CLASS ante una serie de conjuntos de datos comparándolo con distintas técnicas de la literatura; en el segundo experimento mostramos el comportamiento de la extensión OFP_CLASS$_{\text{LQD}}$ ante conjuntos de datos que contienen datos LQD; por último, en el tercer experimento mostramos el comportamiento de BAGOFP_CLASS fren-

te a las técnicas OFP_CLASS y $\text{OFP\_CLASS}_{\text{LQD}}$ y los conjuntos de datos tratados en los dos experimentos anteriores.

10.5.1 Experimentos con OFP_CLASS

En esta sección vamos a mostrar varios resultados experimentales que muestran el comportamiento de la técnica de discretización OFP_CLASS.

10.5.1.1 Marco Experimental

Para analizar el rendimiento de la técnica propuesta OFP_CLASS, hemos utilizado 14 conjuntos de datos disponibles en el repositorio de aprendizaje automático UCI, [77]. Estos conjuntos de datos han sido también utilizados en los siguientes trabajos [58, 125, 126, 154]. Por lo tanto vamos a comparar los resultados obtenidos con la técnica OFP_CLASS, con los obtenidos por las técnicas presentadas en [58, 125, 126, 154]. Estas técnicas reflejan diferentes tipos de discretización fuzzy. Todos los atributos numéricos de estos conjuntos de datos están inicialmente sin discretizar y las técnicas deben de obtener la discretización para todos ellos.

Técnicas utilizados y experimentos realizados

Las técnicas que utilizaremos son las siguientes:

- **FSGA**: En [58], se desarrolla un algoritmo genético que optimiza los parámetros para un particionamiento fuzzy con intervalos adaptativos y además que elimina los atributos irrelevantes o ruidosos, para reducir la cantidad de datos transformados. Las particiones obtenidas son representadas por el solapamiento de intervalos. Los valores del atributo son transformados en vectores binarios. Cada vector deberá tener como máximo dos unos, indicando que el valor del atributo del ejemplo pertenece a una o a dos particiones, (el hecho de pertenecer a dos elementos de la partición indica la incertidumbre que tiene asociada). Una vez obtenidas las particiones de los atributos, los ejemplos son transformados en vectores de ceros y unos. La función de fitness es una red neuronal que aprende y clasifica un subconjunto de ejemplos transformados del conjunto de datos. Para la evaluación utilizan dos algoritmos de aprendizaje automático una red neuronal backpropagation (ANN) y un árbol de decisión C4.5. Estos algoritmos serán denotados con FSGA_{ANN} y $\text{FSGA}_{\text{C4.5}}$ respectivamente. Al igual que el algoritmo OFP_CLASS, esta técnica crea un fichero con las particiones de los atributos continuos, por lo que estas particiones pueden ser utilizadas a posteriori sin la necesidad de ejecutar la técnica cada vez que se necesiten.

- **NFDT**: En [154], se introduce una técnica para la inducción de los árboles de decisión fuzzy que determina la ubicación y la incertidumbre asociada para cada frontera de decisión durante el proceso de construcción del árbol. En este proceso se genera una partición fuzzy de los atributos que proporciona una estimación de confianza de la clasificación a través de la propagación de la función de pertenencia. La superposición de los elementos de la partición es definida por una distribución normal con su media y desviación estándar. Estas distribuciones son obtenidas utilizado los mejores puntos de corte de los subconjuntos del conjunto de datos en el nodo. Las particiones de los atributos son específicas para el clasificador o el algoritmo que los genera. Estas particiones no pueden ser utilizadas por otros clasificadores.
- **FMVR**: En [125], se construye particiones fuzzy por medio de la combinación de algoritmos de clustering fuzzy (FCM) mediante la regla del voto mayoritario. Un algoritmo de emparejamiento de clases basado en k-vecinos más cercanos proporciona la correspondencia entre las clases de los componentes de la partición fuzzy. Las particiones fuzzy resultantes son obtenidas sobre la base de las particiones generadas por cada uno de los algoritmos FCM, llevando a cabo un consenso utilizando la regla del voto mayoritario fuzzy. Las particiones de los atributos son específicos del clasificador o algoritmo que las genera.
- **FWMVR**: En [126], se construye una técnica similar a la anterior, donde se utilizan los mismos algoritmos. Sin embargo el método de consenso es llevado a cabo utilizando el voto mayoritario ponderado.
- **FID**: En [104], se propone FID 3.4. FID 3.4 construye un árbol de decisión fuzzy. Además, en este trabajo se define el procedimiento para generar particiones desde los datos para los atributos numéricos usando el mismo árbol de decisión que construye. El procedimiento crea un nuevo fichero con los atributos particionados.

Con estas técnicas y la técnica propuesta OFP_CLASS realizamos dos conjuntos de experimentos. En el primero comparamos los resultados obtenidos por todos las técnicas utilizando como evaluador de las particiones las técnicas propuestas por los respectivos autores. Para evaluar las diferentes particiones obtenidas por OFP_CLASS utilizamos el árbol de decisión fuzzy que hemos presentado en el Capítulo 6. A partir de ahora, denotaremos por $\text{OFP\_CLASS}_{\text{FDT}_{\text{LQD}}}$ al árbol de decisión fuzzy que utiliza las particiones creadas por la técnica OFP_CLASS. El segundo experimento se lleva a cabo para comparar las particiones obtenidas por las distintas técnicas pero utilizando un mismo evaluador. El evaluador que vamos a utilizar es el árbol de decisión fuzzy propuesto por Janikov en [104].

Conjuntos de datos y parámetros usados en las técnicas

Para obtener estos resultados hemos utilizado varios conjuntos de datos del repositorio de la UCI, [77], cuyas características se muestran en la Tabla 10.2. En esta tabla se muestra el número de ejemplos ($|E|$), el número de atributos ($|A|$), el número de atributos numéricos (Cont.) y el número de clases ($|C|$) para cada conjunto de datos. "Abbr" indica la abreviatura del conjunto de datos utilizada en los experimentos. En la Tabla 10.2, el conjunto de datos SON* toma como base el conjunto de datos SON, pero tiene seleccionados los 19 atributos más relevantes por medio de la herramienta Weka, [91].

Tabla 10.2: Descripción de los conjuntos de datos

Conjunto de datos	**Abbr**	$\|E\|$	$\|A\|$	**Cont.**	C	**Conjunto de datos**	**Abbr**	$\|E\|$	$\|A\|$	**Cont.**	C
Australian credit	AUS	690	14	6	2	Sonar*	SON*	208	19	19	2
German Credit	GER	1000	24	24	2	SPECTF heart	SPE	267	44	44	2
Glass	GLA	214	9	9	6	Thyroid Disease	THY	215	5	5	3
Ionosphere	ION	351	34	34	2	Wisconsin Breast C.	WDBC	569	30	30	2
Iris Plants	IRP	150	4	4	3	Waveform	WDG	5000	21	21	3
Pima Indian Diabet.	PIM	768	8	8	2	Wine recognition	WIN	178	13	13	3
Sonar	SON	208	60	60	2	Zoo	ZOO	101	16	1	7

Los parámetros de las técnicas se presentan en la Tabla 10.3. Todos los parámetros para las técnicas son los más recomendados por los respectivos autores. Para desarrollar los diferentes experimentos consideramos un 3×5-fold cross-validation.

Tabla 10.3: Parámetros usados en las técnicas

Técnica	**Parámetros**
FSGA	tamaño-población=500, generaciones=100.000, parada=5000 fallos consecutivos en el reemplazamiento de uno de los padres, cruce multi-punto=5, p_c=0.3, p_m=0.1, evaluación-fitness=ANN con 80% del conjunto de datos
NFDT	criterio-división=ganancia de información, tamaño-subconjuntos= 50, criterio-parada=mínimo_número_ejemplos y nodo_puro
FMVR	ϵ (convergencia) =0.0001; m (fuzzifier)=2; número de iteraciones=100
FWMVR	ϵ (convergencia) =0.0001; m (fuzzifier)=2; número de iteraciones=100
FID	mínimo_número_de_ejemplos=2, discretización top-down, ganancia_mínima=0.1
OFP_CLASS	tamaño-población=100, generaciones=200, p_c=0.3, p_m=0.1, torneo=2

El índice de ordenación utilizado para ordenar los datos con valores crisp, fuzzy e intervalos se calcula como se indica en 10.1.

Sea B_i un conjunto fuzzy o un intervalo en el atributo i del ejemplo e:

$$Y(B_i) = \int_0^1 M(B_{i\alpha})d\alpha \tag{10.1}$$

donde $Y(B_i)$ es el valor que representa al dato fuzzy o intervalar del atributo i y $M(B_{i\alpha})$ es el valor medio de los elementos $B_{i\alpha}$, siendo $B_{i\alpha}$ los $\alpha - cortes$ al conjunto fuzzy o intervalo B_i.

Validación de los experimentos mediante tests no paramétricos

A partir de los resultados experimentales llevamos a cabo un análisis de los resultados usando técnicas estadísticas. Más concretamente seguimos la metodología propuesta en [79] y utilizamos tests no paramétricos.

Cuando comparamos multiples técnicas, utilizamos el test de Friedman y el procedimiento de Holm como test post-hoc. El test de Friedman es un test no paramétrico equivalente al ANOVA de medidas repetidas. Bajo la hipótesis nulas, se establece que las técnicas son equivalentes, por lo que un rechazo de esta hipótesis supone la existencia de diferencias en el rendimiento de todas las técnicas estudiadas. Si se detectan diferencias, a continuación se lleva a cabo el procedimiento de Holm usado como un test post-hoc para encontrar si el control o técnica propuesta muestra diferencias estadísticas con respecto a las otras técnicas en comparación. Para llevar a cabo este análisis estadístico, utilizamos el paquete R, [101].

10.5.1.2 Comparando OFP_CLASS$_{\text{FDT}_{\text{LQD}}}$ con otras técnicas de particionamiento

Los resultados de precisión se muestran en la Tabla 10.4, donde %train y %test son los porcentajes en clasificación medios (media y desviación estándar) para los datos de training y test, respectivamente. En negrita se muestran los mejores resultados de la comparación.

Realizamos el análisis estadístico considerando la precisión en el test. Aplicamos el test de Friedman para comprobar si existen diferencias significativas entre todas las técnicas. La hipótesis nula es rechazada con un 99 % de nivel de confianza (α=0.01), es decir, aceptamos que hay diferencias significativas entre las técnicas.

En la Figura 10.8 los valores medios del ranking utilizando el test de Friedman para las técnicas FSGA$_{\text{ANN}}$, FSGA$_{\text{C4.5}}$, FMVR, FWMVR, NFDT, FID y OFP_CLASS$_{\text{FDT}_{\text{LQD}}}$, respectivamente. El valor más bajo indica que su técnica asociada es la mejor. Un test post-hoc es necesario para distinguir si la técnica de control OFP_CLASS$_{\text{FDT}_{\text{LQD}}}$, la cual obtiene el valor más bajo del ranking en el test de Friedman) es significativamente mejor que el resto de técnicas. La Tabla 10.5 muestra todas las posibles comparaciones de las hipótesis de la técnica de control con las otras técnicas, ordenadas por sus p-values y asociando sus niveles de significancia α. El procedimiento de Holm rechaza la hipótesis, por lo que la técnica OFP_CLASS$_{\text{FDT}_{\text{LQD}}}$

Tabla 10.4: Precisión en train y test

Conjuntos	$FSGA_{ANN}$		$FSGA_{C4.5}$		FMVR		FWMVR		NFDT		FID		$OFP\_CLASS_{FDT_{LQD}}$	
de datos	%train	%test	%train	%test	%train	%test	%train	%test	%train	%test	%train	%test	%train	%test
AUS	$87.71_{(0.6)}$	$85.46_{(2.9)}$	$86.12_{(0.6)}$	$85.46_{(2.4)}$	$63.12_{(1.5)}$	$63.00_{(5.5)}$	$55.51_{(1.1)}$	$55.51_{(4.5)}$	$87.52_{(1.0)}$	$85.12_{(2.5)}$	$90.48_{(2.8)}$	$84.59_{(2.1)}$	$89.55_{(0.8)}$	$\mathbf{85.51}_{(3.4)}$
GER	$90.68_{(0.5)}$	$69.67_{(3.1)}$	$80.10_{(0.6)}$	$71.67_{(3.3)}$	$56.05_{(3.1)}$	$54.83_{(6.3)}$	$53.23_{(6.0)}$	$52.10_{(7.2)}$	$77.12_{(0.8)}$	$74.23_{(3.5)}$	$70.08_{(0.8)}$	$70.00_{(2.7)}$	$79.38_{(0.7)}$	$\mathbf{75.17}_{(2.9)}$
GLA	$79.67_{(1.2)}$	$63.53_{(6.8)}$	$77.22_{(1.7)}$	$66.32_{(8.4)}$	$40.20_{(4.8)}$	$40.00_{(7.3)}$	$42.45_{(2.6)}$	$42.10_{(4.1)}$	$73.03_{(2.7)}$	$68.24_{(7.5)}$	$69.56_{(8.5)}$	$65.09_{(8.5)}$	$83.80_{(1.7)}$	$\mathbf{74.77}_{(4.7)}$
ION	$98.77_{(0.4)}$	$91.46_{(3.7)}$	$94.75_{(0.8)}$	$92.17_{(2.7)}$	$70.51_{(1.2)}$	$69.95_{(4.6)}$	$70.30_{(1.4)}$	$70.17_{(4.7)}$	$91.50_{(2.2)}$	$87.76_{(4.8)}$	$94.87_{(1.0)}$	$91.08_{(3.9)}$	$94.85_{(0.4)}$	$\mathbf{93.74}_{(2.0)}$
IRP	$95.22_{(0.8)}$	$91.56_{(4.3)}$	$94.44_{(0.9)}$	$90.89_{(4.2)}$	$90.89_{(1.3)}$	$90.22_{(4.2)}$	$92.22_{(1.3)}$	$91.56_{(3.7)}$	$96.28_{(1.3)}$	$95.33_{(4.9)}$	$94.39_{(1.7)}$	$95.11_{(4.8)}$	$97.50_{(0.9)}$	$\mathbf{97.33}_{(3.8)}$
PIM	$92.39_{(1.0)}$	$73.31_{(3.5)}$	$83.06_{(1.4)}$	$75.30_{(1.8)}$	$66.11_{(2.5)}$	$67.01_{(4.9)}$	$65.93_{(2.5)}$	$66.97_{(3.7)}$	$79.07_{(1.7)}$	$73.26_{(2.4)}$	$73.64_{(0.7)}$	$73.13_{(2.7)}$	$80.27_{(0.7)}$	$\mathbf{77.34}_{(2.4)}$
SON	$99.55_{(0.9)}$	$77.08_{(7.2)}$	$97.52_{(1.1)}$	$70.53_{(5.0)}$	$56.37_{(1.9)}$	$55.10_{(6.2)}$	$54.80_{(4.7)}$	$53.52_{(6.7)}$	$83.69_{(3.6)}$	$72.04_{(6.3)}$	$83.37_{(4.8)}$	$75.48_{(6.9)}$	$82.65_{(1.6)}$	$\mathbf{78.03}_{(6.1)}$
SON*	$97.71_{(1.0)}$	$75.14_{(6.6)}$	$91.62_{(1.0)}$	$81.71_{(4.4)}$	$68.71_{(5.3)}$	$68.89_{(9.3)}$	$64.95_{(9.2)}$	$65.17_{(8.8)}$	$81.62_{(7.0)}$	$73.62_{(6.4)}$	$83.70_{(3.9)}$	$74.20_{(6.9)}$	$85.38_{(1.1)}$	$\mathbf{79.49}_{(5.5)}$
SPE	$99.22_{(0.4)}$	$73.54_{(5.5)}$	$87.77_{(3.8)}$	$74.16_{(7.6)}$	$52.12_{(2.3)}$	$52.07_{(5.2)}$	$59.45_{(10.5)}$	$56.48_{(12.9)}$	$86.75_{(4.1)}$	$79.83_{(6.0)}$	$79.77_{(1.1)}$	$78.64_{(3.8)}$	$88.26_{(1.1)}$	$\mathbf{84.14}_{(5.6)}$
THY	$96.98_{(0.7)}$	$93.33_{(4.0)}$	$94.73_{(1.0)}$	$93.33_{(4.0)}$	$79.37_{(17.7)}$	$79.12_{(17.5)}$	$82.97_{(17.0)}$	$82.53_{(15.5)}$	$96.24_{(1.7)}$	$91.16_{(4.0)}$	$92.56_{(2.4)}$	$91.01_{(4.7)}$	$97.59_{(0.8)}$	$\mathbf{95.84}_{(3.6)}$
WDBC	$99.18_{(0.3)}$	$94.49_{(1.9)}$	$96.61_{(0.5)}$	$93.91_{(1.5)}$	$93.04_{(0.8)}$	$92.91_{(2.2)}$	$92.30_{(1.9)}$	$92.97_{(3.1)}$	$94.05_{(1.6)}$	$92.23_{(2.1)}$	$96.73_{(1.4)}$	$94.50_{(2.8)}$	$97.14_{(0.3)}$	$\mathbf{96.20}_{(1.8)}$
WDG	$94.79_{(0.7)}$	$\mathbf{78.90}_{(1.7)}$	$86.32_{(1.4)}$	$76.35_{(1.8)}$	$40.97_{(0.5)}$	$41.02_{(1.1)}$	$56.89_{(8.0)}$	$57.36_{(8.2)}$	$72.51_{(1.4)}$	$71.46_{(1.8)}$	$72.05_{(1.2)}$	$68.87_{(1.7)}$	$81.73_{(0.6)}$	$78.33_{(1.9)}$
WIN	$100.0_{(0.0)}$	$93.82_{(2.6)}$	$95.74_{(0.9)}$	$90.45_{(4.3)}$	$95.50_{(1.5)}$	$96.22_{(3.6)}$	$92.83_{(2.0)}$	$92.85_{(4.6)}$	$95.29_{(1.5)}$	$93.37_{(4.6)}$	$99.35_{(0.3)}$	$95.13_{(3.6)}$	$100.00_{(0.0)}$	$\mathbf{96.99}_{(3.1)}$
ZOO	$72.11_{(2.2)}$	$72.11_{(8.7)}$	$72.11_{(2.2)}$	$71.16_{(9.4)}$	$60.20_{(6.0)}$	$60.00_{(8.6)}$	$61.10_{(7.1)}$	$60.00_{(8.6)}$	$98.35_{(1.0)}$	$94.06_{(3.9)}$	$99.92_{(0.2)}$	$\mathbf{94.13}_{(4.8)}$	$97.44_{(0.9)}$	$94.02_{(7.2)}$
Media	$93.14_{(0.8)}$	$80.96_{(4.5)}$	$88.44_{(1.3)}$	$80.96_{(4.3)}$	$66.65_{(3.6)}$	$66.45_{(6.2)}$	$67.50_{(5.4)}$	$67.09_{(6.9)}$	$86.64_{(2.3)}$	$82.27_{(4.4)}$	$85.75_{(2.2)}$	$82.21_{(4.3)}$	$89.68_{(0.8)}$	$\mathbf{86.21}_{(3.9)}$

es estadísticamente mejor respecto a la precisión que el resto de técnicas con un valor de α de 0.01.

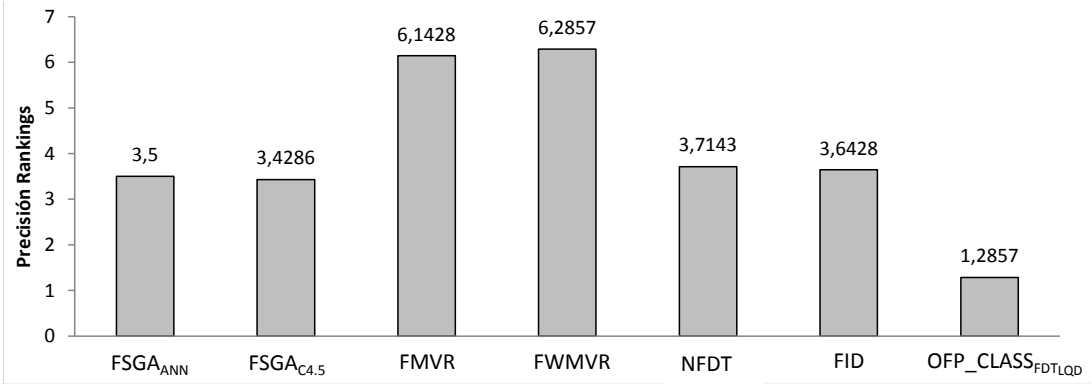


Figura 10.8: Precisión de los rankings de Friedman

Tabla 10.5: Resultados del procedimiento del test estadístico post hoc de Holm (OFP_-CLASS$_{FDT_{LQD}}$ es la técnica de control)

Técnica	**p-value**	**α/i**	**Hipótesis (α=0.01)**
FRVR	6.104e-05	0.00167	Rechazada
FWMVR	6.104e-05	0.002	Rechazada
FID	0.0001221	0.0025	Rechazada
NFDT	0.0001221	0.00333	Rechazada
FSGA$_{ANN}$	0.0001831	0.005	Rechazada
FSGA$_{C4.5}$	0.0006104	0.01	Rechazada

La ejecución de las técnicas FSGA, FMVR, FWMVR, NFDT, FID y OFP_CLASS tardan sobre 20, 9, 11, 18, 56 y 12 minutos, respectivamente, de media de tiempo de CPU sobre un Pentium Dual Core 2Ghz con 2GB de memoria. Como podemos observar entre el resto de técnicas, el coste computacional de OFP_CLASS es aceptable.

10.5.1.3 **Comparando OFP_CLASS con las técnicas FSGA y FID usando el mismo clasificador como evaluador**

Una vez que hemos demostrado el buen rendimiento, en términos de precisión, de la técnica OFP_CLASS$_{FDT_{LQD}}$, en este segundo experimento vamos a analizar el rendimiento de las particiones obtenidas. En este experimento vamos a comparar las particiones fuzzy obtenidas por FSGA, FID y OFP_CLASS utilizando el mismo evaluador. El evaluador es el árbol de decisión fuzzy propuesto por Janikov en [104]. Vamos a denotar por FSGA$_{FID}$ y OFP_-CLASS$_{FID}$, al clasificador FID que utiliza las particiones generadas por las técnicas FSGA y

OFP_CLASS respectivamente. En este experimento no comparamos con las técnicas NFDT, FMWR y FWMVR, porque ellos no obtienen particiones generales que puedan ser usadas por otros clasificadores. Las particiones obtenidas por estas técnicas son muy específicas para el clasificador o algoritmo que las genera.

Los resultados de precisión se muestran en la Tabla 10.6. %train y %test son los porcentajes medios de precisión obtenidos en clasificación (media y desviación estándar) para los datos de training y test, respectivamente. En negrita se muestran los mejores resultados de la comparación. Además, en la Tabla 10.6 también se muestra el tamaño medio de las particiones obtenidas por las técnicas FSGA, FID y OFP_CLASS (columnas 2, 4 y 6 respectivamente). Por último, si nosotros definimos la cardinalidad de una partición fuzzy para un conjunto de datos como la suma de las cardinalidades de sus atributos, la columna "ACard" indica la media de las cardinalidades de las particiones fuzzy obtenidas para las diferentes validaciones.

Al igual que en el experimento anterior, el análisis estadístico se lleva a cabo considerando solamente la precisión del test. Aplicamos el test de Friedman para verificar si existen diferencias significativas entre todas las técnicas. Aplicando este test, rechazamos la hipótesis nula (p-value = 0.0007819) con un 99 % de nivel de confianza (α=0.01), es decir, aceptamos que existen diferencias significativas.

Los valores medios del ranking utilizando el test de Friedman son 2.5714, 2.2857 y 1.1428 para las técnicas FSGA_{FID}, FID y $\text{OFP\_CLASS}_{\text{FID}}$ respectivamente (en la Figura 10.9 se especifican los valores medios de los rankings para cada técnica usando el test de Friedman). El valor más bajo indica que la técnica que lo tiene asociado es el mejor. Tras aplicar el test de Friedman, un test post-hoc es necesario para distinguir si la técnica de control ($\text{OFP\_CLASS}_{\text{FID}}$, la cual obtiene el valor más bajo en el ranking computado por el test de Friedman) es significativamente mejor que el resto de técnicas. La Tabla 10.7 muestra todas las posibles hipótesis de comparación entre la técnica de control y las otras técnicas y sus p-values ajustados según el test post-hoc. El procedimiento de Holm rechaza todas las hipótesis, por lo que $\text{OFP\_CLASS}_{\text{FID}}$ es estadísticamente mejor en precisión que el resto de las técnicas con un valor de α de 0.01. Con estos resultados, podemos concluir que las particiones fuzzy obtenidas por la técnica OFP_CLASS son las mejores particiones fuzzy para los atributos numéricos en la tarea de clasificación. Además en la Tabla 10.6, también se puede verificar que la media de cardinalidad de las particiones fuzzy (columna "ACard") también es la más baja para la técnica OFP_CLASS.

Tabla 10.6: Precisión del train y test usando FID como evaluador

		$FSGA_{FID}$			FID			OFP_CLASS_{FID}	
Conjuntos de datos	**"ACard" para particiones FSGA**	**%train**	**%test**	**"ACard" para particiones de FID**	**%train**	**%test**	**"ACard" para particiones OFP_CLASS**	**%train**	**%test**
AUS	30	$88.39_{(2.7)}$	$\mathbf{87.20}_{(4.0)}$	24	$90.48_{(2.8)}$	$84.01_{(2.1)}$	46	$85.42_{(1.3)}$	$84.78_{(3.0)}$
GER	55	$70.00_{(0.6)}$	$\mathbf{70.00}_{(2.3)}$	64	$70.08_{(0.8)}$	$70.00_{(2.7)}$	21	$70.00_{(0.6)}$	$\mathbf{70.00}_{(2.3)}$
GLA	18	$53.62_{(0.9)}$	$49.84_{(6.1)}$	43	$69.56_{(8.5)}$	$65.09_{(8.5)}$	36	$76.64_{(4.0)}$	$\mathbf{70.15}_{(4.5)}$
ION	85	$93.35_{(2.6)}$	$86.43_{(4.4)}$	37	$94.87_{(1.0)}$	$90.79_{(3.9)}$	27	$91.60_{(1.4)}$	$\mathbf{91.57}_{(2.6)}$
IRP	12	$92.56_{(1.3)}$	$89.56_{(7.4)}$	9	$94.39_{(1.7)}$	$95.11_{(4.8)}$	10	$97.94_{(0.5)}$	$\mathbf{97.10}_{(3.8)}$
PIM	30	$74.06_{(0.9)}$	$\mathbf{73.70}_{(3.0)}$	29	$73.64_{(0.7)}$	$72.24_{(2.7)}$	36	$72.27_{(1.0)}$	$72.88_{(2.9)}$
SON	217	$58.91_{(12.7)}$	$53.36_{(4.9)}$	74	$83.37_{(4.8)}$	$75.48_{(6.9)}$	49	$84.01_{(3.5)}$	$\mathbf{75.87}_{(6.0)}$
SON*	105	$55.89_{(6.4)}$	$53.99_{(6.7)}$	54	$83.70_{(3.9)}$	$74.20_{(6.9)}$	50	$82.45_{(1.8)}$	$\mathbf{75.95}_{(6.2)}$
SPE	54	$79.55_{(1.3)}$	$79.27_{(4.5)}$	68	$79.77_{(1.1)}$	$78.64_{(3.8)}$	16	$79.40_{(1.2)}$	$\mathbf{79.40}_{(4.6)}$
THY	24	$79.84_{(6.2)}$	$77.36_{(5.8)}$	12	$92.56_{(2.4)}$	$91.01_{(4.7)}$	18	$97.40_{(1.3)}$	$\mathbf{94.11}_{(3.5)}$
WDBC	85	$96.21_{(0.9)}$	$93.74_{(3.0)}$	46	$96.73_{(1.4)}$	$94.14_{(2.8)}$	25	$94.86_{(1.0)}$	$\mathbf{94.42}_{(2.8)}$
WDG	48	$57.35_{(6.6)}$	$55.71_{(8.5)}$	105	$72.05_{(1.2)}$	$68.87_{(1.7)}$	55	$72.47_{(0.9)}$	$\mathbf{70.99}_{(1.2)}$
WIN	40	$98.55_{(0.9)}$	$87.98_{(7.2)}$	19	$99.35_{(0.3)}$	$95.13_{(3.6)}$	27	$99.44_{(0.3)}$	$\mathbf{96.06}_{(3.5)}$
ZOO	3	$99.92_{(0.2)}$	$94.38_{(6.2)}$	0	$99.92_{(0.2)}$	$94.13_{(4.8)}$	4	$99.92_{(0.2)}$	$\mathbf{94.68}_{(7.1)}$
Media	57.57	$78.44_{(3.2)}$	$75.18_{(5.3)}$	41.71	$85.75_{(2.2)}$	$82.06_{(4.3)}$	30	$85.99_{(1.4)}$	$\mathbf{83.43}_{(3.9)}$

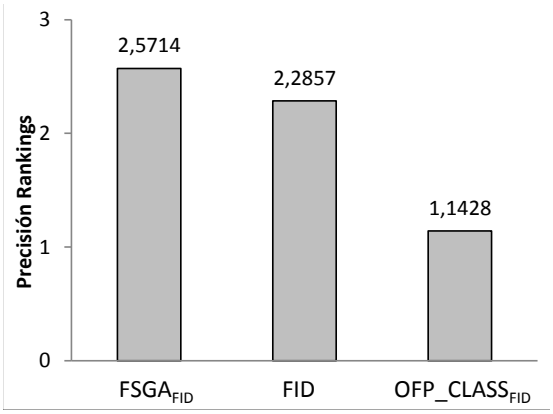


Figura 10.9: Precisión de los Rankings de Friedman

Tabla 10.7: Resultados del procedimiento del test estadístico post hoc de Holm (OFP_CLASS_{FID} es la técnica de control)

Técnica	**p-value**	α**/i**	**Hipótesis** (α**=0.01**)
FID	6.104e-05	0.005	Rechazada
$FSGA_{FID}$	0.003357	0.01	Rechazada

10.5.2 Experimentos con OFP_CLASS_{LQD}

En esta sección vamos a mostrar diferentes experimentos para evaluar si las particiones fuzzy que se construyen directamente sobre datos LQD son mejores que las particiones fuzzy que se construyen sobre los datos transformados (datos LQD convertidos en datos crisp).

10.5.2.1 Marco Experimental

Los experimentos han sido diseñados para medir el comportamiento de las particiones fuzzy generadas con las técnicas OFP_CLASS y OFP_CLASS_{LQD} en el ensamble FRF_{LQD} utilizando los conjuntos de datos y los resultados presentados en [160, 161], donde los autores utilizan un clasificador fuzzy basado en reglas para clasificar conjuntos de datos que contienen explícitamente datos imprecisos, tales como valores missing e intervalos. Además, los autores de estos trabajos utilizan particiones uniformes para evaluar los conjuntos de datos y en nuestro caso, vamos a mostrar cómo los resultados pueden ser mejores cuando se usan las particiones fuzzy generados por OFP_CLASS, aunque ellas sean construidas utilizando un conjunto de datos modificado para eliminar los LQD y sustituirlos por valores crisp, en lugar del conjunto de datos original. También vamos a mostrar cómo los resultados de clasificación son todavía mejores si no se modifican los datos para construir las particiones fuzzy usando la técnica OFP_CLASS_{LQD}. Por último, para comparar nuestros resultados con los resultados obtenidos

en [160, 161], vamos a definir una configuración experimental bastante similar a la propuesta por ellos.

Conjuntos de datos y parámetros para FRF_{LQD}

Para evaluar las particiones fuzzy, vamos a los utilizar los conjuntos de datos del mundo real sobre diagnóstico médico y sobre alto rendimiento en atletas, [160, 161], que hemos descrito en la Subsección 7.4.3 (disponibles en "http://sci2s.ugr.es/ keel/"). Estos conjuntos de datos se detallan en la Tabla 10.8.

Tabla 10.8: Conjuntos de datos

Conjuntos de datos	**|E|**	**|A|**	**C**	**Conjuntos de datos**	**|E|**	**|A|**	**C**
100ml-4-I	52	4	2	Dyslexic-12	65	12	4
100ml-4-P	52	4	2	Dyslexic-12-01	65	12	3
Long-4	25	4	2	Dyslexic-12-12	65	12	3

La Tabla 10.8 muestra el número de ejemplos ($|E|$), el número de atributos ($|A|$) y el número de clases (C) para cada conjunto de datos. "Abbr" indica la abreviación del nombre del conjunto de datos utilizada durante los experimentos.

Todos los ensambles FRF_{LQD} utilizan un tamaño de bosque de 100 árboles. El número de atributos elegidos aleatoriamente en un nodo dado es $\log_2(|\cdot|+1)$, donde $|\cdot|$ es el número de atributos disponibles en ese nodo y cada árbol que compone el ensamble FRF se construye a tamaño máximo (nodo puro o conjunto de atributos disponibles vacío) y sin poda.

10.5.2.2 Evaluación de las particiones fuzzy

Los experimentos realizados se centran en los dos tipos de conjuntos de datos. En el primero, se comparan los resultados del clasificador GGFS propuesto en [160], el cual utiliza particiones uniformes, con los resultados obtenidos por el ensamble FRF_{LQD} cuando se utilizan particiones construidas con la técnica OFP_CLASS aplicada a los conjuntos de datos modificados (crisp). Así, cambiamos los datos intervalares o fuzzy por su valor medio. En el segundo, comparamos los resultados de OFP_CLASS con $\text{OFP\_CLASS}_{\text{LQD}}$.

Atletismo de alto rendimiento

Los resultados obtenidos sobre estos conjuntos de datos se muestran en la Tabla 10.9.

Los resultados obtenidos en clasificación por la técnica extendida de generalización de un GFS (GGFS) propuesto en [160] y por el ensamble FRF_{LQD} al usar las particiones fuzzy generadas por la técnica $\text{OFP\_CLASS}_{\text{LQD}}$, son muy prometedores porque ellos representan

Tabla 10.9: Resultados comparativos para los conjuntos de datos de atletismo de alto rendimiento

		Conjuntos de datos					
		100ml-4-I		100ml-4-P		Long-4	
	Técnica	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
OFP_CLASS$_{LQD}$ Fuzzy partition	FRF_{LQDSM1}	[0.107,0.305]	[0.130,0.323]	[0.043,0.235]	[0.093,0.290]	[0.191,0.484]	[0.083,0.349]
	FRF_{LQDSM2}	[0.110,0.306]	[0.150,0.343]	[0.045,0.237]	[0.110,0.307]	[0.165,0.449]	[0.083,0.349]
	$FRF_{LQDMWL1}$	[0.070,0.265]	**[0.073,0.267]**	[0.032,0.224]	**[0.060,0.257]**	[0.085,0.364]	**[0.033,0.299]**
	$FRF_{LQDMWL2}$	[0.060,0.254]	[0.113,0.306]	[0.043,0.235]	**[0.060,0.257]**	[0.111,0.391]	[0.083,0.349]
	$FRF_{LQDMWLT1}$	[0.070,0.267]	**[0.073,0.267]**	[0.032,0.224]	**[0.060,0.257]**	[0.085,0.364]	**[0.033,0.299]**
	$FRF_{LQDMWLT2}$	[0.060,0.252]	[0.093,0.286]	[0.038,0.231]	**[0.060,0.257]**	[0.107,0.386]	[0,083,0.349]
OFP_CLASS Fuzzy partition	FFR_{LQDSM1}	[0.139,0.331]	[0.150,0.343]	[0.098,0.291]	[0.133,0.310]	[0.120,0.404]	[0.200,0.467]
	FRF_{LQDSM2}	[0.141,0.333]	[0.150,0.343]	[0.096,0.288]	[0.093,0.290]	[0.115,0.391]	[0.200,0.467]
	$FRF_{LQDMWL1}$	[0.077,0.269]	[0.093,0.287]	[0.075,0.269]	[0.073,0.270]	[0.116,0.396]	[0.100,0.417]
	$FRF_{LQDMWL2}$	[0.060,0.252]	[0.093,0.287]	[0.077,0.269]	[0.073,0.270]	[0.102,0.382]	[0.100,0.367]
	$FRF_{LQDMWLT1}$	[0.077,0.269]	[0.093,0.287]	[0.075,0.267]	[0.073,0.270]	[0.107,0.387]	[0.150,0.417]
	$FRF_{LQDMWLT2}$	[0.062,0.254]	[0.093,0.287]	[0.077,0.269]	[0.073,0.270]	[0.094,0.373]	[0.067,0.333]
	Crisp [160]	0.259	0.384	0.288	0.419	0.327	0.544
	GGFS [160]	[0.089,0.346]	[0.189,0.476]	[0.076,0.320]	[0.170,0.406]	[0.000,0.279]	[0.349,0.616]

la información de una forma más natural y apropiada. Además en este problema, estamos facilitando al entrenador la información por valores y en términos lingüísticos.

Los resultados del ensamble FRF_{LQD} son muy competitivos con todas las particiones fuzzy, pero con las particiones obtenidas con el algoritmo OFP_CLASS$_{LQD}$ son mejores, ya que las particiones se han construido teniendo en cuenta la naturaleza de los datos, sin llevar a cabo ninguna transformación en los mismos. Para cada conjunto de datos el mejor resultado obtenido con respecto al test se encuentra resaltado en negrita.

Diagnóstico de Dislexia

Todos los experimentos han sido repetidos 100 veces mediante bootstrap con reemplazamiento del conjunto de entrenamiento. El conjunto de test lo componen los ejemplos OOB no utilizados en el aprendizaje.

La Tabla 10.10 muestra los resultados obtenidos cuando ejecutamos el ensamble FRF_{LQD} con las particiones fuzzy obtenidas por la técnica OFP_CLASS y con las particiones fuzzy obtenidas con la técnica OFP_CLASS$_{LQD}$ para los conjuntos de datos "Dyslexic-12", "Dyslexic-12-01" y "Dyslexic-12-12'.

Además en la Tabla 10.10, también comparamos estos resultados con el mejor resultado obtenido en [161] ($(*)$: partición utilizada- cuatro etiquetas; $(**)$ partición utilizada - cinco etiquetas). De nuevo, en esta tabla se muestra el intervalo $[media\_min\_error, media\_max\_error]$

Tabla 10.10: Comparando los resultados para los conjuntos de datos de dislexia

		Conjuntos de datos					
		Dyslexic-12		Dyslexic-12-01		Dyslexic-12-12	
	Técnica	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
OFP_CLASS$_{LQD}$ Partición Fuzzy	FRF_{LQDSM1}	[0.000,0.238]	**[0.000,0.398]**	[0.022,0.223]	[0.039,0.377]	[0.001,0.263]	[0.035,0.422]
	FRF_{LQDSM2}	[0.000,0.228]	[0.000,0.399]	[0.008,0.184]	**[0.022,0.332]**	[0.009,0.245]	[0.032,0.411]
	$FRF_{LQDMWL1}$	[0.000,0.270]	[0.000,0.406]	[0.017,0.231]	[0.045,0.383]	[0.001,0.273]	[0.019,0.430]
	$FRF_{LQDMWL2}$	[0.000,0.270]	[0.000,0.407]	[0.020,0.241]	[0.056,0.385]	[0.001,0.267]	[0.026,0.406]
	$FRF_{LQDMWLT1}$	[0.000,0.263]	[0.000,0.402]	[0.012,0.216]	[0.038,0.365]	[0.000,0.265]	**[0.019,0.427]**
	$FRF_{LQDMWLT2}$	[0.000,0.266]	[0.000,0.404]	[0.015,0.221]	[0.049,0.373]	[0.000,0.262]	[0.024,0.422]
OFP_CLASS Partición Fuzzy	FRF_{LQDSM1}	[0.000,0.320]	[0.002,0.511]	[0.000,0.282]	[0.000,0.413]	[0.000,0.405]	[0.000,0.477]
	FRF_{LQDSM2}	[0.000,0.327]	[0.001,0.515]	[0.000,0.253]	[0.000,0.389]	[0.000,0.402]	[0.000,0.469]
	$FRF_{LQDMWL1}$	[0.000,0.261]	[0.003,0.419]	[0.000,0.264]	[0.000,0.400]	[0.000,0.335]	[0.000,0.422]
	$FRF_{LQDMWL2}$	[0.000,0.270]	[0.003,0.423]	[0.000,0.276]	[0.000,0.407]	[0.000,0.343]	[0.000,0.414]
	$FRF_{LQDMWLT1}$	[0.000,0.264]	[0.004,0.419]	[0.000,0.243]	[0.000,0.386]	[0.000,0.331]	[0.000,0.422]
	$FRF_{LQDMWLT2}$	[0.000,0.267]	[0.003,0.417]	[0.000,0.259]	[0.000,0.394]	[0.000,0.343]	[0.000,0.418]
(*)	Crisp CF^0 [161]	0.444	[0.572,0.694]	0.336	[0.452,0.533]	0.390	[0.511,0.664]
(*)	GGFS [161]	–	[0.421,0.558]	–	[0.219,0.759]	–	[0.199,0.757]
(*)	GGFS CF^0 [161]	[0.003,0.237]	[0.405,0.548]	[0.005,0.193]	[0.330,0.440]	[0.003,0.243]	[0.325,0.509]
(**)	Crisp CF^0 [161]	0.556	[0.614,0.731]	0.460	[0.508,0.605]	0.485	[0.539,0.692]
(**)	GGFS [161]	–	[0.490,0.609]	–	[0.323,0.797]	–	[0.211,0.700]
(**)	GGFS CF^0[161]	[0.038,0.233]	[0.480,0.621]	[0.000,0.187]	[0.394,0.522]	[0.000,0.239]	[0.393,0.591]

obtenido para cada conjunto de datos de acuerdo al proceso de decisión descrito en el Algoritmo 11. Para cada conjunto de datos el mejor resultado obtenido con respecto al test se encuentra resaltado en negrita.

Como comentario sobre todos los experimentos, debemos de añadir que se puede observar que el ensamble FRF_{LQD} con las particiones fuzzy obtenidas por la técnica OFP_CLASS$_{LQD}$ son mejores que los resultados en test obtenidos por FRF_{LQD} con las particiones fuzzy creadas con la técnica OFP_CLASS. Además FRF_{LQD} obtiene una significante mejora sobre la técnica GGFS crisp. En estos experimentos se muestra que cuando las particiones fuzzy son obtenidas con los datos originales utilizando la técnica OFP_CLASS$_{LQD}$, la precisión es mayor, (el intervalo de error está más cercano a cero y tiene menos imprecisión). Además como también se discute en [161] es preferible utilizar un algoritmo que sea capaz de aprender con LQD en lugar de eliminar la información imperfecta para utilizar un algoritmo convencional.

10.5.3 Experimentos con BAGOFP_CLASS

Ahora, detallamos el marco experimental en el cual vamos a trabajar para evaluar el comportamiento de la técnica BAGOFP_CLASS. Para ello vamos a comparar los resultados de esta técnica con los resultados de la técnica OFP_CLASS y de la técnica OFP_CLASS$_{\text{LQD}}$, para verificar que la mejora introducida funciona, cuando se trata con conjuntos de datos de tamaño pequeño y sobre conjuntos de datos con LQD.

10.5.3.1 Marco Experimental

A continuación vamos a presentar los conjuntos de datos utilizados y vamos a describir la configuración de los experimentos que vamos a llevar a cabo, así como la medida de rendimiento y los test estadísticos empleados para analizar los resultados.

Los conjuntos de datos y configuración experimental

La técnica de discretización propuesta va a ser evaluado mediante un conjunto de experimentos sobre varios conjuntos de datos seleccionados del repositorio de aprendizaje automático UCI, [77]. Además, se utilizan 3 conjuntos de datos multidimensionales (ADE, PRO, SRB), que se encuentran disponibles en [65]. Estos conjuntos de datos que se han utilizado como test se describen resumidamente en la Tabla 10.11

Tabla 10.11: Conjuntos de datos

Conjuntos de datos	Abbr	|E|	|A|	|C|	regla	Conjuntos de datos	Abbr	|E|	|A|	|C|	regla
Australian credit	AUS	690	14	2	SI	Adenocarcinoma	ADE	76	9868	2	NO
Breast Cancer W.	BCW	699	9	2	SI	Apendicitis	APE	106	7	2	NO
Statlog Heart	HEA	270	13	2	SI	Glass	GLA	214	9	7	NO
Iris Plant	IRP	150	4	3	SI	Ionosphere	ION	351	34	2	NO
Vehicle	VEH	946	18	4	SI	Prostate	PRO	102	6033	2	NO
						Sonar	SON	208	60	2	NO
						SPECTF	SPE	267	44	2	NO
						Srbct	SRB	63	2308	4	NO
						Wis. D. Breast C.	WDB	569	31	2	NO
						Wine	WIN	178	13	3	NO

La Tabla 10.11 muestra para cada conjunto de datos, el número de ejemplos ($|E|$), el número de atributos y el número de clases ($|C|$). Todos los conjuntos de datos tienen atributos numéricos, excepto AUS que tiene seis atributos numéricos y 8 atributos nominales. Además la última columna "regla", de la Tabla 10.11 muestra si el conjunto de datos verifica o no la

condición $|E| \geq 10 \cdot |A| \cdot |C|$, es decir, si consideramos o no que el conjunto de datos es de tamaño pequeño o no. "Abbr" indica la abreviatura utilizada durante los experimentos para cada conjunto de datos.

Para evaluar las particiones obtenidas tanto por la técnica OFP_CLASS como por la técnica BAGOFP_CLASS, vamos a utilizar como clasificador el ensamble Fuzzy Random Forest FRF_{LQD},[42]. Este ensamble como ya hemos comentado se compone de un conjunto de árboles de decisión y necesita para poder trabajar, disponer de una discretización de los atributos numéricos mediante una partición fuzzy.

Los parámetros utilizados para estos experimentos son los siguientes:

- Parámetros para el ensamble FRF_{LQD}.
  - Tamaño ensamble: 500 árboles de decisión.
  - Selección aleatoria del conjunto de atributos disponibles: $\sqrt{|A|}$ (siendo $|A|$ el número total de atributos)
- Parámetros para los algoritmos genéticos en ambas técnicas $\text{OFP\_CLASS}_{\text{LQD}}$ y BAGOFP_CLASS.
  - Número de individuos: 100
  - Número de generaciones: 250
  - Probabilidad de Cruce: 0.8
  - Probabilidad de Mutación: 0.1

Los valores de los parámetros δ y β de la técnica BAGOFP_CLASS son establecidos más adelante en esta sección ya que realizamos un breve estudio de cómo afectan en el comportamiento de la técnica y seleccionamos los mejores valores para llevar a cabo la experimentación.

Estimación del rendimiento de clasificación y validación de los resultados experimentales

Para analizar los resultados obtenidos en este estudio, empleamos la siguiente medida de rendimiento: para comparar los resultado en los experimentos utilizamos la precisión como medida de rendimiento (número de aciertos relativos divididos por el número total de clasificaciones) Además más específicamente, la precisión media es obtenida como la media de la precisión de un validación cruzada de 5 repetida 3 veces.

Para completar el estudio experimental, realizamos un análisis en cada subsección usando técnicas estadísticas. Concretamente seguimos la metodología propuesta por García et al. en [79], utilizando test no paramétricos.

Utilizamos el test de signos de Wilcoxon para comparar dos técnicas. Este test es un procedimiento estadístico no paramétricos para llevar a cabo una comparación de pares entre dos técnicas. Éste es análogo al test de pareados t-test. Así este test de pares tiene como objetivo detectar diferencias significativas entre dos medias de muestras, es decir, entre el comportamiento de dos técnicas. Para llevar a cabo el análisis estadísticos, utilizamos el paquete R, [101].

Parámetros δ y β de la técnica BAGOFP_CLASS

Como hemos comentado al inicio de esta sección, el uso del bagging está justificado por el hecho de que el proceso de bagging proporciona una ganancia substancial en precisión cuando la técnica de clasificación utilizada es inestable, (árboles de decisión, en nuestro caso). En esta subsección vamos a realizar un pequeño estudio para mostrar computacionalmente este hecho. Como resultado, vamos a seleccionar los valores para δ y β de la técnica BAGOFP_CLASS que usaremos en los experimentos. Vamos a mostrar el comportamiento del clasificador FRF_{LQD} usando las diferentes particiones generadas por la técnica BAGOFP_CLASS y modificando los parámetros δ y β de la siguiente manera: primero establecemos el valor de $\beta = 1$ y luego vamos variando δ desde 1 hasta 20. Después establecemos el valor de $\delta = 20$ y variamos el valor de β desde 2 hasta 30. En las Figuras 10.10 y 10.11 se muestran los resultados obtenidos usando los conjunt os de datos GLAS y ION. El eje horizontal (eje x) muestra los diferentes valores de (δ,β) y el eje vertical (eje y) muestra la media de precisión de un validación cruzada de 3 repetida 5 veces.

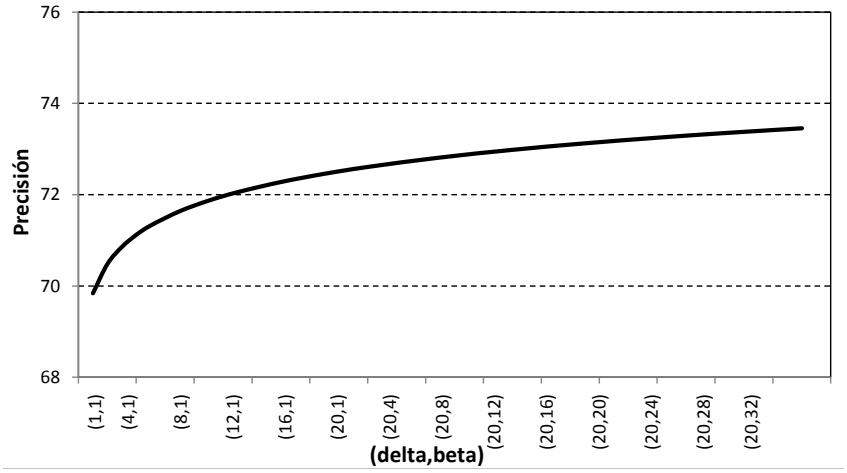


Figura 10.10: Comportamiento de la técnica BAGOFP_CLASS con el conjunto de datos GLAS cuando variamos los parámetros δ y β

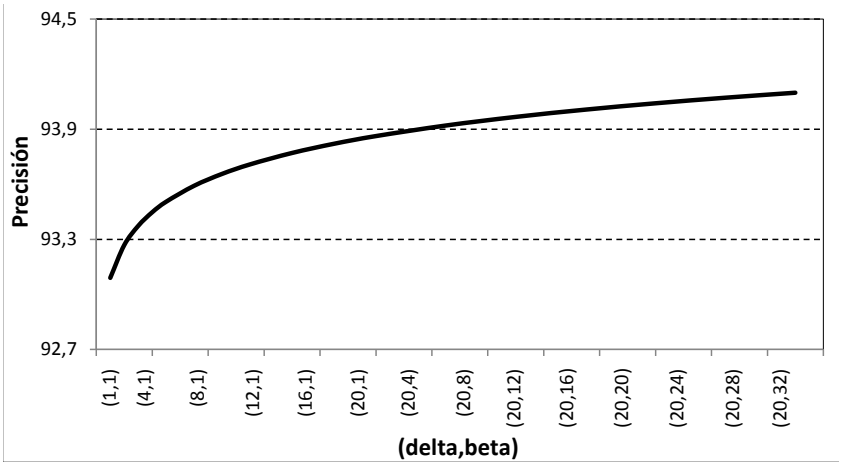


Figura 10.11: Comportamiento de la técnica BAGOFP_CLASS con el conjunto de datos ION cuando variamos los parámetros δ y β

Como se puede apreciar en las Figuras 10.10 y 10.11, en los primeros cambios de δ, la precisión sufre grandes cambios, que posteriormente se van estabilizando.

Los valores de $\delta = 20$ y $\beta = 30$ tienen un rendimiento aceptable con todos los conjuntos de datos utilizados en este trabajo. Por lo tanto, utilizaremos estos valores para los experimentos llevados a cabo en la siguiente subsección.

Los valores de los parámetros δ y β que usaremos, por tanto, en los experimentos son:

- En la primera etapa - δ = 20
- En la segunda etapa - β = 30

10.5.3.2 **Evaluación del rendimiento en clasificación**

Comparando con la técnica OFP_CLASS

Primero, vamos a comparar la técnica OFP_CLASS, [43] con la técnica propuesta BAGOFP_-CLASS utilizando los conjuntos de datos que verifican la regla $|E| \geq 10 \cdot |A| \cdot |C|$. El resultado de precisión medio se muestra en la Tabla 10.12. Después, vamos a comparar estas dos técnicas de discretización utilizando los conjuntos de datos que no verifican la condición. Los resultados medios se muestran en la Tabla 10.13.

En las Tablas 10.12 y 10.13, training y test son los porcentajes medios de precisión (media y desviación estándar) para los conjuntos de datos de training y test, respectivamente. Además se

muestran los p-values obtenidos cuando comparamos los resultados de la fase de test utilizando el test de signos de Wilcoxon.

Tabla 10.12: Resultados obtenidos por FRF_{LQD} después de clasificar los conjuntos de datos que verifican $|E| \geq 10 \cdot |A| \cdot |C|$ utilizando distintas discretizaciones.

	OFP_CLASS		**BAGOFP_CLASS**		
	training	test	training	test	p-value
AUS	$99.85_{0.15}$	$86.42_{2.94}$	$100_{0.00}$	$86.81_{3.12}$	0.1768
BCW	$99.74_{0.10}$	$96.13_{1.27}$	$98.19_{0.22}$	$95.90_{1.55}$	0.7772
HEA	$97.84_{0.54}$	$79.01_{5.34}$	$98.92_{0.41}$	$79.26_{6.56}$	0.5092
IRP	$97.78_{0.92}$	$96.67_{3.26}$	$98.98_{0.62}$	$96.22_{3.16}$	0.6578
VEH	$93.60_{0.47}$	$71.16_{3.48}$	$99.99_{0.02}$	$71.55_{4.80}$	0.4325
media	$97.76_{0.44}$	$85.88_{3.26}$	$99.21_{0.26}$	$85.95_{3.84}$	0.5805

Tabla 10.13: Resultados obtenidos por FRF_{LQD} después de clasificar los conjuntos de datos que no verifican $|E| \geq 10 \cdot |A| \cdot |C|$ utilizando distintas discretizaciones.

	OFP_CLASS		**BAGOFP_CLASS**		
	training	test	training	test	p-value
ADE	$84.32_{2.05}$	$81.94_{9.96}$	$100_{0.00}$	$85.06_{9.83}$	0.01102
APP	$90.33_{1.80}$	$88.33_{7.95}$	$94.02_{0.86}$	$90.25_{5.72}$	0.00529
GLA	$85.82_{1.03}$	$69.80_{5.60}$	$100_{0.00}$	$73.38_{6.68}$	0.01199
ION	$95.89_{0.41}$	$93.07_{2.38}$	$100_{0.00}$	$94.11_{2.05}$	0.00424
PRO	$94.93_{1.36}$	$89.84_{6.98}$	$100_{0.00}$	$93.78_{5.36}$	0.00464
SON	$93.59_{1.59}$	$80.31_{5.11}$	$100_{0.00}$	$83.68_{4.72}$	0.00105
SPE	$79.81_{0.97}$	$79.41_{4.12}$	$100_{0.00}$	$81.78_{4.58}$	0.00108
SRB	$94.58_{1.72}$	$78.46_{13.54}$	$100_{0.00}$	$96.79_{3.77}$	0.00109
WDB	$93.99_{0.41}$	$93.79_{2.23}$	$100_{0.00}$	$95.49_{1.66}$	0.00692
WIN	$94.33_{1.28}$	$93.82_{3.54}$	$100_{0.00}$	$97.57_{1.76}$	0.00253
media	$90.76_{1.26}$	$84.88_{6.14}$	$99.40_{0.09}$	$89.19_{4.61}$	2.2e-16

En la Tabla 10.12, se observa que los resultados obtenidos con las dos técnicas son muy similares entre sí. Analizando los diferentes p-values con $\alpha = 0.05$ podemos concluir que:

- No existen diferencias significativas entre los conjuntos de datos.
- Globalmente, analizando todos los resultados para los diferentes conjuntos de datos, concluimos que no hay diferencias significativas entre evaluar con la discretización obtenida

por la técnica OFP_CLASS y evaluar con la discretización obtenida por la nueva técnica BAGOFP_CLASS.

En la Tabla 10.13, se puede observar que los resultados obtenidos por la nueva técnica parecen ser mejores que los resultados obtenidos por OFP_CLASS. Analizando los diferentes p-values con $\alpha = 0.05$, podemos concluir que:

- Para todos los conjuntos de datos, el análisis indica que hay diferencias significativas entre las dos técnicas, siendo la técnica BAGOFP_CLASS la mejor.
- Globalmente y analizando todos los resultados obtenidos por los diferentes conjuntos de datos, podemos concluir que hay diferencias significativas entre evaluar con la discretización obtenida por la técnica OFP_CLASS y evaluar con la discretización de la nueva técnica, siendo la técnica que obtiene mejor precisión BAGOFP_CLASS.

La Figura 10.12 muestra de una manera ilustrativa, los resultados obtenidos por las técnicas BAGOFP_CLASS y OFP_CLASS con conjuntos de datos que sí verifican la regla, es decir, que contienen suficientes ejemplos.

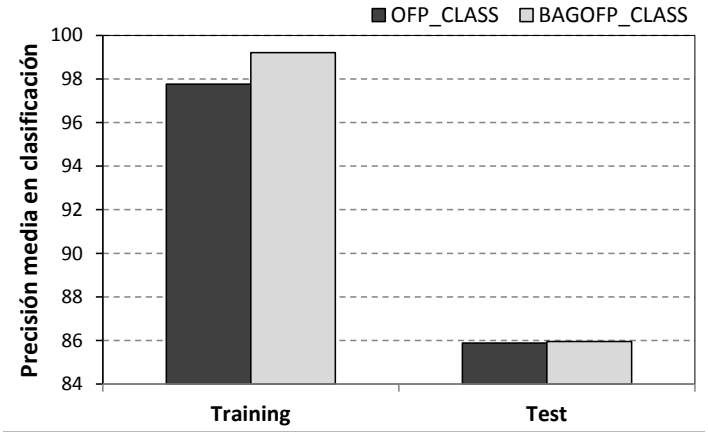


Figura 10.12: Comparando los resultados de los conjuntos de datos que verifican $|E| \geq 10 \cdot |A| \cdot |C|$

La Figura 10.13 muestra de forma ilustrativa los resultados obtenidos usando las técnicas BAGOFP_CLASS y OFP_CLASS con conjuntos de datos de tamaño pequeño.

En general, con los conjuntos de datos utilizados (tanto los que verifican como los que no verifican la regla $|E| \geq 10 \cdot |A| \cdot |C|$), podemos concluir que la técnicaa propuesta es útil. Cuando un conjunto de datos no verifica la condición, la diferencia fundamental de la nueva técnicas de discretización es el particionamiento de los atributos. Los atributos más importantes en la clasificación son particionados probablemente en más partes y en partes más precisas.

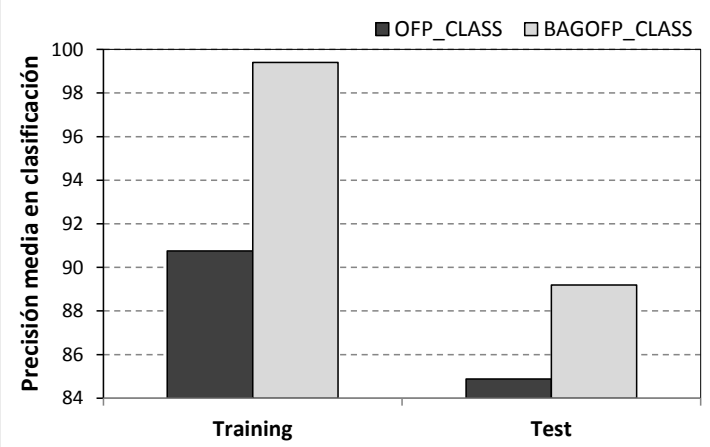


Figura 10.13: Comparando los resultados de los conjuntos de datos que NO verifican $|E| \geq 10 \cdot |A| \cdot |C|$

Desde un punto de vista computacional, la introducción de bagging y de un proceso iterativo en ambas fases puede incrementar el tiempo de computación de la técnica. Sin embargo, el proceso de bagging puede ser computacionalmente eficiente, ya que puede ser implementado de manera paralela o distribuida, con lo cual el consumo de tiempo puede verse reducido considerablemente.

Comparando con la técnica OFP_CLASS$_{\text{LQD}}$

Vamos a comparar la técnica OFP_CLASS$_{\text{LQD}}$ con la técnica propuesta BAGOFP_CLASS utilizando los conjuntos de datos de "Atletismo de alto rendimiento" (Long-4, 100ml-4-I y 100ml-4-P) y los de "Dislexia" (Dyslexic-12, Dyslexic-12-01 y Dyslexic-12-12), [160, 161].

Hemos utilizado una validación cruzada de 10 para los conjuntos de datos Long-4, 100ml-4-I y 100ml-4-P. La Tabla 10.14 muestra los resultados obtenidos por el ensamble FRF$_{\text{LQD}}$ con 6 de sus métodos de combinación al usar las particiones generadas por las técnicas OFP_CLASS$_{\text{LQD}}$ y BAGOFP_CLASS. En esta Tabla 10.14 se muestra el intervalo $[media\_min\_error, media\_max\_error]$ obtenido para cada conjunto de datos de acuerdo al proceso de decisión descrito en el Algoritmo 11. Para cada conjunto de datos, el mejor resultado obtenido con respecto al test se encuentra resaltado en negrita.

Para los conjuntos de datos Dyslexic-12, Dyslexic-12-01 y Dyslexic-12-12, los experimentos han sido repetidos 100 veces mediante bootstrapp con reemplazamiento. Los conjuntos de test lo componen los ejemplos OOB. La Tabla 10.15, muestra los resultados obtenidos cuando ejecutamos el ensamble FRF$_{\text{LQD}}$ con las particiones fuzzy obtenidas por el algoritmo

Tabla 10.14: Resultados comparativos para los conjuntos de datos de atletismo de alto rendimiento de OFP_CLASS$_{LQD}$ y BAGOFP_CLASS

		Conjuntos de datos					
		100ml-4-I		100ml-4-P		Long-4	
	Técnica	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
OFP_CLASS$_{LQD}$ Fuzzy partition	FFR$_{LQDSM1}$	[0.107,0.305]	[0.130,0.323]	[0.043,0.235]	[0.093,0.290]	[0.191,0.484]	[0.083,0.349]
	FRF$_{LQDSM2}$	[0.110,0.306]	[0.150,0.343]	[0.045,0.237]	[0.110,0.307]	[0.165,0.449]	[0.083,0.349]
	FRF$_{LQDMWL1}$	[0.070,0.265]	[0.073,0.267]	[0.032,0.224]	[0.060,0.257]	[0.085,0.364]	**[0.033,0.299]**
	FRF$_{LQDMWL2}$	[0.060,0.254]	[0.113,0.306]	[0.043,0.235]	[0.060,0.257]	[0.111,0.391]	[0.083,0.349]
	FRF$_{LQDMWLT1}$	[0.070,0.267]	[0.073,0.267]	[0.032,0.224]	[0.060,0.257]	[0.085,0.364]	**[0.033,0.299]**
	FRF$_{LQDMWLT2}$	[0.060,0.252]	[0.093,0.286]	[0.038,0.231]	[0.060,0.257]	[0.107,0.386]	[0,083,0.349]
BAGOFP_CLASS Fuzzy partition	FFR$_{LQDSM1}$	[0.024,0.218]	[0.153,0.347]	[0.019,0.214]	[0.060,0.257]	[0.1201,0.369]	[0.083,0.349]
	FRF$_{LQDSM2}$	[0.023,0.215]	[0.093,0.287]	[0.028,0.221]	**[0.040,0.237]**	[0.116,0.365]	[0.083,0.349]
	FRF$_{LQDMWL1}$	[0.019,0.211]	**[0.053,0.247]**	[0.000,0.193]	[0.060,0.257]	[0.1102,0.359]	**[0.033,0.299]**
	FRF$_{LQDMWL2}$	[0.017,0.209]	[0.090,0.283]	[0.000,0.193]	[0.060,0.257]	[0.107,0.356]	[0.083,0.349]
	FRF$_{LQDMWLT1}$	[0.017,0.209]	**[0.053,0.247]**	[0.000,0.193]	**[0.040,0.237]**	[0.106,0.355]	**[0.033,0.299]**
	FRF$_{LQDMWLT2}$	[0.017,0.209]	[0.090,0.283]	[0.000,0.193]	[0.060,0.257]	[0.107,0.356]	[0.083,0.349]

BAGOFP_CLASS y con las particiones fuzzy obtenidas con el algoritmo OFP_CLASS$_{LQD}$ para esos conjuntos de datos. Al igual que en el resto de tablas comparativas, el mejor resultado con respecto al test se encuentra resaltado en negrita.

Tabla 10.15: Comparando los resultados para los conjuntos de datos de dislexia de OFP_CLASS$_{LQD}$ y BAGOFP_CLASS

		Conjuntos de datos					
		Dyslexic-12		Dyslexic-12-01		Dyslexic-12-12	
	Técnica	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
OFP_CLASS$_{LQD}$ Partición Fuzzy	FRF$_{LQDSM1}$	[0.000,0.238]	[0.000,0.398]	[0.022,0.223]	[0.039,0.377]	[0.001,0.263]	[0.035,0.422]
	FRF$_{LQDSM2}$	[0.000,0.228]	[0.000,0.399]	[0.008,0.184]	[0.022,0.332]	[0.009,0.245]	[0.032,0.411]
	FRF$_{LQDMWL1}$	[0.000,0.270]	[0.000,0.406]	[0.017,0.231]	[0.045,0.383]	[0.001,0.273]	[0.019,0.430]
	FRF$_{LQDMWL2}$	[0.000,0.270]	[0.000,0.407]	[0.020,0.241]	[0.056,0.385]	[0.001,0.267]	[0.026,0.406]
	FRF$_{LQDMWLT1}$	[0.000,0.263]	[0.000,0.402]	[0.012,0.216]	[0.038,0.365]	[0.000,0.265]	[0.019,0.427]
	FRF$_{LQDMWLT2}$	[0.000,0.266]	[0.000,0.404]	[0.015,0.221]	[0.049,0.373]	[0.000,0.262]	[0.024,0.422]
BAGOFP_CLASS Partición Fuzzy	FRF$_{LQDSM1}$	[0.000,0.228]	**[0.000,0.390]**	[0.014,0.197]	[0.033,0.361]	[0.001,0.277]	**[0.023,0.390]**
	FRF$_{LQDSM2}$	[0.000,0.227]	[0.000,0.393]	[0.010,0.185]	[0.025,0.332]	[0.001,0.280]	[0.023,0.395]
	FRF$_{LQDMWL1}$	[0.000,0.242]	[0.000,0.404]	[0.008,0.197]	[0.036,0.357]	[0.002,0.283]	[0.023,0.399]
	FRF$_{LQDMWL2}$	[0.000,0.247]	[0.000,0.397]	[0.010,0.185]	[0.025,0.328]	[0.001,0.282]	[0.023,0.400]
	FRF$_{LQDMWLT1}$	[0.000,0.241]	[0.000,0.406]	[0.006,0.192]	**[0.019,0.329]**	[0.000,0.275]	[0.022,0.417]
	FRF$_{LQDMWLT2}$	[0.000,0.244]	[0.000,0.395]	[0.006,0.197]	[0.042,0.350]	[0.002,0.282]	[0.023,0.400]

Como podemos observar en las dos Tablas 10.14 y 10.15, la técnica BAGOFP_CLASS obtiene mejores resultados en la mayoría de los casos (siendo iguales en el resto). Además, BAGOFP_CLASS extrae una mayor cantidad de información, lo que se refleja en una menor amplitud de los intervalos obtenidos.

CAPÍTULO

11

Selección de Atributos

11.1 Introducción

La selección de atributos es uno de los principales temas en el AID y más específicamente en la tarea de clasificación. Se ha demostrado que una selección de atributos adecuada potencia la extracción de conocimiento y crea modelos más interpretables.

Como comentamos en el Capítulo 4, en la literatura se pueden encontrar una variedad de técnicas para llevar a cabo el proceso de selección de atributos, pero la mayoría de estas técnicas asumen que los datos están expresados con valores sin imprecisión y/o incertidumbre. En la actualidad, los investigadores están comenzando a hacer un esfuerzo por extender estas técnicas con el fin de que puedan trabajar con datos con imprecisión y/o incertidumbre, es decir, con LQD.

En este capítulo proponemos una técnica para seleccionar atributos que trabaja dentro del marco de la lógica fuzzy y que es capaz de trabajar con los LQD que se encuentran explícitamente en los conjuntos de datos, [36]. Esta técnica será validada mediante un conjunto de experimentos con distintos conjuntos de datos (con y sin LQD). A partir de esta técnica, se propone una mejora para poder tratar con conjuntos de datos con un número elevado de atributos.

11.2 FRF-fs: Una técnica de selección de atributos

La técnica que proponemos para seleccionar atributos (que denotaremos como FRF-fs), [36], [35] se puede clasificar como una técnica Filtro-Wrapper con búsqueda secuencial hacia delante sobre el subconjunto obtenido por la técnica de filtrado y utilizando el ranking obtenido

de estos atributos. La Figura 11.1 muestra el marco del planteamiento de la propuesta FRF-fs, la cual se desarrolla en las siguientes etapas:

(1) Se aplica la técnica de filtrado: se lleva a cabo un proceso de escalado y de discretización en el conjunto de atributos. Los atributos discretizados, serán el conjunto de atributos preseleccionados inicialmente.

(2) Se realiza un ranking de los atributos preseleccionados utilizando el ensamble FRF.

(3) Se aplica la técnica Wrapper: se utiliza una técnica de clasificación basada en validación cruzada para definir el conjunto final de atributos seleccionados.

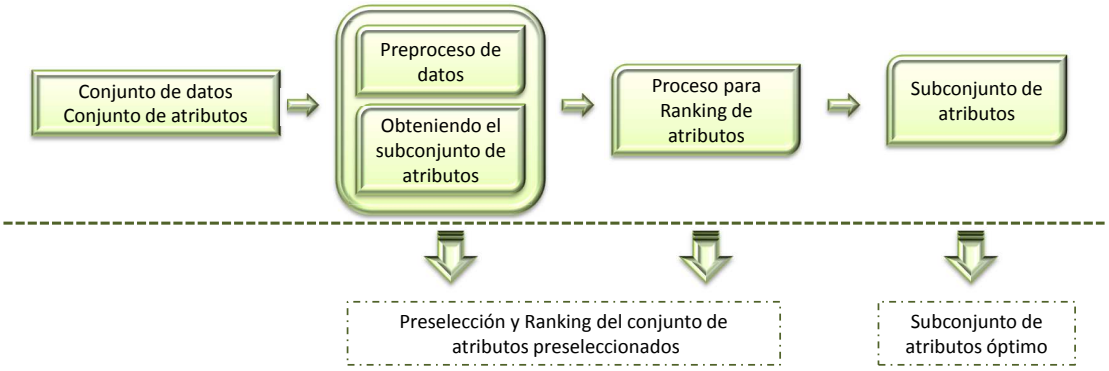


Figura 11.1: Esquema de FRF-fs

Es importante resaltar, tal y como se puede apreciar en la Figura 11.1, que después de cada etapa del proceso de selección de atributos propuesto, el usuario obtiene información útil, tal como el conjunto de atributos preseleccionados, un ranking de importancia de dichos atributos y el conjunto final de atributos seleccionados. Además, hay que tener en cuenta que la propuesta que presentamos es capaz de trabajar explícitamente con LQD ya que todas las técnicas que usa FRF-fs permiten el tratamiento de estos datos. En la Figura 11.2 se puede observar un esquema más detallado de esta propuesta la cual vamos a explicar con más detalle en los siguientes apartados.

11.2.1 Técnica de filtrado para preselección de atributos

11.2.1.1 Preproceso de datos

Inicialmente se preprocesan los datos mediante el escalado y la discretización de los atributos numéricos. La principal ventaja de escalar los datos es evitar que atributos con un rango mayor de sus dominios predominen sobre los atributos de menor rango. Cada atributo es linealmente escalado en el rango $[0, 1]$ mediante $v' = \frac{v - min_a}{max_a - min_a}$, donde v es el valor original, v' es el valor escalado, y, max_a y min_a son los límites máximos y mínimos del atributo a respectivamente.

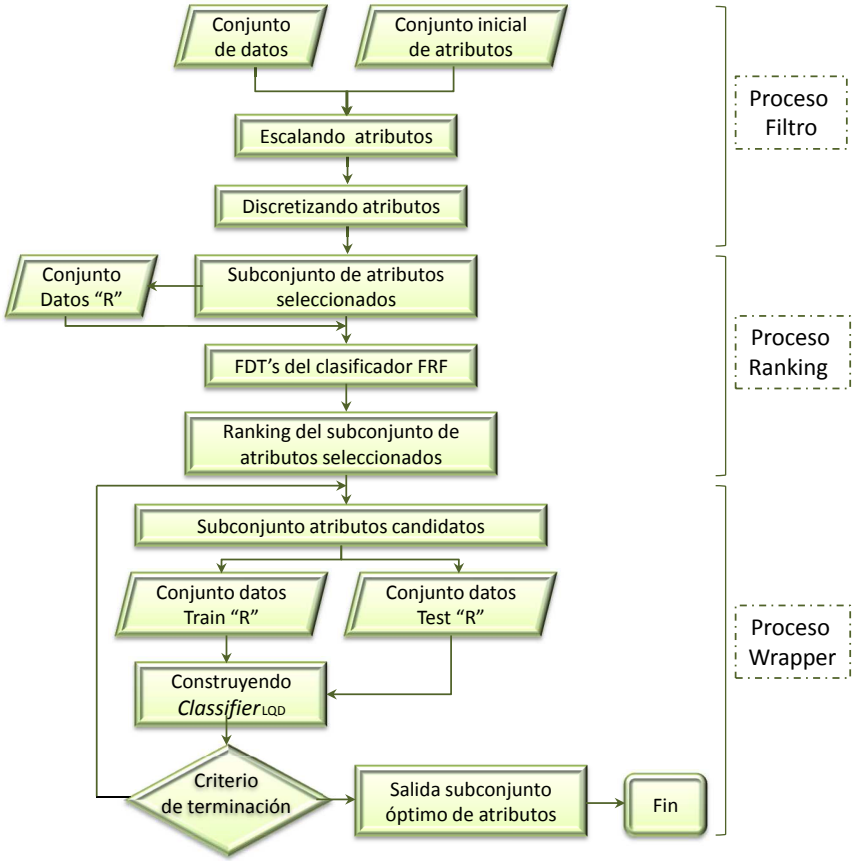


Figura 11.2: Etapas detallas de FRF-fs

Para llevar a cabo la discretización utilizamos la técnica OFP_CLASS$_{\text{LQD}}$, [43], que hemos presentado en el Capítulo 10. La técnica OFP_CLASS$_{\text{LQD}}$ obtiene una discretización fuzzy de los atributos numéricos. Utilizamos esta técnica de discretización ya que permite trabajar también con LQD directamente pero cualquier otra técnica de la literatura podría ser utilizada para llevar a cabo la discretización.

11.2.1.2 **Preselección de atributos**

Para las etapas (1), (2) y (3) del proceso de selección de atributos utilizamos el ensamble FRF$_{\text{LQD}}$ y el árbol FDT$_{\text{LQD}}$ como técnicas de aprendizaje las cuales requieren que los atributos numéricos estén discretizados. Hay que tener en cuenta que durante el proceso de discretización puede que algunos atributos sean discretizados en más de una partición y otros solamente se discretizan en una única partición, puesto que la técnica considera que el atributo es irrelevante. Por lo tanto, estos atributos puede ser eliminados sin que esto afecte al poder de discriminación en el conjunto de datos original. Tras eliminar los atributos no discretizados, obtenemos un primer subconjunto de atributos, es decir, obtenemos una preselección de éstos. Para continuar

con el proceso de selección, transformamos el conjunto de datos inicial en otro conjunto de datos que solamente contiene los atributos preseleccionados.

11.2.2 **Ranking de importancia de los atributos**

Con el nuevo conjunto de datos de atributos preseleccionados, proponemos una medida para calcula un ranking de importancia de tales atributos. Para el cálculo de esta medida utilizamos la información generada por el ensamble FRF_{LQD} utilizando este nuevo conjunto de datos. Para ello aplicamos FRF_{LQD} al nuevo conjunto de datos y se obtiene el ensamble. A partir del ensamble, recopilamos toda la información que proporciona cada árbol de decisión fuzzy FDT_{LQD}. En el Algoritmo 20 se describe cómo se recopila toda esta información que proporciona cada árbol de decisión fuzzy FDT_{LQD}.

Algoritmo 20 Obtener información del ensamble FRF_{LQD}

INFFRF(**in:** E, $Particiones\text{-}Fuzzy$, T; **out:** INF)
begin

1. Construir un ensamble Fuzzy Random Forest
2. **for** cada $\text{FDT}_{\text{LQD}t}$, $t = 1$ hasta T del ensamble FRF_{LQD} **do**
   Guardar el atributo a seleccionado para dividir cada nodo N, la ganancia de información del nodo, IG_{Na}, y la profundidad de este nodo P_{Na} en INF_a.
   Obtener la precisión de clasificación Acc_t del $\text{FDT}_{\text{LQD}t}$ con su correspondiente conjunto de datos OOB_t.
   end for

end

Más específicamente, la información obtenida de cada $\text{FDT}_{\text{LQD}t}$ es la siguiente:

- La ganancia de información del nodo N para el atributo a (IG_{Na}) donde el atributo a es el seleccionado como mejor para dividir este nodo N.
- Nivel de profundidad del nodo N (P_{Na}) donde el atributo a es el seleccionado como mejor para dividir el nodo N.
- Precisión de clasificación Acc_t de cada $\text{FDT}_{\text{LQD}t}$ cuando clasifica el conjunto de datos OOB_t, siendo OOB_t el conjunto de ejemplos que no han sido utilizados en la construcción del $\text{FDT}_{\text{LQD}t}$.

Esta información la utilizamos para calcular la medida de importancia de cada atributo. El Algoritmo 21 detalla cómo se combina esta información para obtener la medida de importancia, donde p_i es el peso asignado al atributo a que estará establecido en función del lugar donde aparezca el atributo en el $\text{FDT}_{\text{LQD}t}$. Después de combinar la información, la salida de algoritmo es una matriz (IMP) donde se ordenan para cada $\text{FDT}_{\text{LQD}t}$ y para cada atributo a el valor de importancia obtenido.

Algoritmo 21 Combinación de la información INF

IMPFRF(in: INF, T; **out:** IMP)
begin

for cada $\text{FDT}_{\text{LQD}t}$, $t = 1$ hasta T **do**
for cada atributo a=1 hasta $|A|$ **do**
for todos los nodos N donde el atributo a aparece **do**
if $P_{Na} = i$ **then**
$IMP_{ta} = IMP_{ta} + p_i \cdot IG_{Na}$ con $i \geq 0$ y $P_{rootnode} = 0$
end if
end for
for cada atributo a=1 hasta $|A|$ **do**
$IMP_{ta} = \frac{IMP_{ta} - min(IMP_t)}{max(IMP_t) - min(IMP_t)}$
$IMP_{ta} = IMP_{ta} \cdot Acc_t$
end for
El vector IMP_t se ordena en orden descendiente, $IMP_{t_{\sigma_t}}$, donde σ_t es la permutación obtenida cuando ordenamos IMP_t
end for
end for

end

La idea subyacente de la medida de importancia de cada atributo es hacer uso de los atributos seleccionados en cada FDT_{LQD} y de los nodos de decisión construidos, junto con el comportamiento del FDT_{LQD} al clasificar su OOB. La importancia de un atributo viene determinado por el nivel de profundidad en un FDT_{LQD} donde ha sido seleccionado. Así un atributo que aparece en los primeros nodos es más importante que un atributo que aparece en nodos más profundos. Además un FDT_{LQD} que tiene una precisión en clasificación mayor es mejor que otro al clasificar el conjunto OOB. Para tomar la decisión final, y obtener un valor de importancia para cada atributo, agregamos toda la información obtenida. Como resultado del Algoritmo 21 se obtiene para cada FDT_{LQD} del ensamble FRF_{LQD} un ranking de importancia de los atributos. Más específicamente se obtiene T rankings de importancia para cada atributo a. Por ello, y para obtener un único ranking definitivo, aplicamos un operador OWA.

Los operadores OWA (Ordered Weighted Averaging) fueron introducidos por Yager en 1988, [207]. Estos operadores son conocidos como operadores de compensación. Son operadores de agregación de información numérica la cual se considera que tiene un orden.

Sea $Y = \{y_1, \ldots, y_n\}$ *, con* $y_i \in [0, 1]$*, el conjunto de evaluaciones que queremos agregar y sea* $W = \{w_1, \ldots, w_n\}$ *su vector de pesos asociados, tal que* $w_i \in [0, 1]$*, con* $1 \leq i \leq n$*, y* $\sum_{i=1}^{n} w_i = 1$*. El operador OWA O, se define como:*

$$O(y_1, \ldots, y_n) = \sum_{j=1}^{n} w_j \cdot b_j$$

donde b_j *es el j-ésimo valor en el conjunto* Y *(*$B = \{b_1, \ldots, b_n\}$ *tal que* $b_i \geq b_j$*, si* $i < j$*).*

Nosotros disponemos de T conjuntos ordenados. Dado el vector de pesos W, el vector $RANK$ (ranking del subconjunto de atributos preseleccionados) se calcula como:

$$OWAIMP_t = W \cdot IMP_{t_{\sigma_t}}, \text{ para } t = 1, \ldots, T$$

$$RANK_a = \sum_{t=1}^{T} OWAIMP_{t\sigma_t(a)}, \text{ para } a = 1, \ldots, |A|$$

El vector $RANK$ se ordena se manera descendiente: $RANK_\sigma$.

11.2.3 **Wrapper para la obtención del subconjunto final de atributos**

Una vez que se obtiene el ranking de atributos preseleccionados, $RANK_\sigma$, debemos de encontrar el subconjunto de atributos óptimos. Para ello, se obtienen distintos subconjuntos de atributos usando el ranking $RANK_\sigma$. Los subconjuntos de atributos obtenidos por este proceso se evalúan usando alguna técnica que soporte LQD y mediante validación cruzada. A la técnica utilizada la llamaremos $\text{Classifier}_{\text{LQD}}$. En el Algoritmo 22 se muestra detallado el proceso de la técnica wrapper.

A partir del conjunto ordenado de atributos preseleccionados, se construye una secuencia de ascendente de modelos con $\text{Classifier}_{\text{LQD}}$ añadiendo y validando los atributos de uno en uno. La introducción secuencial de atributos se lleva a cabo en dos fases:

- En la primera fase, se construyen dos subconjuntos: El subconjunto CF_{base} y el subconjunto CF_{comp}. Un atributo f_i se añade al subconjunto CF_{base} solamente si el decrecimiento del ratio de error usando los atributos de $CF_{base} \cup \{f_i\}$ excede el umbral δ_1. La idea es que la disminución del error al añadir el atributo f_i sea lo suficientemente significativo para que este atributo pertenezca al subconjunto CF_{base}. Si cuando se clasifica utilizando el subconjunto $CF_{base} \cup \{f_i\}$ la disminución del error es más pequeña que el umbral δ_1 o el error se incrementa pero este incremento es menor que δ_2, el atributo f_i pasa a formar parte del subconjunto CF_{comp}.
- La segunda fase empieza con los dos subconjuntos creados anteriormente CF_{base} y CF_{comp}. Se fija el subconjunto CF_{base} y se añaden subgrupos de atributos desde CF_{comp} construyendo varios modelos con $\text{Classifier}_{\text{LQD}}$. Esta fase determina el conjunto final de atributos con el error mínimo de acuerdo a las condiciones que se reflejan en el paso 5. del Algoritmo 22. Estas condiciones se interpretan como "seleccionar un subconjunto que decremente el error en una cantidad superior al umbral δ_3 o decremente el error en una cantidad menor que el error δ_3 pero usando un menor número de atributos".

Tras detallar el algoritmo FRF-fs, vamos a llevar a cabo una serie de experimentos con el fin de poder evaluar el rendimiento del algoritmo.

Algoritmo 22 Técnica Wrapper

FRF-fs(in: E, CF (conjunto de atributos), $RANK_\sigma$; **out:** CF_{opt} (conjunto de atributos seleccionados))
begin

1. Inicializar:

   1.1 $CF_{comp} = \{\}$ y $CF_{base} = \{f_1\}$ donde f_1 es el primer atributo del ranking $RANK_\sigma$
   1.2 ERR_1 =Classifier$_{\text{LQD}}(E, CF_{base})$ usando validación cruzada, $BE = ERR_1$

2. **for** cada $f_i \in CF$, con $i = 2, \ldots, |CF|$ en el orden determinado por $RANK_\sigma$ **do**
   ERR_B =Classifier$_{\text{LQD}}(E, CF_{base} \cup \{f_i\})$ usando validación cruzada
   if $((BE - ERR_B) > \delta_1)$ **then** $CF_{base} = CF_{base} \cup \{f_i\}$
   else
   if $((ERR_B - BE) < \delta_2)$ **then** $CF_{comp} = CF_{comp} \cup \{f_i\}$
   end if
   end if
   end for
3. **for** cada $f_i \in CF$, con $i = 2, \ldots, |CF|$ en el orden determinado por $RANK_\sigma$ **do**
   ERR_B =Classifier$_{\text{LQD}}(E, CF_{base} \cup \{f_i\})$ usando validación cruzada
   if $((BE - ERR_B) > \delta_1)$ **then** $CF_{base} = CF_{base} \cup \{f_i\}$
   else
   if $((ERR_B - BE) < \delta_2)$ **then** $CF_{comp} = CF_{comp} \cup \{f_i\}$
   end if
   end if
   end for
4. $CF_{aux} = CF_{base}$
5. **for** cada $f_i \in CF_{comp}$, con $i = 1, \ldots, |CF_{comp}|$ en el orden determinado por $RANK_\sigma$ **do**
   $B = CF_{base},\ STOP = 0,\ j = i$
   while $(STOP < \delta_2)$ y $(j \leq |CF_{comp}|)$) **do**
   $B = B \cup \{f_j\}$
   ERR_B =Classifier$_{\text{LQD}}(D, B)$ usando validación cruzada
   if $((BE - ERR_B) \geq \delta_3)$ ó $(0 \leq (BE - ERR_B) < \delta_3$ y $|CF_{aux}| > |B|))$ **then**
   $CF_{aux} = B, BE = ERR_B$
   else
   if $((ERR_B - BE) > \delta_2)$ **then** $STOP = (ERR_B - BE)$
   end if
   end if
   $j = j + 1$
   end while
   end for
6. $CF_{opt} = CF_{aux}$

end

11.3 Resultados experimentales

En esta sección vamos a detallar los elementos y el proceso seguido para llevar a cabo una serie de experimentos con el fin de validar nuestra propuesta. En los experimentos que vamos a realizar vamos a comparar con otros trabajos de la literatura que tienen como base la técnica

Random Forest y además vamos a realizar otros experimentos trabajando con conjuntos de datos que contienen LQD para evaluar el comportamiento de la técnica frente a este tipo de datos. Para ello, antes de llevar a cabo los experimentos vamos a presentar los conjuntos de datos utilizados en los mismos, después describiremos las técnicas de la literatura utilizadas para la comparación de resultados, luego describimos la configuración de los parámetros para llevar a cabo los experimentos y finalmente presentaremos las medidas empleadas para evaluar las propuestas y los tests estadísticos utilizados para analizar los resultados.

11.3.1 **Descripción de los conjuntos de datos**

Vamos a llevar a cabo dos tipos de experimentos: Para el primero hemos seleccionado 10 conjuntos de datos de alta dimensión [65] (conjuntos de datos microarray) para llevar a cabo un estudio del comportamiento de nuestra propuesta cuando el número de atributos se incrementa en relación al número de ejemplos y clases.

La aplicación de técnicas a estos tipos de conjuntos de datos tratan de discriminar las muestras de tejidos cáncerosos de acuerdo a sus niveles de expresión génica, identificar un pequeño subconjunto de genes que son responsables de la enfermedad y descubrir fármacos potenciales para la misma, [85]. Es decir, mejorar la tecnología permitirá un mejor entendimiento de los conceptos y conocimientos sobre los procesos celulares relacionados con el cáncer. Así, dado que la determinación del tipo de cáncer y su etapa es crucial para la asignación de un tratamiento apropiado, [88], un objetivo principal del análisis de datos de expresión génica es la identificación de conjuntos de genes que pueden servir como plataformas de clasificación o diagnóstico. Dado que los datos obtenidos de los estudios de expresión de genes relacionados con el cáncer suelen consistir en mediciones del nivel de expresión de miles de genes [6], se requiere de metodologías del AID que ayuden de manera eficiente a la extracción de información biológica relevante a partir de grandes conjuntos de datos de expresión génica, [65, 88, 156].

La Tabla 11.1 muestra una descripción de estos conjuntos de datos.

Tabla 11.1: Conjuntos de datos de microarrays

Conjuntos de datos	**Abbr**	**|E|**	**|A|**	**|C|**	**Conjuntos de datos**	**Abbr**	**|E|**	**|A|**	**|C|**
Leukaemia	LEU	38	3051	2	Brain	BRA	42	5597	5
Breast 2 cl.	BR2	77	4869	2	Colon	COL	62	2000	2
Breast 3 cl.	BR3	95	4869	3	Lymphoma	LYM	62	4026	3
NCI 60	NCI	61	5244	8	Prostate	PRO	102	6033	2
Adenocarcinoma	ADE	76	9868	2	Srbct	SRB	63	2308	4

Para el segundo experimento hemos utilizado 14 conjuntos de datos del repositorio de conjuntos de datos de UCI, [77]. En la Tabla 11.2 se muestran los atributos de estos 14 conjuntos

de datos. Estos últimos conjuntos de datos han sido modificados para introducirles diferentes tipos de imperfección. Para ello se ha utilizado la herramienta de software *"NIP imperfection processor"*, [39], que más adelante detallaremos. Hemos llevado a cabo tal modificación para realizar un estudio acerca del comportamiento de nuestra propuesta FRF-fs cuando los conjuntos de datos contienen LQD.

Tabla 11.2: Conjuntos de datos del repositorio UCI

Conjuntos de datos	Abbr	|E|	|A|	|C|	?	Conjuntos de datos	Abbr	|E|	|A|	|C|	?
Australian credit	AUS	690	14 (6-8)	2	Y	Pima Indian Diabetes	PIM	768	8 (8-0)	2	Y
Wis. Br. Cancer (org)	BCW	699	9 (9-0)	2	Y	Sonar	SON	208	60 (60-0)	2	N
German (credit card)	GER	1000	24 (24-0)	2	N	SPECTF heart	SPE	267	44 (44-0)	2	N
Glass identificarion	GLA	214	9 (9-0)	7	N	Vehicle	VEH	946	18 (18-0)	4	N
Statlog Heart	HEA	270	13 (13-0)	2	N	Wis. Diag. Br. Cancer	WDC	569	31 (31-0)	2	N
Ionosphere	ION	351	34 (34-0)	2	N	Wine	WIN	178	13 (13-0)	3	N
Iris Plant	IRP	150	4 (4-0)	3	N	Zoo	ZOO	101	16 (15-1)	7	N

En las Tablas 11.1 y 11.2 se muestra para cada conjunto de datos, el número de ejemplos ($|E|$), el número de atributos ($|A|$) indicando entre paréntesis, si los atributos son numéricos o nominales, el número de clases ($|C|$) y la columna "Abbr" indica la abreviación del conjunto de datos usando en los experimentos. La columna ? de la Tabla 11.2 muestra si los conjuntos de datos contienen valores missing.

11.3.2 Técnicas de la literatura utilizados en la comparación de los resultados

En el primer experimento, FRF-fs se compara con varios enfoques para la selección de atributos de la literatura, especialmente con aquellos que se basan en el ensamble Random Forest. Como se menciona en [65], el ensamble Random Forest, desarrollado por Leo Breiman [26] tiene varias características que lo hacen ideal para trabajar con conjuntos de datos de microarrays: a) Puede usarse cuando hay más atributos que ejemplos; b) Puede utilizarse tanto para problemas con clases binarias como para problemas multiclase; c) Tiene un buen rendimiento en la predicción incluso cuando los atributos contienen ruido, es decir, muestra una fuerte robustez con respecto a grandes conjuntos de atributos; d) No se sobreajusta a los datos; e) Puede manejar tanto atributos nominales como numéricos; f) Incorpora las interacciones entre los atributos de predicción; g) La salida es invariante a las transformaciones monótonas de los atributos; h) Devuelve la medida de la importancia de los atributos; i) No necesita de un ajuste fino de parámetros para lograr un excelente rendimiento.

Los enfoques de la literatura que utilizaremos son: un Random Forest como medida de precisión de conjuntos de datos sin selección de atributos (RF), las técnicas basadas en Random

Forest propuestas en [65] (RF.du) y en [82] (RF.ge), y también con la técnica, no basada en Random Forest, el vecino más cercano para selección de atributos (NN.vs).

En [65] se propone una estrategia basada en la eliminación de atributos recursivamente. Más precisamente, primero se calcula la importancia de los atributos con Random Forest. Después de este paso, se elimina el 20 % de los atributos que tienen menos importancia y construyen un nuevo ensamble Random Forest con los atributos restantes. Finalmente, se selecciona el conjunto de atributos que tenga el mejor ratio de error OOB del ensamble Random Forest, definiendo este error como

$$errOOB = \frac{1}{n} Card\{i \in \{1, \ldots, n\} / y_i \neq \hat{y}_i\}$$

donde $\hat{y}_i$ es la etiqueta más frecuente predicha por el ensamble de árboles donde (x_i, y_i) está en su muestra OOB. La proporción de atributos eliminados es un parámetro arbitrario de su técnica que no depende de los datos.

En [82] se menciona que "... distinguimos dos objetivos en la selección de atributos: 1) Encontrar un conjunto de atributos altamente relacionado con el atributo clase tomando como propósito la interpretación; 2) Encontrar un conjunto pequeño de atributos lo suficientemente buenos para una predicción del atributo clase". La primera de ellas es para destacar todas los atributos importantes, incluso con alta redundancia, para el propósito de la interpretación, y la segunda es para encontrar un conjunto de atributos importantes para la predicción. Como se menciona en el artículo, se guían por una situación típica para emparejar dos características. La primera es la alta dimensionalidad y la segunda es la presencia de grupos de predictores altamente correlacionados. Además también abordan específicamente los trabajos [65] y [83].

En el segundo experimento, FRF-fs va a ser comparado con un ensamble FRF_{LQD} con y sin selección de atributos debido a la escasez de trabajos que trabajen directamente con LQD.

11.3.3 **Técnicas para la comparación y validación de los resultados**

Para analizar los resultados obtenidos en este estudio vamos a emplear las siguientes medidas de rendimiento:

Para comparar los resultados entre las diferentes técnicas utilizadas en el primer experimento con conjuntos de datos de microarray, empleamos las técnicas de estimación utilizadas en [65]. Los diferentes conjuntos de atributos seleccionados obtenidos por todas las técnicas son evaluados estimando la tasa de error de predicción utilizando la técnica".632+bootstrap" con 200 muestras bootstrap. La técnica .632+bootstrap utiliza la media ponderada del error de resubstitución y el error en los ejemplos no usados para el entrenamiento. Esta media es ponderada por una cantidad que refleja la cantidad de sobreajuste. Denotamos por "X"$_{0{,}632+bs}$, la técnica 0.632+bootstrap usando los atributos seleccionados por la técnica "X".

Con el fin de comparar los resultados en el segundo experimento con conjuntos de datos con LQD, vamos a utilizar la precisión como medida, siendo esta precisión el número de aciertos con respecto al total de ejemplos clasificados. Más específicamente, la precisión media obtenida como la media de las precisiones de una validación cruzada de tamaño 5 repetida 3 veces y usando como clasificador un ensamble FRF_{LQD}. Con el ensamble pretendemos evaluar la calidad de los atributos seleccionados por la técnica FRF-fs cuando los conjuntos de datos contienen explícitamente LQD. Vamos a denotar como $\text{FRF-fs}_{3\times 5\text{-FRF}_{\text{LQD}}}$ la precisión al clasificar con el clasificador FRF_{LQD} los atributos seleccionados por la técnica FRF-fs.

Además, para este último experimento, mostramos la ratio de reducción (red. rate) en la selección de atributos. Esta ratio de reducción se calcula como $1 - \frac{\#fe}{|A|}$ donde $\#fe$ es el número de atributos seleccionados.

Para completar el estudio experimental, llevamos a cabo en cada experimento un análisis de los resultados utilizando técnicas estadísticas. Siguiendo la metodología propuesta por [79], los test estadísticos que utilizamos son no paramétricos. Utilizamos el test de rangos de signos de Wilcoxon para comparar dos técnicas (análogo a los t-test pareados en los procedimientos estadísticos paramétricos). Así este test se lleva a cabo por pares con el objetivo de detectar diferencias significativas entre dos muestras, es decir, entre el comportamiento de dos técnicas. Para llevar a cabo los test estadísticos hemos utilizado la herramienta R, [101].

11.3.4 **Configuración de los experimentos**

Para el primer experimento los parámetros utilizados son: para la técnica RF.du, un ensamble con 2000 árboles de decisión, la fracción $dropped = 0{,}2$ y $mtryfactor = 1$, y para la técnica RF.ge se utilizan 50 ensambles con 2000 árboles de decisión y $mtry = |A|/2$ siendo $|A|$ número de atributos. Para la técnica Random Forest los parámetros utilizados son mtry=$\sqrt{|Attr|}$, 5000 árboles de decisión, y el número de ejemplos en un nodo para que sea hoja es 1.

Los parámetros utilizados en FRF-fs tanto para el primer experimento como para el segundo son:

- Parámetros para el ensamble:
  - Número de FDT_{LQD}'s 500.
  - Selección de atributos aleatorios en cada nodo del FDT_{LQD} desde el conjunto disponible: $\sqrt{|A|}$
- Vector para combinar información INF (Algoritmo 21): $p = (1, \frac{6}{7}, \frac{2}{3}, \frac{2}{4}, \frac{2}{5}, \frac{2}{P_{Na}+1}, \ldots)$ con P_{Na} profundidad del nodo N el cual contiene el atributo a.

- Vector de pesos normalizados para el cálculo de $OWAIMP$: $W = (1, \frac{1}{2}, \ldots, \frac{1}{|A|})$ siendo $|A|$ el número de atributos.
- La técnica Wrapper:
  - Se utiliza una validación cruzada de 3 para evaluar el rendimiento de cada grupo de atributos seleccionados para ser analizado.
  - La técnica Classifier$_{\text{LQD}}$, es un ensamble FRF$_{\text{LQD}}$ con 200 FDT$_{\text{LQD}}$'s.
  - Umbrales: $\delta_1 = \delta_3$=0.5, δ_2 =0.05.

11.3.5 **Resultados y análisis del primer experimento**

En esta subsección, analizamos la precisión en clasificación cuando llevamos a cabo la selección de atributos, comparando los resultados obtenidos por FRF-fs con los obtenidos por otras técnicas comentadas en la subsección 11.3.3. En concreto comparamos el rendimiento predictivo de FRF$_{\text{LQD}}$, [48], [47], con las siguientes técnicas: a) Random Forest sin selección de atributos (RF), b) una técnica que lleva a cabo la selección de atributos basándose en la técnica vecinos más cercanos, (NN.vs), es el único que no está basado en random forest y mostramos los mejores resultados del mismo según [65]; y c) dos técnicas que llevan a cabo la selección de atributos basándose en Random Forest RF.du [65] y RF.ge [82].

Para este experimento, tal como se ha comentado, hacemos uso de los conjuntos de datos presentados en la Tabla 11.1. Para el cálculo de la ratio de error (o ratio de precisión) empleamos la técnica .632+bootstrap.

En la Tabla 11.3 se muestran los resultados medios obtenidos y en la Figura 11.3 se muestran los mismos resultados gráficamente. % acc es el porcentaje medio de precisión en clasificación obtenido con la técnica .632+bootstrap y $\#fe$ es el número de atributos seleccionados.

Como puede observarse, las dos mejores técnicas para seleccionar atributos sobre los conjuntos de datos de microarrays son RF.ge y FRF-fs. Además, hay que tener en cuenta que la precisión media de las dos técnicas es mejor que el obtenido por la técnica Random Forest sin selección de atributos. La Tabla 11.4 muestra los resultados específicos de estas dos técnicas, mostrando la media de precisión con su desviación estándar, el número de atributos seleccionados y el p-value obtenido al comparar estos resultados con el test de signos de Wilcoxon, tomando como hipótesis nula que las técnicas tienen igual comportamiento y como hipótesis alternativa que las técnicas tienen un comportamiento diferente.

La Figura 11.4 muestra los resultados de estas dos técnicas junto con la técnica RF.du (se = 1), la cual es la de mejor comportamiento de acuerdo a [65].

Tabla 11.3: Resultados obtenidos por RF, NN.vs, FR.du, FR.ge y FRF-fs

	RF$_{0.632+bs}$ Unselect	**NN.vs**$_{0.632+bs}$		**RF.du**$_{0.632+bs}$ s.e.=0		s.e.=1		**RF.ge**$_{0.632+bs}$		**FRF-fs**$_{0.632+bs}$	
	% acc.	% acc.	#fe	% acc.	#fe	% acc.	#fe	% acc.	#fe	% acc.	#fe
LEU	94.90	94.40	23	91.30	2	92.50	2	99.99	1	99.99	1
BR2	65.80	66.30	23	66.30	9	66.80	14	90.38	8	86.56	7
BR3	64.90	57.60	45	65.40	14	63.60	7	80.73	8	81.77	6
NCI	74.80	76.30	880	67.30	60	64.70	30	91.19	26	89.55	24
ADE	87.50	81.90	73	81.50	3	79.30	3	95.90	6	95.97	9
BRA	84.60	80.60	158	78.40	14	78.40	14	98.08	9	96.88	13
COL	87.30	84.20	9	84.10	5	82.30	3	91.70	8	92.58	8
LYM	99.10	96.00	15	95.30	14	95.80	12	99.88	12	99.97	6
PRO	92.30	91.90	6	93.90	5	93.60	3	96.07	6	96.41	8
SRB	97.90	96.90	17	96.10	18	96.20	18	98.01	9	98.32	9
Media	84.91	82.61	124.9	81.96	14.4	81.32	9.6	94.19	9.3	93.80	9.1

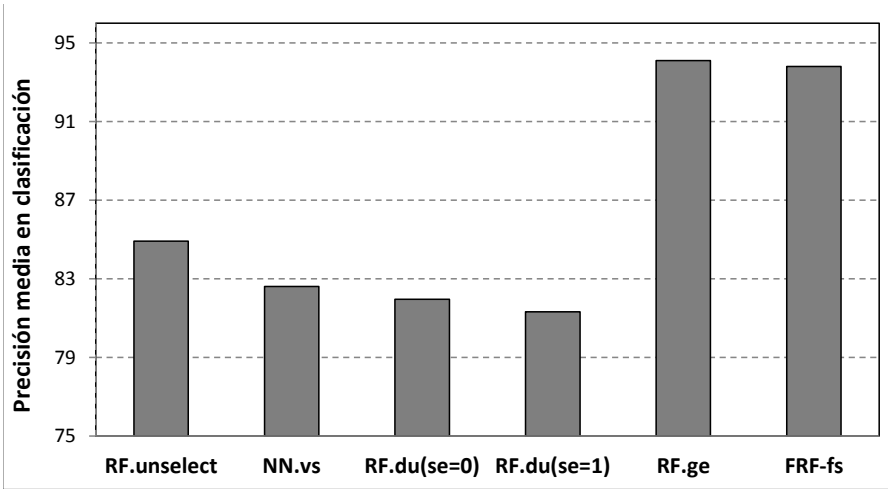


Figure 11.3: Precisión media de las diferentes técnicas

Es importante tener en cuenta que estos resultados muestran que estas dos técnicas, RF.ge y FRF-fs, tienen un buen rendimiento predictivo. Analizando la Tabla 11.4, y observando los diferentes p-values con α =0.05, podemos concluir:

a) Para los conjuntos de datos LEU y ADE, aceptamos la hipótesis nula, por lo tanto no hay diferencias significativas entre los resultados obtenidos por las dos técnicas.

b) Para los conjuntos de datos BR2, NCI y BRA, rechazamos la hipótesis nula, por lo tanto aceptamos la hipótesis alternativa indicando que hay diferencias entre las técnicas. Para analizar más en detalle qué técnica es mejor, realizamos de nuevo el test estadístico pero

Tabla 11.4: Analizando los resultados obtenidos por RF.ge y FRF-fs

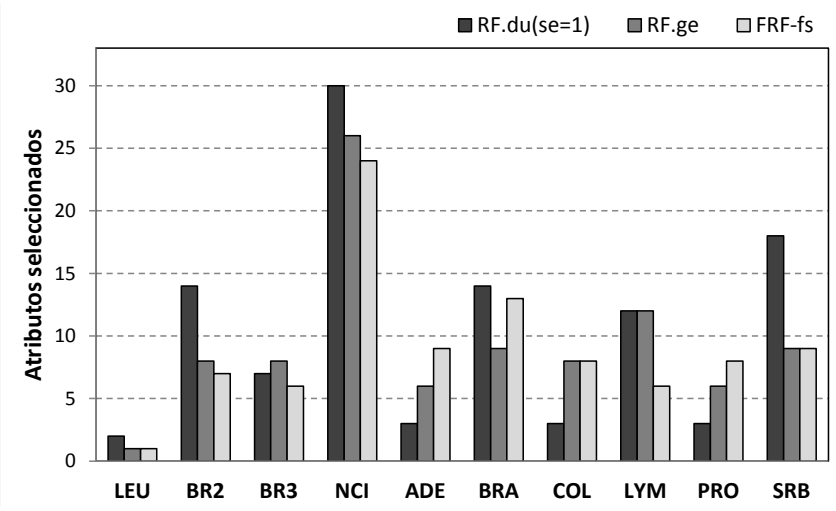
	RF.ge$_{0.632+bs}$		**FRF-fs**$_{0.632+bs}$		
	% acc.	#fe	% acc.	#fe	**p-value**
LEU	$99.99_{0.02}$	1	$99.99_{0.02}$	1	0.17360
BR2	$\mathbf{90.38}_{0.24}$	8	$86.56_{0.12}$	7	0.00391
BR3	$80.73_{0.16}$	8	$\mathbf{81.77}_{0.31}$	6	0.00391
NCI	$\mathbf{91.19}_{0.39}$	26	$89.55_{0.14}$	24	0.00391
ADE	$95.90_{0.20}$	6	$95.97_{0.05}$	9	0.49610
BRA	$\mathbf{98.08}_{0.25}$	9	$96.88_{0.07}$	13	0.00391
COL	$91.70_{0.18}$	8	$\mathbf{92.58}_{0.14}$	8	0.00391
LYM	$99.88_{0.05}$	12	$\mathbf{99.97}_{0.02}$	6	0.00391
PRO	$96.07_{0.07}$	6	$\mathbf{96.41}_{0.04}$	8	0.00391
SRB	$98.01_{0.06}$	9	$\mathbf{98.32}_{0.09}$	9	0.00391
media	$94.19_{0.16}$	9.3	$93.80_{0.10}$	9.1	0.49040

esta vez tomamos como hipótesis nula que no hay diferencias significativas entre las técnicas y como hipótesis alternativa que la técnica RF.ge es mejor que la técnica FRF-fs. El p-value obtenido para los tres conjuntos de datos es 0.00195, por lo tanto con un valor de $\alpha = 0.05$, rechazamos la hipótesis nula y aceptamos la hipótesis alternativa, es decir que para los conjuntos de datos BR2, NCI y BRA la técnica que obtiene un mejor comportamiento es RF.ge, además, es la técnica que consigue la mejor media.

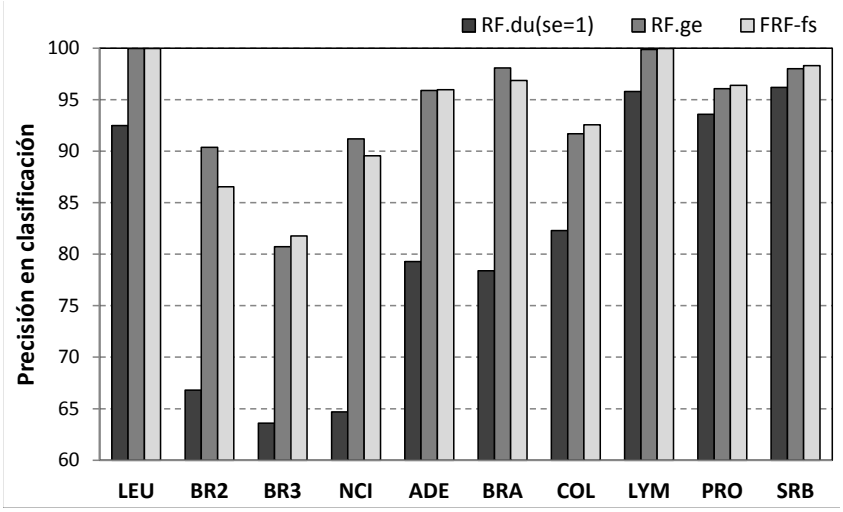
c) Para los conjuntos de datos BR3, ADE, COL, LYM, PRO y SRB el análisis de los p-value indica que hay diferencias significativas, puesto que rechazamos la hipótesis nula. De nuevo vamos a realizar un nuevo test estadístico, tomando como hipótesis nula que no hay diferencias significativas entre las dos técnicas y como hipótesis alternativa que la técnica RF-ge es mejor que la técnica FRF-fs. El p-value obtenido para todos los conjuntos de datos es 1.0, por lo tanto con un valor de $\alpha = 0.05$, deberíamos de aceptar la hipótesis nula porque $1.0 > \alpha$, pero ya hemos demostrado estadísticamente que hay diferentes significativas entre las técnicas, por lo que la conclusión es que la técnica FRF-fs es mejor que la técnica RF-ge.

d) Por último globalmente, después del análisis de todos los resultados obtenidos para los diferentes conjuntos de datos, podemos concluir que no hay diferencias significativas entre los resultados de las dos técnicas al aceptar la hipótesis nula. Además, ambas técnicas obtienen una reducción de atributos similar.

También es necesario destacar el hecho de que la reducción de atributos en los diferentes conjuntos de datos es muy significativa en todos los casos, obteniendo incluso mejor rendi-



a)



b)

Figura 11.4: a) Número de atributos seleccionados; b) Precisión de las tres mejores técnicas usando los atributos seleccionados

miento. Este efecto sobre los conjuntos de datos es muy importante por ejemplo para el uso futuro en diagnóstico clínico. Además, podemos concluir que la técnica FRF-fs es competitiva comparada con las técnicas basadas en Random Forest y además esta técnica es capaz de seleccionar atributos desde conjuntos de datos que contienen LQD. Esta capacidad se muestra en el siguiente experimento.

11.3.6 **Resultados y análisis del segundo experimento**

Este experimento está diseñado para evaluar el rendimiento de la técnica propuesta cuando se aplica a conjuntos de datos que contienen LQD. Los conjuntos de datos utilizados en este

experimento están resumidos en la Tabla 11.2. A estos conjuntos de datos les hemos añadido explícitamente un 10 % de valores fuzzy y un 10 % de valores intervalares. Estos porcentajes no afectan a la clase. Además algunos de estos conjuntos de datos contienen valores missing.

El rendimiento del subconjunto de atributos seleccionados se evalúa con el ensamble FRF_{LQD}. En las diferentes tablas que mostramos a continuación se indica el porcentaje medio de precisión en clasificación (media y desviación estándar) utilizando el subconjunto de atributos seleccionados y una validación cruzada de tamaño 5 repetida 3 veces.

11.3.6.1 **Selección de atributos en conjuntos de datos con valores fuzzy**

Los resultados de precisión se muestran en la Tabla 11.5 y gráficamente en la Figura 11.5. Training y test son los porcentajes medios de precisión (media y desviación estándar) para los datos training y test respectivamente. Este valor de precisión es obtenido mediante un ensamble FRF_{LQD} con 500 árboles con los conjuntos de datos sin seleccionar atributos (columna $Unselec_{3\times5\text{-}FRF_{LQD}}$ de la Tabla 11.5) y con los conjuntos de datos habiendo seleccionado atributos con FRF-fs (columna FRF-$fs_{3\times5\text{-}FRF_{LQD}}$ de la Tabla11.5). Además, se muestran los p-values obtenidos cuando comparamos los resultados con el test de Wilcoxon.

Tabla 11.5: Precisión en los conjuntos de datos con 10% de valores fuzzy

	$Unselect_{3\times5\text{-}FRF_{LQD}}$		**FRF-$fs_{3\times5\text{-}FRF_{LQD}}$**				
	training	test	training	test	#fe	red. rate	p-value
AUS	$100.0_{0.00}$	$86.71_{3.43}$	$94.46_{0.45}$	$85.85_{3.38}$	$4.7_{3.79}$	66.7	0.06345
BCW	$99.49_{0.13}$	$95.90_{1.14}$	$98.39_{0.27}$	$95.52_{1.54}$	$5.3_{2.08}$	40.7	0.3914
GER	$100.0_{0.00}$	$72.67_{3.05}$	$100.0_{0.00}$	$72.60_{3.09}$	$14.3_{0.58}$	40.3	0.5912
GLA	$98.95_{0.69}$	$75.08_{7.51}$	$97.86_{0.70}$	$74.77_{8.47}$	$6.0_{0.00}$	33.3	0.7297
HEA	$100.0_{0.00}$	$77.16_{5.64}$	$96.82_{0.29}$	$75.19_{5.57}$	$5.0_{1.00}$	61.5	0.2998
ION	$99.17_{0.14}$	$94.11_{3.84}$	$98.72_{0.40}$	$94.40_{2.61}$	$13.3_{6.03}$	60.8	1.0000
IRP	$97.61_{0.74}$	$96.67_{3.01}$	$96.00_{1.09}$	$96.00_{4.35}$	$2.0_{0.00}$	50.0	0.05349
PIM	$100.0_{0.00}$	$76.43_{1.78}$	$100.0_{0.00}$	$75.69_{1.64}$	$5.7_{0.58}$	29.2	0.00135
SON	$100.0_{0.00}$	$84.13_{7.18}$	$100.0_{0.00}$	$84.62_{6.43}$	$15.7_{5.86}$	73.9	0.2052
SPE	$100.0_{0.00}$	$79.78_{5.02}$	$99.75_{0.20}$	$81.02_{7.08}$	$14.3_{8.08}$	67.4	0.09521
VEH	$100.0_{0.00}$	$72.85_{2.51}$	$100.0_{0.00}$	$73.52_{2.83}$	$15.0_{3.46}$	16.7	0.1566
WDC	$100.0_{0.00}$	$95.20_{2.20}$	$100.0_{0.00}$	$95.25_{2.40}$	$17.7_{3.21}$	43.0	0.2086
WIN	$99.95_{0.10}$	$96.25_{3.07}$	$98.92_{0.91}$	$96.63_{2.59}$	$7.0_{0.00}$	46.2	0.1863
ZOO	$100.0_{0.00}$	$94.72_{5.12}$	$99.42_{0.66}$	$95.05_{6.54}$	$7.7_{2.88}$	52.1	0.00971
Media	$99.65_{0.13}$	$85.55_{3.89}$	$98.60_{0.35}$	$85.44_{4.06}$	$9.5_{2.68}$	48.7	0.386

Sobre la Tabla 11.5 se puede observar que el resultado de precisión medio obtenido con la selección de atributos por la técnica FRF-fs es bastante similar a los resultados obtenidos

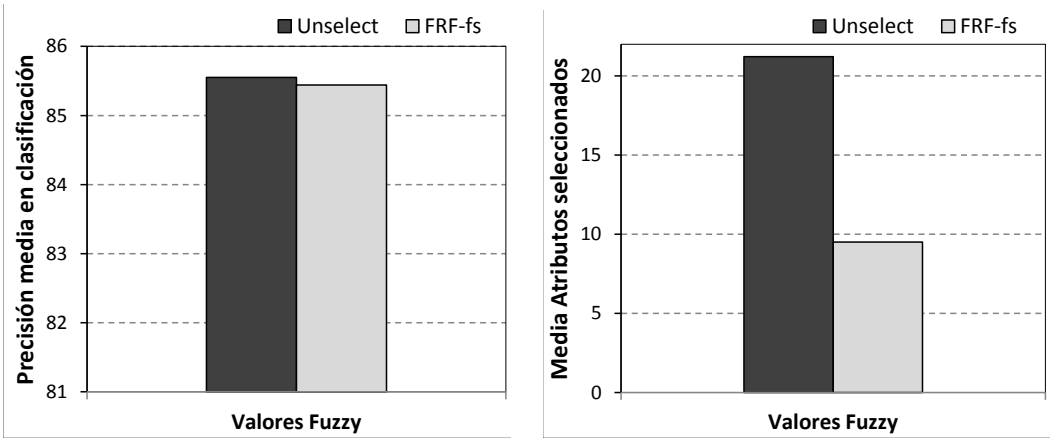


Figura 11.5: Comparando la precisión media y la media de reducción utilizando todas los atributos (sin selección) y utilizando los atributos seleccionados

cuando no se lleva a cabo la selección. Analizando los diferentes p-values con $\alpha = 0.05$, podemos concluir:

a) Para el conjunto de datos PIM, la técnica obtiene peor resultado en precisión que la evaluación sin selección.

b) No hay diferencias significativas en los otros conjuntos de datos.

c) Globalmente, analizando todos los resultados podemos concluir que no hay diferencias significativas entre la evaluación con todas los atributos y con la selección obtenida por FRf-fs.

Además, podemos concluir que a pesar de que los conjuntos de datos contienen información LQD explícita hay un importante ratio de reducción con una media del 48.7 %. Por tanto FRF-fs lleva a cabo una selección de atributos que es robusta ante conjuntos de datos con valores fuzzy.

11.3.6.2 **Selección de atributos en conjuntos de datos con valores intervalares**

Los resultados de precisión se muestran en la Tabla 11.6 y de forma gráfica en la Figura 11.6. Training y test son los porcentajes medios de clasificación (media y desviación estándar) para los datos de training y de test respectivamente. Estos valores de precisión se obtienen por medio de la clasificación con un ensamble FRF_{LQD} con 500 árboles, para los conjuntos de datos completos sin selección de atributos (columna $\text{Unselec}_{3\times5\text{-FRF}_{\text{LQD}}}$ de la Tabla 11.6) y para los conjuntos de datos con los atributos seleccionados por FRf-fs (columna $\text{FRF-fs}_{3\times5\text{-FRF}_{\text{LQD}}}$ de la Tabla 11.6). Además, también se muestran los p-values obtenidos tras aplicar el test de estadístico de Wilcoxon para comparar los resultados.

Tabla 11.6: Precisión en los conjuntos de datos con 10% de valores intervalares

	Unselect$_{3\times 5\text{-FRF}_{LQD}}$		**FRF-fs$_{3\times 5\text{-FRF}_{LQD}}$**				
	training	test	training	test	#fe	red. rate	p-value
AUS	$100.0_{0.00}$	$86.62_{3.58}$	$91.46_{0.53}$	$86.14_{2.36}$	$5.0_{1.73}$	64.3	0.3001
BCW	$100.0_{0.00}$	$96.14_{1.56}$	$98.87_{0.14}$	$95.99_{1.39}$	$6.0_{2.00}$	33.3	0.2979
GER	$99.87_{0.09}$	$74.10_{3.02}$	$99.86_{0.09}$	$74.23_{2.97}$	$20.0_{1.73}$	16.7	0.1696
GLA	$99.88_{0.19}$	$78.19_{5.93}$	$99.73_{0.19}$	$79.13_{6.69}$	$7.0_{1.00}$	22.2	0.1959
HEA	$99.63_{0.21}$	$78.27_{6.34}$	$96.85_{0.63}$	$80.25_{5.92}$	$6.7_{2.08}$	48.7	0.5279
ION	$100.0_{0.00}$	$94.68_{3.02}$	$100.0_{0.00}$	$94.49_{3.34}$	$14.0_{6.24}$	58.8	0.9249
IRP	$96.83_{0.79}$	$94.00_{2.88}$	$92.11_{9.17}$	$90.67_{12.26}$	$1.0_{0.00}$	75.0	0.5086
PIM	$100.0_{0.00}$	$75.48_{2.05}$	$98.05_{0.26}$	$74.87_{2.02}$	$4.0_{1.00}$	50.0	0.1726
SON	$100.0_{0.00}$	$83.81_{1.70}$	$100.0_{0.00}$	$83.01_{4.31}$	$19.7_{3.51}$	67.2	0.7295
SPE	$100.0_{0.00}$	$79.78_{5.01}$	$99.97_{0.07}$	$81.52_{4.92}$	$16.0_{1.73}$	63.6	0.03734
VEH	$100.0_{0.00}$	$69.82_{2.79}$	$99.67_{0.19}$	$71.75_{3.20}$	$10.7_{2.08}$	40.7	0.00209
WDC	$100.0_{0.00}$	$95.90_{1.53}$	$100.0_{0.00}$	$95.61_{1.75}$	$23.0_{5.29}$	25.8	0.3554
WIN	$100.0_{0.00}$	$95.69_{2.11}$	$100.0_{0.00}$	$94.38_{1.39}$	$7.3_{2.31}$	43.6	0.1658
ZOO	$100.0_{0.00}$	$96.04_{5.14}$	$99.59_{0.29}$	$96.37_{4.17}$	$7.7_{2.08}$	52.1	0.1015
Media	$99.73_{0.09}$	$85.61_{3.33}$	$98.30_{0.83}$	$85.60_{4.07}$	$10.6_{2.34}$	47.3	0.4955

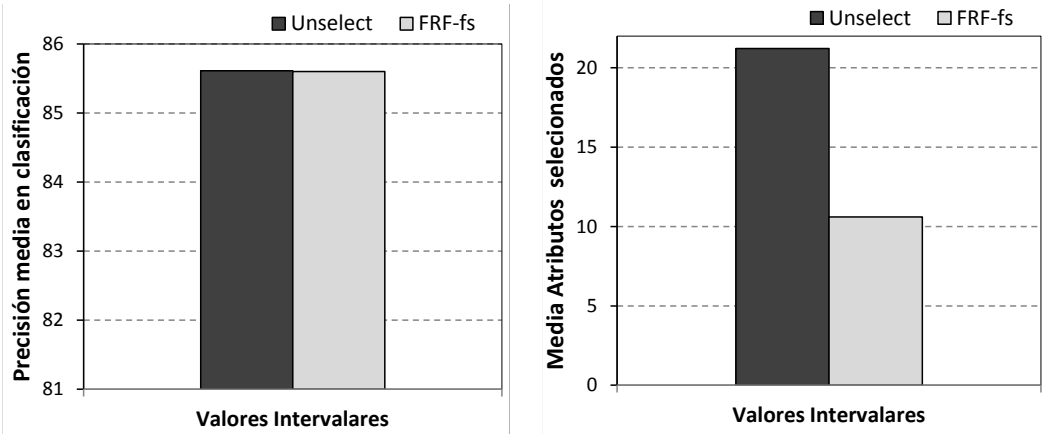


Figura 11.6: Comparando la precisión media y la media de reducción utilizando todos los atributos (sin selección) y utilizando los atributos seleccionados

Observando los resultados de precisión de la Tabla 11.6, podemos indicar que parecen bastante similares. Analizando los diferentes p-values con $\alpha = 0.05$, podemos concluir:

a) Para los conjuntos de datos SPE y VEH, la evaluación con la selección realizada por FRF-fs obtiene mejor precisión que la evaluación sin selección.

b) No existen diferencias significativas en el resto de conjuntos de datos.

c) Globalmente, y analizando todos los resultados obtenidos, podemos concluir que no hay

diferencias significativas entre evaluar con todos los atributos y evaluar sólo con la selección obtenida por FRF-fs.

Además, debemos de señalar que a pesar de que los conjuntos de datos contienen información LQD explícitamente, se consigue un ratio de reducción alto con un valor medio de 47.3 %. De nuevo, el comportamiento de FRF-fs es estable ante un conjunto de datos con valores intervalares. Por lo tanto hemos propuesto una técnica de selección de atributos con un rendimiento comparable a las mejores técnicas de la literatura y con un comportamiento robusto ante conjunto de datos con LQD.

11.4 Utilizando BAGOFP_CLASS en la técnica FRF-fs

Como hemos visto en la Sección 11.2, la primera etapa del algoritmo FRF-fs consiste en escalar y discretizar los atributos. Esta primera fase determina la calidad de los atributos seleccionados para las siguientes etapas.

En los experimentos llevados a cabo hasta ahora, en esta primera fase hemos utilizado la técnica OFP_CLASS$_{\text{LQD}}$ como técnica de discretización. Aunque los resultados que hemos obtenido han sido bastante satisfactorios, vamos a aplicar la técnica de discretización BAGOFP_CLASS [38] en aquellos conjuntos de datos donde el número de atributos es muy superior al número de ejemplos y clases (como ocurre con los conjuntos de datos de microarrays presentados en la Tabla 11.1). Por esta razón, vamos a repetir la experimentación realizada en el apartado 11.3.5, utilizando como algoritmo de discretización el algoritmo BAGOFP_CLASS en lugar de OFP_CLASS$_{\text{LQD}}$. Tras ejecutar la experimentación, vamos a comparar los resultados obtenidos con los resultados al utilizar el algoritmo OFP_CLASS$_{\text{LQD}}$ y con los resultados obtenidos en [82] que ya hemos mostrado en la Tabla 11.4.

11.4.1 Configuración del experimento

Los parámetros utilizados en la ejecución del algoritmo de selección de atributos con BAGOFP_CLASS son los mismos que hemos utilizado anteriormente:

- Parámetros para el ensamble:
  - Número de FDT$_{\text{LQD}}$s 500.
  - Selección aleatoria en cada nodo desde el conjunto disponible: $\sqrt{|A|}$
- Vector para combinar información INF (Algoritmo 21): $p = (1, \frac{6}{7}, \frac{2}{3}, \frac{2}{4}, \frac{2}{5}, \frac{2}{P_{Na}+1}, \ldots)$ con P_{Na} profundidad del nodo N el cual contiene el atributo a.

- Vector de pesos normalizados para el cálculo de $OWAIMP$: $W = (1, \frac{1}{2}, \ldots, \frac{1}{|A|})$ siendo $|A|$ el número de atributos.
- La técnica Wrapper:
  - Se utiliza una validación cruzada de tamaño 3 para evaluar el rendimiento de cada grupo de atributos seleccionados.
  - La técnica de aprendizaje automático, $\text{Classifier}_{\text{LQD}}$, es un ensamble FRF_{LQD} con 200 FDT_{LQD}s.
  - Umbrales: $\delta_1 = \delta_3 = 0.5$, $\delta_2 = 0.05$.
- Parámetros de BAGOFP_CLASS: $\delta = 20$ y $\beta = 30$

11.4.2 **Resultados y análisis del experimento**

En la Tabla 11.7 se muestran los resultados medios obtenidos por el algoritmo de selección de atributos FRF-fs utilizando como algoritmos de discretización $\text{OFP\_CLASS}_{\text{LQD}}$ (FRF-fs($\text{OFP\_CLASS}_{\text{LQD}}$)) y BAGOFP_CLASS (FRF-fs(BAGOFP_CLASS)). % acc es el porcentaje medio de clasificación obtenido con la técnica .632+bootstrap, $\#fe$ es el número de atributos seleccionados y p-value indica el p-value obtenido tras aplicar el test de signos de Wilcoxon tomando como hipótesis nula que las técnicas tienen igual comportamiento y como alternativa que las técnicas tienen diferente comportamiento, tomando como valor de α =0.05.

Tabla 11.7: Resultados obtenidos por FRF-fs con $\text{OFP\_CLASS}_{\text{LQD}}$ y BAGOFP_CLASS

	FRF-fs($\text{OFP\_CLASS}_{\text{LQD}}$)$_{0.632+bs}$		**FRF-fs(BAGOFP_CLASS)**$_{0.632+bs}$		
	% acc.	#fe	% acc.	#fe	**p-value**
LEU	$99.99_{0.02}$	1	$99.99_{0.02}$	1	1.00000
BR2	$86.56_{0.12}$	7	$87.55_{0.19}$	6	0.00391
BR3	$81.77_{0.31}$	6	$82.78_{0.17}$	7	0.00391
NCI	$89.55_{0.14}$	24	$89.55_{0.14}$	24	1.00000
ADE	$95.97_{0.05}$	9	$96.47_{0.17}$	6	0.00391
BRA	$96.88_{0.07}$	13	$97.38_{0.09}$	12	0.00391
COL	$92.58_{0.14}$	8	$93.29_{0.21}$	4	0.00391
LYM	$99.97_{0.02}$	6	$99.98_{0.01}$	4	0.44120
PRO	$96.41_{0.04}$	8	$96.62_{0.04}$	7	0.00391
SRB	$98.32_{0.09}$	9	$99.75_{0.04}$	13	0.00391
media	$93.80_{0.10}$	9.1	$94.34_{0.10}$	8.4	2.416e-14

Analizando la Tabla 11.7, y observando los diferentes p-values, podemos concluir:

a) Para los conjuntos de datos LEU, NCI y LYM aceptamos la hipótesis nula por lo que no hay diferencias significativas entre los resultados obtenidos por las dos técnicas.

b) Para el resto de conjuntos de datos, rechazamos la hipótesis nula y aceptamos la alternativa, lo que indica que hay diferencias significativas entre las dos técnicas. Para concretar qué técnica es mejor vamos a realizar un nuevo análisis estadístico manteniendo como hipótesis nula que no hay diferencias significativas entre las dos técnicas y como hipótesis alternativa que la técnica FRF-fs(BAGOFP_CLASS) es mejor que la técnica FRF-fs($\text{OFP\_CLASS}_{\text{LQD}}$). El p-value es el mismo para todos los conjuntos de datos siendo de 0.00195, por lo que con un valor de α = 0.05 rechazamos la hipótesis nula y aceptamos la hipótesis alternativa, por tanto FRF-fs(BAGOFP_CLASS) es el mejor.

c) Globalmente, después del análisis de todos los resultados obtenidos para los diferentes conjuntos de datos, podemos concluir que hay diferencias significativas entre los resultados de las dos técnicas, siendo FRF-fs(BAGOFP_CLASS) la mejor técnica, obteniendo además una reducción mayor en el número de atributos. Para afirmar esto, hemos realizado un test estadístico tomando como hipótesis alternativa que FRF-fs(BAGOFP_CLASS) es la mejor técnica y el p-value que hemos obtenido en conjunto es de 2.42e-14, por lo que con un α = 0.05 rechazamos la hipótesis nula y aceptamos la hipótesis alternativa siendo FRF-fs(BAGOFP_CLASS) la mejor técnica.

Por último, vamos a comparar los resultados obtenidos con FRF-fs(BAGOFP_CLASS) con RF.ge [82] que es la técnica que mejor resultado ha obtenido anteriormente de acuerdo a los resultados de la Tabla 11.3. En la Tabla 11.8 se muestran los resultados medios obtenidos por el algoritmo RF.ge y los obtenidos por el algoritmo FRF-fs(BAGOFP_CLASS). $\%$ acc es el porcentaje medio de clasificación obtenido con la técnica .632+bootstrap, $\#fe$ es el número de atributos seleccionados y $p - value$ indica el p-value obtenido tras aplicar el test de signos de Wilcoxon, tomando como hipótesis nula que las técnicas tienen similar comportamiento y como alternativa que las técnicas tienen comportamiento diferentes.

Antes de analizar los resultados debemos de tener en cuenta que las dos técnicas, RF.ge y FRF-fs(BAG_OFPCLASS), tienen un buen rendimiento predictivo. Analizando la Tabla 11.8, y observando los diferentes p-values con α = 0.05, podemos concluir:

a) Para el conjunto de datos LEU, aceptamos la hipótesis nula, por lo tanto no hay diferencias significativas entre los resultados obtenidos por las dos técnicas.

b) Para el resto de conjuntos de datos, el valor de los p-values nos indican que hay diferencias significativas entre las técnicas. Vamos a realizar un nuevo análisis estadístico suponiendo la misma hipótesis nula y como hipótesis alternativa tomamos que la técnica

Tabla 11.8: Resultados obtenidos por RF.ge y FRF-fs(BAGOFP_CLASS)

	RF.ge$_{0.632+bs}$		**FRF-fs(BAGOFP_CLASS)**$_{0.632+bs}$		
	% acc.	#fe	% acc.	#fe	**p-value**
LEU	$99.99_{0.02}$	1	$99.99_{0.02}$	1	1.00000
BR2	$90.38_{0.24}$	8	$87.55_{0.19}$	6	0.00391
BR3	$80.73_{0.16}$	8	$82.78_{0.17}$	7	0.00391
NCI	$91.19_{0.39}$	26	$89.55_{0.14}$	24	0.00391
ADE	$95.90_{0.20}$	6	$96.47_{0.17}$	6	0.00391
BRA	$98.08_{0.25}$	9	$97.38_{0.09}$	12	0.00391
COL	$91.70_{0.18}$	8	$93.29_{0.21}$	4	0.00391
LYM	$99.88_{0.05}$	12	$99.98_{0.01}$	4	0.00391
PRO	$96.07_{0.07}$	6	$96.62_{0.04}$	7	0.00391
SRB	$98.01_{0.06}$	9	$99.75_{0.04}$	13	0.00391
media	$94.19_{0.16}$	9.3	$94.34_{0.10}$	8.4	0.28090

RF.ge es mejor que la técnica FRF-fs(BAGOFP_CLASS). Los resultados de este nuevo análisis son:

b.1) Para los conjuntos de datos BR2, NCI y BRA el p-value obtenido es 0.00195, por lo tanto con un valor de α = 0.05, rechazamos la hipótesis nula y aceptamos la hipótesis alternativa, es decir, que para los conjuntos de datos BR2, NCI y BRA la técnica que obtiene un mejor comportamiento es RF.ge y además es la técnica que consigue la mejor media.

b.2) Para los conjuntos de datos BR3, ADE, COL, LYM, PRO y SRB, el p-value obtenido es 1.0, por lo tanto con un valor de α = 0.05, deberíamos de aceptar la hipótesis nula porque $1.0 > \alpha$, pero ya hemos indicado que hay diferencias significativas entre las técnicas, por lo que la conclusión es que la técnica FRF-fs(BAGOFP_-CLASS) es mejor que la técnica RF-ge.

c) Por último y globalmente, después del análisis de todos los resultados obtenidos para los diferentes conjuntos de datos, podemos concluir que no hay diferencias significativas entre los resultados de las dos técnicas, ya que el p-value de 0.28090 nos indica que debemos de aceptar la hipótesis nula. Además ambas técnicas obtienen una reducción de atributos similar.

Por lo tanto, podemos concluir que FRF-fs(BAGOFP_CLASS) tiene un comportamiento competitivo con las mejores técnicas de la literatura cuando trabaja con conjuntos de datos de microarrays, teniendo además como ventaja adicional su capacidad de trabajar con LQD.

Por último, vamos a mostrar los atributos (genes) seleccionados por cada uno de las técnicas: RF.ge, FRF-fs(OFP_CLASS$_{LQD}$) y FRF-fs(BAGOFP_CLASS).

Genes seleccionados por la técnica**RF.ge**:

- ADENO (6 genes): 3622, 5574, 7704, 8733, 8842, 8892
- BRAIN (9 genes): 56, 1049, 1394, 2445, 2804, 2848, 3338, 3815, 4080
- BREASTC-2c (8 genes): 44, 336, 1135, 1399, 3460, 3700, 4359, 4843
- BREASTC-3c (8 genes): 44, 216, 1135, 2712, 3144, 3167, 3661, 4371
- COLON (8 genes): 249, 493, 513, 822, 1582, 1635, 1771, 1892
- LEUKEMIA (1 gen): 2124
- LYMPHOMA (12 genes): 734, 740, 1622, 2733, 2736, 2801, 3704, 3763, 3765, 3787, 3804, 3838
- NCI (26 genes): 188, 204, 1446, 1686, 1699, 2259, 3129, 3351, 3404, 3490, 3895, 4284, 4309, 4397, 4565, 4566, 4580, 4646, 4654, 4660, 4672, 4675, 4677, 4680, 4987, 5176
- PROSTATE (6 genes): 1839, 2619, 3118, 3969, 5016, 5035
- SRBCT (9 genes): 246, 509, 545, 742, 1003, 1389, 1645, 1708, 2050

Genes seleccionados por la técnica **FRF-fs(OFP_CLASS)**

- ADENO (9 genes): 869, 1758, 2858, 5279, 5574, 6954, 7197, 7937, 3600
- BRAIN (13 genes): 56, 423, 2515, 2804, 2990, 3446, 4210, 5175, 5213, 5255, 5300, 5460, 5537
- BREASTC-2c (7 genes): 413, 1872, 2272, 2594, 3700, 3765, 4359
- BREASTC-3c (6 genes): 44, 316, 2443, 2957, 3167, 3979
- COLON (8 genes): 245, 267, 493, 802, 822, 972, 1004, 1444
- LEUKEMIA (1 gen): 2124
- LYMPHOMA (6 genes): 2774, 3595, 3597, 3747, 3763, 3767
- NCI (24 genes): 80, 1371, 2254, 2420, 3490, 3994, 4077, 4177, 4221, 4284, 4306, 4322, 4369, 4389, 4396, 4407, 4421, 4511, 4556, 4558, 4580, 4633, 4819, 5176
- PROSTATE (8 genes): 1359, 1881, 2619, 4087, 4183, 4287, 4335, 5016
- SRBCT (9 genes): 246, 255, 742, 836, 1770, 1876, 1911, 1955, 2046

Genes seleccionados por la técnica **FRF-fs(BAGOFP_CLASS)**

- ADENO (6 genes): 869, 2858, 5279, 5574, 6954, 7937
- BRAIN (12 genes): 56, 423, 1049, 1332, 2076, 2804, 3373, 3815, 3989, 4143, 4543, 4884
- BREAST-2c (6 genes): 316, 1399, 2272, 3286, 3700, 4359
- BREAST-3c (7 genes): 300, 316, 366, 2272, 2712, 2727, 2982
- COLON (4 genes): 377, 1168, 1679, 1843
- LEUKEMIA (1 gen): 2124
- LYMPHOMA (4 genes): 2535, 2669, 3015, 3763
- NCI (24 genes): 80, 1371, 2254, 2420, 3490, 3994, 4077, 4177, 4221, 4284, 4306, 4322, 4369, 4389, 4396, 4407, 4421, 4511, 4556, 4558, 4580, 4633, 4819, 5176.
- PROSTATE (7 genes): 1881, 2293, 2534, 2619, 4212, 5016, 5278
- SRBCT (13 genes): 107, 246, 545, 1003, 1203, 1353, 1389, 1577, 1613, 1627, 1954, 1955, 2050

CAPÍTULO

12

Imputación de valores missing y selección de ejemplos

12.1 Introducción

Ya hemos comentado que una de las formas más frecuente de LQD en los conjuntos de datos son los valores missing y que se han desarrollado muchas técnicas para reemplazar/imputar estos valores missing por otros valores estimados utilizando la información disponible en el conjunto de datos.

Sin embargo, la mayoría de las técnicas de imputación desarrolladas suponen la existencia de un conjunto de ejemplos que carecen de datos de baja calidad. De ahí que nuestra propuesta sea la de desarrollar una técnica de imputación predictiva que sea capaz de llevar a cabo la imputación de valores missing a partir de ejemplos que contengan explícitamente valores de baja calidad.

La técnica de imputación que proponemos está basado en la técnica k-vecinos más cercanos, ya que esta técnica tiene como ventaja que el modelo de los datos es global y, por lo tanto, permite predecir cualquier atributo sin necesidad de aprender modelos distintos para cada atributo. Sin embargo, entre sus principales desventajas se encuentra el crecimiento del costo computacional en el cálculo de los vecinos más cercanos conforme aumenta el tamaño del conjunto de datos. Es por esto, que proponemos además la extensión de una técnica de selección de ejemplos que permita reducir el conjunto de datos inicial por un subconjunto representativo de estos y que mantengan la calidad de la imputación. Esta extensión, evidentemente, permite llevar a cabo la selección de ejemplos desde LQD.

Dado que la técnica de imputación que proponemos está basada en la técnica k-vecinos más cercanos, en las siguientes secciones se describe la técnica de imputación k-vecinos más cercanos desde datos crisp, para posteriormente llevar a cabo la extensión de la misma para imputar valores missing a partir de LQD. A continuación, proponemos la técnica CNN_{LQD} para la selección de ejemplos desde LQD. Por último, presentamos un conjunto de experimentos que validan las técnicas propuestas.

12.2 Imputación de valores missing desde datos crisp

La técnica de imputación k-vecinos más cercanos es una de las más utilizadas como técnica de imputación desde datos crisp, [13, 70]. Es un procedimiento paramétrico para estimar los valores missing de los atributos de los ejemplos de entrada, basándose en la información proporcionada por los k-vecinos más cercanos cada ejemplo.

Sea E un conjunto de ejemplos descritos por un conjunto A de atributos y C el conjunto de posibles clases de los ejemplos. Los atributos pueden ser nominales o numéricos y deben contener al menos dos atributos, siendo uno de ellos un atributo nominal que actúa como atributo clase. Sin perdida de generalidad vamos a suponer que el atributo de clase es el último atributo.

Sea z un ejemplo con algunos valores missing en sus atributos nominales y numéricos, (al menos uno de los atributos en el ejemplo debe de ser conocido para poder calcular las distancias). Como función distancia podemos usar la misma que comentamos en la Sección 8.2 del Capítulo 8 o cualquier otra que permita medir cómo de cercanos están dos ejemplos entre sí. Para imputar los valores missing de z, tomamos los k ejemplos de E más cercanos a z. El proceso de imputación se lleva a cabo como se muestra en el Algoritmo 23.

Como podemos observar en el Algoritmo 23, todos los valores imputados son crisp, dado que todos los valores de entrada también lo son.

12.3 Imputación de valores missing desde datos de baja calidad

Tras presentar cómo trabaja la técnica de imputación KNN que es capaz de tratar sólamente con datos crisp, vamos a extender esta técnica para que sea capaz de trabajar con LQD, [143], [45], .

Al igual que al definir la técnica KNN_{LQD} para clasificar, debemos de considerar que los atributos numéricos van a estar expresados mediante cuádruplas y que atributos nominales van a estar expresados mediante subconjuntos crisp/fuzzy. Además también debemos de definir una métrica que nos sirva como medida de distancia $d_{LQD}(\cdot,\cdot)$ entre dos ejemplos e_j y e_h descritos con algunos valores de baja calidad y que estará definida de forma general igual que en la técnica KNN_{LQD} para clasificar (Sección 8.3).

Algoritmo 23 Proceso de imputación mediante KNN

ImputacionKNN(in: z, E, k ($1 \leq k \leq |E|$); **out:** $Valores\_Imputados$)
begin

1. Inicializar:
   - 1.1 Calcular la distancia $d(e_j, z)$ entre z y cada $e_j \in E$
   - 1.2 Seleccionar $K \subseteq E$, el conjunto de los k ejemplos de E más cercanos a z
   - 1.3 Calcular la clase más frecuente en K. Sea c tal clase y K_l el subconjunto de ejemplos de K pertenecientes a esa clase
2. **if** (la clase de z es missing) **then**
   - 2.1 **for all** (atributos missing z^i en z) **do**
     - **if** (z^i numérico) **then** $z^i = \frac{1}{|K_l|} \sum_{j=1,\ldots,|K_l|} e_j^i$
     - **else**
       - **if** (z^i nominal) **then** z^i=valor más frecuente del atributo en K_l
       - **end if**
     - **end if**
   - **end for**

   end if
3. **if** (la clase de z es conocida) **then**
   - **if** ($clase(z) = c$) **then** estimar los valores missing como en el paso 2.1
   - **else**
     - la imputación no se lleva a cabo
   - **end if**

   end if

end

Utilizando la función $d_{LQD}(\cdot, \cdot)$, obtenemos los k ejemplos más cercanos al ejemplo z. Denotamos a este conjunto como K_{LQD}. A partir de este conjunto se realiza la imputación de los valores missing de z. Puesto que la clase de los ejemplos puede ser imprecisa (un subconjunto crisp/fuzzy), debemos de definir cómo tal valor de clase es considerado en la imputación. En la técnica de imputación KNN, si la clase de z es conocida, solamente los vecinos de k que pertenezcan a esta clase se consideran en la imputación. Ahora, en la técnica KNN_{LQD}, ponderamos la contribución de cada vecino a la imputación en proporción a la distancia de su clase con la clase de z. Si la clase de z es missing, se obtiene la clase más frecuente entre sus vecinos, la cual puede ser imprecisa. Este valor de clase imputado en z es el que se considera para obtener el peso de cada vecino en la imputación de los otros valores missing de z. El Algoritmo 24 muestra el proceso de la técnica KNN_{LQD}.

Las funciones f_1 y f_2, y por lo tanto d_{LQD}, pueden definirse como cualquier función válida con datos de baja calidad. En el Algoritmo 24, el umbral α define el mínimo nivel o grado de creencia, en media, por el cual el usuario está dispuesto a llevar a cabo la imputación. Esto es,

Algoritmo 24 Proceso de imputación mediante KNN_{LQD}

ImputacionKNN$_{\text{LQD}}$(in: z, E, k $(1 \leq k \leq |E|)$; **out:** $Valores\_Imputados$)
begin

1. Inicializar:
   1.1 Calcular la distancia $d_{LQD}(e_j, z)$ entre z y cada $e_j \in E$
   1.2 Seleccionar $K_{LQD} \subseteq E$, el conjunto de los k ejemplos de E más cercanos a z
2. **if** (la clase de z es missing) **then**
   $clase(z) = \{\frac{\sum_{e_j \in K_{LQD}} \mu(e_j,c)}{|K_{LQD}|}/c\}; c \in C$ donde $\mu(e_j, c)$ es el grado de creencia del valor de clase c en el j-ésimo ejemplo
   end if
3. **for all** (ejemplo e_j en K_{LQD}) **do**
   Calcular $dclass_j = f_2(e_j^n, z^n)$
   $p_j = 1 - dclass_j$
   end for
4. Calcular $P_{K_{LQD}} = \sum_{e_j \in K_{LQD}} p_j$
5. **if** $(\frac{P_{K_{LQD}}}{k} \geq \alpha)$ **then**
   for all (atributo missing z^i en z) **do**
   if (z^i es numérico) **then** $z^i = \frac{\sum_{e_j \in K_{LQD}} p_j \cdot e_j^i}{P_{K_{LQD}}}$
   else
   if (z^i es nominal) **then**
   $z^i = \{\frac{\sum_{e_j \in K_{LQD}} p_j \cdot \mu(e_j,v)}{|K_{LQD}|}/v\}; \forall v \in \Omega_i = \{$dominio del atributo $i\}$, donde $\mu(e_j, v)$ es el grado de creencia del valor v en el i-ésimo atributo en el ejemplo e_j
   end if
   end if
   end for
   else
   La imputación no se realiza
   end if

end

$\frac{P_{K_{LQD}}}{k}$ se define como la media de los grados de creencia en los cuales se basa la imputación y el usuario puede determinar el mínimo grado de creencia, en media, que permite para llevar a cabo la imputación. Cuando el usuario define $\alpha = 0$, todas las imputaciónes son llevadas a cabo independientemente del grado de creencia, sin embargo, cuando $\alpha = 1$ solamente se llevarán a cabo las imputaciones basadas en ejemplos con el mismo valor de clase que el ejemplo a imputar. Por lo tanto, el usuario debe de determinar el valor de k, el valor de α y una métrica específica d_{LQD} para calcular la distancia.

12.4 Selección de ejemplos desde datos de baja calidad

Hemos comentado anteriormente que la técnica KNN, tiene como desventaja el aumento de la complejidad computacional en la búsqueda de los k vecinos más cercanos conforme aumenta

la dimensión del conjunto de datos. Para intentar solucionar esta desventaja se han propuesto distintas aproximaciones y técnicas que están basadas en la reducción de ejemplos. El objetivo de estas técnicas es obtener un conjunto de ejemplos representativo del conjunto original con un tamaño bastante inferior y con similar o superior precisión de clasificación para los nuevos ejemplos de entrada. En la literatura, son conocidas como técnicas de reducción, selección de ejemplos o selección de prototipos. Dependiendo de la estrategia que sigue cada técnica, puede eliminar ruido, redundancia o ambas, [78].

Una clasificación muy extendida de las técnicas de selección de ejemplos distingue tres clases de técnicas: edición, condensación y técnicas híbridas, [7]. El objetivo de las técnicas de edición es eliminar ruido, el objetivo de las técnicas de condensación es eliminar redundancia y las técnicas híbridas tratan de eliminar ruido y redundancia. Sin embargo, estas técnicas han sido diseñadas suponiendo que los ejemplos (que constituyen el clasificador) carecen de imperfección, es decir, que los datos con los cuales se trabaja son datos crisp.

Como punto de partida en el estudio y diseõ de técnicas de selección de ejemplos que puedan trabajar con LQD, hemos elegido la técnica de condensación CNN. La decisión la hemos tomando en base a los buenos resultados que ofrece a pesar de la simplicidad del mismo. Es importante resaltar que esta técnica es un estudio preliminar para analizar como funciona la técnica al extenderla para trabajar con LQD y que en trabajos futuros se abordará el estudio y disenõ de otras técnicas. Una revisión más detallada de la técnica CNN y de otras técnicas de condensación podemos encontrarla en [7, 200] y una comparativa bastante completa en [78].

A continuación vamos a presentar muy brevemente la técnica de condensación CNN, para posteriormente abordar una primera extensión de la técnica para que sea capaz de trabajar con LQD que denominaremos CNN_{LQD}, [143]. Por último, mostramos algunos resultados experimentales que miden la precisión de la imputación llevada a cabo con la técnica KNN_{LQD} sobre conjuntos de datos sin condensar y condensados con la técnica CNN_{LQD}.

12.4.1 **Técnica de condensación CNN**

La técnica CNN, (Condensed Nearest Neighbors), es una de las más utilizadas y fue desarrollada por Hart en 1966 [94]. Si bien es una de las primeras técnicas de selección de ejemplos que aparece en la literatura, tiene un buen comportamiento frente a nuevas propuestas manteniendo cierta simplicidad. Este algoritmo construye un subconjunto de ejemplos S a partir del conjunto de ejemplos E tal que todo ejemplo de E estará más cerca a un ejemplo de S de la misma clase que a otro ejemplo de S de distinta clase. El algoritmo comienza seleccionando un ejemplo de cada clase de E y los inserta en S. Cada ejemplo de E se clasifica con la regla $1 - NN$ usando solamente los ejemplos de S. Si un ejemplo no se clasifica bien, se añade a S.

Este proceso se repite hasta que no haya ejemplos en E que se clasifiquen incorrectamente. Su mayor desventaja es que es una técnica altamente dependiente del orden de presentación de los ejemplos. Sin embargo, tiende a quedarse con los ejemplos cercanos a la frontera de decisión de cada clase, ya que estos son los que tienden a estar mal clasificados.

12.4.2 Técnica de condensación CNN_{LQD}

En esta sección vamos a presentar la extensión de la técnica CNN para que pueda llevar a cabo el condensado desde LQD. Esta extensión es denominada CNN_{LQD}. En el Algoritmo 25 se muestra de forma completa el proceso llevado a cabo por la técnica CNN_{LQD}.

Algoritmo 25 Proceso de condensación CNN_{LQD}

CondensarCNN$_{\text{LQD}}$(in: E; **out:** S)
begin

1. Inicializar:
   1.1 $S = \emptyset$
   1.2 **for** ($j = 1$ a $|E|$) **do**
   Calcular $EntropiaF(e_j^n)$
   end for
2. Ordenar E en orden creciente de acuerdo a la entropía fuzzy
3. Seleccionar de E un ejemplo de cada una de las clases existentes y añadirlos a S
4. Eliminar los ejemplos seleccionados en el paso anterior del conjunto E
5. **while** (S cambie) **do**
   for $j = 1$ a $|E|$ **do**
   Calcular el vecino de S más cercano a e_j de acuerdo a la función $d_{LQD}(\cdot,\cdot)$. Sea NN dicho ejemplo
   Calcular $simil_{clase}(clase(e_j), clase(NN)) = 1 - f_2(clase(e_j), clase(NN))$
   if ($simil_{clase} \geq \beta$) **then**
   No se añade el ejemplo e_j al conjunto S
   end if
   Añadir e_j al conjunto S y eliminarlo de E;
   end if
   end for
   end while

end

El Algoritmo 25 ordena los ejemplos del conjunto inicial de acuerdo a la entropía fuzzy ($EntropiaF(\cdot)$), [60], del atributo clase. De esta forma se consideran en primer lugar aquellos ejemplos con valores de clase menos imperfectos.

Como se indica en el Algoritmo 25, para cada ejemplo de E se calcula su vecino más cercano del conjunto S usando la técnica $1-NN$ y se comparan sus valores de clases mediante una medida de similitud donde $f_2(\cdot,\cdot)$ es la función que nos indica la distancia entre los dos valores de clase.

Si la similitud de las clases está por debajo de un valor de umbral β, el ejemplo se considera mal clasificado con los ejemplos de S y se añade a dicho conjunto. El valor β es un parámetro externo de la técnica que permite definir el grado con el que el usuario está dispuesto a considerar que dos ejemplos pertenecen a la misma clase. Cuanto más se acerque su valor a 0, más ejemplos serán considerados como clasificados correctamente y por lo tanto el conjunto final de ejemplos seleccionados será más pequeño. Es decir, valores pequeños de β conducen a conjuntos de datos más condensados. Finalmente tendremos los ejemplos seleccionados durante el proceso de condensación en S.

12.5 Resultados experimentales

Para comprobar el funcionamiento de las técnicas de imputación y de selección de ejemplos propuestas, vamos a mostrar algunos resultados de su aplicación a distintos conjuntos de datos cuando imputamos valores missing. El proceso de imputación se llevará a cabo a partir de conjuntos de datos de baja calidad y de estos mismos conjuntos de datos condensados usando la técnica CNN_{LQD}.

12.5.1 Marco Experimental

La función $d_{LQD}(\cdot,\cdot)$ utilizada es:

$$f_1(e_j^i, e_h^i) = \frac{\sqrt{\frac{(a-a')^2+(b-b')^2+(c-c')^2+(d-d')^2}{4}}}{max_i - min_i}$$

donde e_j^i, e_h^i son valores numéricos de los ejemplos j y h en el atributo i definidos por las cuádruplas (a,b,c,d) y (a',b',c',d') respectivamente y

$$f_2(e_j^i, e_h^i) = 1 - \frac{Card(e_j^i \bigcap e_h^i)}{Card(e_j^i \bigcup e_h^i)}$$

donde e_j^i, e_h^i son valores nominales y $Card(e_j^i \bigcap e_h^i)$ y $Card(e_j^i \bigcup e_h^i)$ están definidas como la cardinalidad de los subconjuntos crisp/fuzzy que resultan de la unión e intersección de e_j^i y e_h^i respectivamente.

La medida de entropía utilizada es la definida en [60]:

$$EntropiaF(\mu(\cdot)) = -\sum_{j=1}^{n}(\mu(cf_j)\log(\mu(cf_j)) + (1-\mu(cf_j)(\log(1-\mu(cf_j)))$$

donde $\mu(\cdot)$ denota un subconjunto crisp/fuzzy y cf_j los valores pertenecientes al subconjunto.

Usamos los valores $k = \sqrt{|E|}$, $\alpha = 0$ y $\beta = 1$ en todos los experimentos. La Tabla 12.1 muestra los conjuntos de datos utilizados [77], indicando el número de ejemplos ($|E|$),

el número de atributos ($|A|$) (entre paréntesis se muestran el número de atributos numéricos y nominales) y el número de clases ($|C|$) para cada conjunto de datos. "Abr" indica la abreviatura del conjunto de datos usada en los experimentos.

Tabla 12.1: Conjuntos de datos

Conjuntos de datos	**Abr**	**|E|**	**|A|**	**I**	**Conjunto de datos**	**Abr**	**|E|**	**|A|**	**|C|**
Iris Plants	IRP	150	5 (4,1)	3	Contraceptive M.	CMC	1473	10 (9,1)	3
W. Breast C. (org)	BCW	683	10 (9,1)	2	Attitude Smok.	SMO	2855	9 (2,7)	3
Pima Indian Diab.	PIM	768	9 (8,1)	2	Waveform	WAV	4999	22 (21,1)	3
German Credit	GER	1000	25 (24,1)	2					

En estos conjuntos de datos hemos introducido datos de baja calidad usando la herramienta NIPip [39]. En concreto, hemos introducido un 5 % de valores fuzzy y un 5 % de intervalos en cada atributo numérico y un 5 % de subconjuntos fuzzy en cada atributo nominal incluida la clase. Una vez añadidos los datos de baja calidad, el conjunto de datos es particionado en los conjuntos test y train (20 % y 80 % respectivamente). En los experimentos de clasificación obtenemos la clase de todos los ejemplos del test y en los experimentos de regresión estimamos el valor del primer atributo numérico de todos los ejemplos del test. Este proceso se repite tres veces y por lo tanto los resultados muestran siempre la media de las tres repeticiones.

Dado que la clase imputada a un ejemplo puede ser un subconjunto fuzzy, para poder medir la precisión de clasificación de las técnicas transformamos dicha clase imputada en un subconjunto crisp.

Sea un ejemplo e con clase imputada el subconjunto fuzzy $\{\mu(e,c_1)/c_1, \mu(e,c_2)/c_2, \ldots, \mu(e,c_{|C|})/c_{|C|}\}$. Dicho subconjunto se transforma en un subconjunto crisp de la siguiente forma: Si c_{max} es la clase con mayor grado en el subconjunto fuzzy, el subconjunto crisp se obtiene como

$$clase(e)_{imp} = \{c | \frac{\mu(e,c_{max}) - \mu(e,c)}{\mu(e,c_{max})} < \gamma\} \cup \{c_{max}\}$$

Una vez obtenido el subconjunto crisp, para obtener los resultados de clasificación, aplicamos el proceso de decisión mostrado en el Algoritmo 26.

A partir del Algoritmo 26, construimos el intervalo de precisión en clasificación como $[min\_acierto, max\_acierto]$ donde $min\_acierto$ se calcula considerando sólo los aciertos (variable $acierto$) y $max\_acierto$ es calculado considerando como aciertos la suma de las variables $acierto + acierto\_error$.

En la imputación de atributos numéricos de un ejemplo e, de nuevo, el valor imputado e^j_{imp} puede ser un valor de baja calidad. Calculamos el error de imputación como el valor medio de los valores $f_1(e^j, e^j_{imp})$.

Algoritmo 26 Decisión en la clasificación

for all ejemplo e de E_{test} **do**
  if $(clase(e) == clase(e)_{imp})$ **then** acierto++;
  else if $(clase(e) \bigcap clase(e)_{imp} \neq \emptyset)$ **then** acierto_error++;
    else error++;
  end if
end for

12.5.2 Precisión en la imputación

En la Tabla 12.2 se muestran los resultados obtenidos en los experimentos. La tabla muestra la imputación a partir de los conjuntos de datos originales y la imputación a partir de los mismos conjuntos de datos condensados. Las columnas "Cl" y "Re" muestran la imputación en clasificación y regresión respectivamente. En "Cl" utilizamos tres posibles valores de γ para el Algoritmo 26. La última columna "TR" muestra la tasa de reducción de ejemplos tras la condensación.

Tabla 12.2: Resultados experimentales para los conjuntos de datos de la Tabla 12.1

	Sin Condensación				**Con Condensación**				
	Cl			**Re**	**Cl**			**Re**	**TR**
	$\gamma = 5\%$	$\gamma = 10\%$	$\gamma = 20\%$		$\gamma = 5\%$	$\gamma = 10\%$	$\gamma = 20\%$		
IRP	[97.8,97.8]	[97.8,97.8]	[95.6,100]	0.30	[97.8,97.8]	[97.8,97.8]	[96.7,97.8]	0.32	64.4
BCW	[97.1,97.1]	[96.8,97.1]	[96.8,97.1]	1.68	[97.1,97.1]	[97.1,97.1]	[96.8,98.0]	1.81	59.1
PIM	[75.4,76.0]	[71.2,78.7]	[70.4,79.3]	2.14	[75.6,75.8]	[67.1,80.4]	[63.8,81.3]	2.08	41.9
GER	[70.8,75.2]	[70.7,75.3]	[66.0,81.2]	1.04	[67.2,76.5]	[66.8,76.5]	[57.7,84.8]	1.07	40.8
CMC	[42.3,51.1]	[37.8,55.8]	[28.8,65.2]	5.80	[42.3,51.1]	[37.7,55.8]	[28.8,65.2]	5.75	22.5
SMO	[68.6,68.7]	[68.6,68.7]	[68.0,68.9]	0.36	[68.6,68.7]	[68.6,68.7]	[68.0,68.9]	0.37	31.6
WAV	[82.9,86.4]	[81.4,87,6]	[77.9,89.9]	0.82	[83.2,86.1]	[79.8,88.6]	[76.1,91.3]	0.82	46.6

Como podemos observar en la Tabla 12.2, los resultados obtenidos muestran un comportamiento muy estable de las técnicas propuestas cuando trabajan con LQD. Estos resultados se ajustan a los obtenidos en la literatura para estas mismos conjuntos de datos sin valores de baja calidad. Además, podemos observar en la imputación del atributo clase cómo al aumentar γ, el intervalo del porcentaje medio de precisión es más amplio. Esto nos indica la falta de decisión a la hora de elegir un valor para imputar. Comparando los resultados obtenidos "sin" y "con" condensación podemos considerar que la condensación ha funcionado correctamente obteniendo unos resultados de precisión buenos con un número de ejemplos reducido considerablemente.

CAPÍTULO

13

Conclusiones parciales y aportaciones

13.1 Conclusiones

A lo largo de los capítulos que componen esta parte hemos propuesto el diseño y la extensión de técnicas que se engloban dentro de la fase de preprocesamiento de datos del AID. Las técnicas propuestas llevan a cabo distintos procesos de preprocesamiento trabajando con conjuntos de datos que contienen explícitamente LQD. Concretamente, los procesos abordados en esta parte son la discretización de atributos numéricos en particiones fuzzy, la selección de atributos, la imputación de valores missing y la selección de ejemplos.

En el Capítulo 10 hemos presentado una técnica de discretización de valores numéricos (OFP_CLASS). Sobre esta técnica se han presentado dos modificaciones: 1) para que la técnica pueda manejar y trabajar de forma explícita con valores de baja calidad OFP_CLASS$_{LQD}$, y 2) para mejorar la calidad de las particiones fuzzy cuando los conjuntos de datos contienen un número de ejemplos pequeño en relación con el número de atributos y número de clases de los ejemplos, BAGOFP_CLASS. La técnica de discretización OFP_CLASS propuesta es una técnica híbrida compuesta por un árbol de decisión fuzzy que obtiene un conjunto de puntos que forman una partición crisp inicial y un algoritmo genético que a partir de los puntos de corte anteriores, los optimiza y genera una partición fuzzy. De esta forma, la técnica OFP_-CLASS puede ser usada para generar tanto una partición crisp como fuzzy. Para poder trabajar con LQD, OFP_CLASS$_{LQD}$, toma como base el árbol FDT$_{LQD}$ propuesto en el Capítulo 6 con algunas modificaciones. Para mejorar la calidad de las particiones, BAGOFP_CLASS, añade un

proceso de bagging tanto en el árbol construido para aportar estabilidad como en el fitness del algoritmo genético posterior. Para evaluar la técnica de discretización en sus distintas versiones, hemos llevado a cabo un serie de experimentos tanto con bases de datos reales relacionadas con el atletismo de alto rendimiento y la dyslexia como con bases de datos del repositorio de la UCI, [77]. Entre los experimentos realizados hemos llevado a cabo una comparación de OFP_CLASS con otras técnicas de la literatura, concluyendo que OFP_CLASS es robusto y con un rendimiento competitivo con el resto de técnicas siendo mejor que algunas de ellas. Otro experimento realizado se ha centrado en analizar el comportamiento de $\text{OFP\_CLASS}_{\text{LQD}}$ con conjuntos de datos con LQD concluyendo que obtiene resultados competitivos con otras técnicas de la literatura y mejores que los obtenidos con OFP_CLASS al imputar/reemplazar los valores con LQD. Por último, hemos realizado una serie de experimentos orientados a analizar la mejora obtenida con BAGOFP_CLASS frente a las versiones anteriores, concluyendo que BAGOFP_CLASS es la mejor técnica.

Tras el capítulo de discretización, en el Capítulo 11 hemos propuesto una técnica para seleccionar atributos desde conjuntos de datos con LQD, FRF-fs. La técnica propuesta es una técnica híbrida que combina una técnica de filtrado con una técnica wrapper, con el fin de seleccionar un subconjunto de atributos óptimos de un conjunto de datos que contiene LQD. En el diseño de la técnica hemos utilizado información proporcionada por el ensamble FRF_{LQD} presentado en el Capítulo 7 como medida de importancia de los atributos. Posteriormente utilizamos un operador OWA para ordenar toda la información proporcionada por el ensamble y así poder crear un ranking de importancia de los atributos y posteriormente obtener el subconjunto de atributos óptimo. Para validar y analizar el comportamiento de FRF-fs hemos realizado diversos experimentos con conjuntos de datos de alta dimensionalidad y podemos concluir que la técnica es robusta y que los resultados obtenidos comparados con otras técnicas de la literatura son bastante competitivos, teniendo como ventaja además de los resultados, la capacidad de tratar LQD. Por último, hemos aplicado FRF-fs a conjuntos de datos de microarrays, realizando un estudio más detallado de los resultados a nivel de precisión y podemos concluir que la técnica tiene un buen comportamiento a la hora de seleccionar genes de los conjuntos de datos de microarrays, no solo obteniendo una alta precisión, sino también aumentando la interpretabilidad de los resultados.

Por último, en el Capítulo 12, hemos realizado la extensión de la técnica del vecino más cercano para clasificar para llevar a cabo un proceso de imputación de valores missing y la extensión de la técnica de condensación CNN (CNN_{LQD}). La técnica de imputación propuesta tiene la capacidad de poder trabajar con conjuntos de datos que contienen valores de baja calidad de forma explícita en los datos. Hemos propuesto una medida distancia para poder calcular los vecinos más cercanos a un ejemplo dado cuando en los valores de los ejemplos aparece

LQD. Un aspecto a comentar sobre la imputación KNN_{LQD} es que cuando se lleva a cabo la imputación de un valor missing, el valor resultante no tiene porqué ser crisp sino que se puede obtener cualquier otro valor de baja calidad, siempre dependiendo del resto de valores que componen el atributo que se está imputando, por lo tanto, de esta manera se respeta durante el proceso de imputación la información que el resto de atributos aportan. Para mejorar los problemas computacionales de la técnica cuando se trabaja con grandes conjuntos de datos, hemos propuesto una primera aproximación para llevar a cabo la selección de ejemplos con LQD, concretamente hemos extendido una técnica de condensación de ejemplos. Para evaluar las técnicas propuestas hemos realizado una serie de experimentos con conjuntos de datos de la UCI a los que hemos añadido LQD, comparando los resultados de imputación obtenidos utilizando y sin utilizar la técnica de condensación CNN_{LQD}, obteniendo unos primeros resultados preliminares bastante optimistas. Dado que la técnica KNN_{LQD} se encuentra en una fase inicial de desarrollo como comentamos anteriormente, queda como vía abierta de futuros trabajos el análisis y estudio de diversas funciones distancias que se adapten a los diversos tipos de LQD con los que trabajamos, la realización de comparaciones experimentales, adaptación de otras técnicas de selección de ejemplos y análisis experimental de sus resultados, etc.

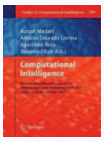
Como conclusión, las técnicas de preprocesamiento de datos que hemos propuesto durante esta parte mejoran la calidad de los conjuntos de datos tanto si contienen como si no contienen LQD. De forma general, llegamos a la conclusión de que adaptando o diseñando técnicas que puedan tratar con valores de baja calidad podemos mejorar la precisión de los resultados ya que se mantiene una mayor cantidad de información relevante en el conjunto de datos, que puede ser útil para la aplicación posterior de una técnica de minería de datos. Por último, debemos de destacar la calidad de los resultados obtenidos por las distintas técnicas propuestas en comparación con otras técnicas de la literatura, teniendo como característica adicional el tratamiento de conjuntos de datos con LQD. Además, hemos comprobado la versatilidad en la extensión del tratamiento de nuevo tipos de LQD en todos ellos.

13.2 **Aportaciones más relevantes**

[43] J.M. Cadenas, M.C. Garrido, R. Martínez, P.P. Bonissone. OFP_CLASS: A hybrid method to generate Optimized Fuzzy Partitions for CLASSification, Soft Computing 16(4), 667–682, 2012.

JCR 1.124

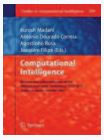
[33] J.M. Cadenas, M.C. Garrido, R. Martínez, Generating Optimized Fuzzy Partitions to Classification and Considerations to Management Imprecise Data, Computational Intelligence (Studies in Computational Intelligence) 399, 152–166, 2012.



Capítulo de Libro

[30] J.M. Cadenas, M.C. Garrido, R. Martínez. Constructing fuzzy partitions from imprecise data. International Conferente on Fuzzy Computation Theory and Applications (FCTA 2011), 379–388, Paris, Francia, 2011. Best student paper award 2011. http://www.fcta.ijcci.org/PreviousAwards.aspx

[37] J.M. Cadenas, M.C. Garrido, R. Martínez. Generating Fuzzy Partitions from Nominal and Numerical Attributes with Imprecise Values. Computational Intelligence (Studies in Computational Intelligence) 465, 167–182, 2013.

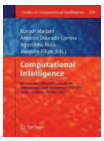


Capítulo de Libro

[38] J.M. Cadenas, M.C. Garrido, R. Martínez. Improving a fuzzy discretization process by bagging. International Conferente on Fuzzy Computation Theory and Applications (FCTA 2013), 201–212, Vilamoura, Portugal, 2013.

[46] J.M. Cadenas, M.C. Garrido, A. Muñoz-Ledesma, R. Martínez. BAGIM-OFP: Particionamiento Fuzzy de Atributos desde Datos Imperfectos en Datasets con Pocos Ejemplos. XVII Congreso Español sobre Tecnologías y Lógica Fuzzy Modelos de optimización y Soft Computing (ESTYLF2014), 235-240, 2014.

[40] J.M. Cadenas, M.C. Garrido, R. Martínez. Fuzzy Discretization Process from small datasets. Computational Intelligence (Studies in Computational Intelligence) 2014. Pend. publicación.



Capítulo de Libro

[36] J.M. Cadenas, M.C. Garrido, R. Martínez. Feature subset selection Filter-Wrapper based on low quality data. Expert Systems with Applications 40, 6241–6252, 2013.

JCR 1.965

[35] J.M. Cadenas, M.C. Garrido, R. Martínez. Towards an Approach to Select Features from Low Quality Datasets. International Conference on Fuzzy Computation Theory and Applications (FCTA 2012), 357–366, Barcelona, España, 2012. Best Paper awards 2012. http://www.fcta.ijcci.org/PreviousAwards.aspx

[48] J.M. Cadenas, M.C. Garrido, R. Martínez, D. Pelta, P.P. Bonissone. Using a Fuzzy Ensamble for Tumor Classification from Gene Expresion Data. International Conferente on Fuzzy Computation Theory and Applications (FCTA 2013), 320–331, Vilamoura, Portugal, 2013.

[47] J.M. Cadenas, M.C. Garrido, R. Martínez, D. Pelta, P.P. Bonissone. Gene Priorization for Tumor Classification using an Embedded Method. Computational Intelligence (Studies in Computational Intelligence), 2014. Pend. publicación.

Capítulo de Libro

[143] R. Martínez, J.M. Cadenas, M.C. Garrido, A. Martínez. Imputing missing values from Low Quality Data by NIP tool. IEEE International Conference on Fuzzy System (FUZZIEEE-2013), 1–8, Hyderabad, India, 2013.

Congreso CORE A

[45] J.M. Cadenas, M.C. Garrido, R. Martínez, A. Martínez. Regla K_M-vecinos más cercanos en bases de datos de baja calidad. Multiconference CAEPIA. Actas del VI Simposio de Teoría y Aplicaciones de Minería de Datos (TAMIDA 2013), 1303–1312, Madrid, Spain, 2013.

Parte IV

UNA HERRAMIENTA SOFTWARE PARA EL TRABAJO CON DATOS DE BAJA CALIDAD

En esta última PARTE IV vamos a presentar una herramienta software que facilita el trabajo en la investigación con datos de baja calidad. En concreto, la herramienta engloba los procesos de preprocesamiento de discretización e imputación. La herramienta software que proponemos, “NIP imperfection processor” (NIPip) es de código abierto y se centra en la fase del AID de preprocesamiento de datos. A pesar de que podemos encontrar una gran cantidad de herramientas tanto de código abierto como privadas que llevan a cabo el proceso de limpieza y preparación de los datos, no existe una herramienta centrada en el preprocesamiento de datos que contienen valores de baja calidad. NIPip esta diseñada en dos paquetes, un primer paquete más interactivo donde se puede preprocesar un conjunto de datos cada vez y un segundo paquete más dedicado a la creación de experimentos (EXpNIPip), que permite el preprocesamiento de una gran cantidad de conjuntos de datos simultáneamente.

En el Capítulo 14, vamos a describir de forma detallada los dos paquetes de la herramienta NIPip exponiendo todas las propiedades y características de la misma. Además para cada paquete de la herramienta se muestran un conjunto de gráficos representativos del funcionamiento de la herramienta. La idea general de la herramienta es definir un marco de trabajo para poder crear y manipular diferentes conjuntos de datos con LQD.

El esquema general de desarrollo de esta parte, se muestra en la Figura 13.1.

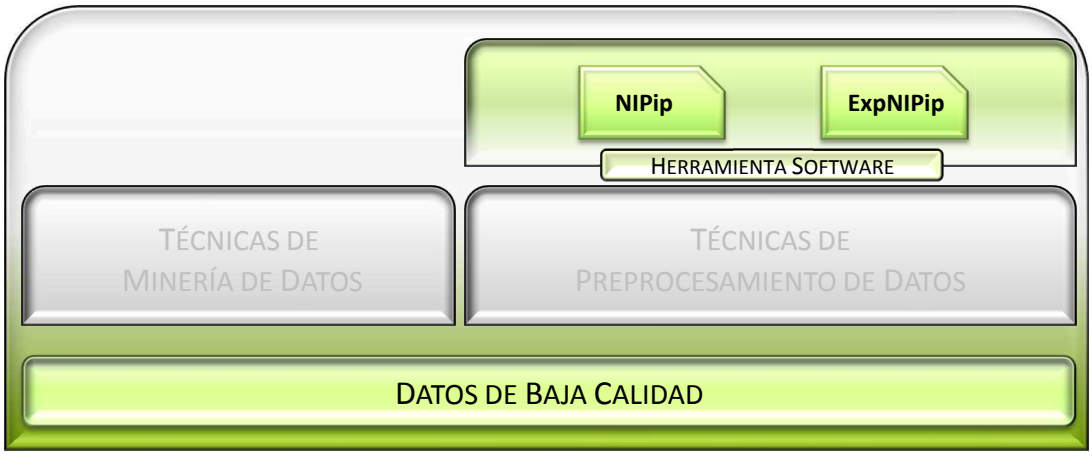


Figura 13.1: Esquema general de la parte

CAPÍTULO

14

Una herramienta software para preprocesar datos de baja calidad

14.1 Introducción

A lo largo de todo el trabajo hemos comentado que la información disponible en muchos problemas del mundo real no es de tanta calidad como sería deseable. Cuando tratamos de abordar esta información desde el AID se pueden adoptar alternativas diferentes. Por un lado, podemos transformar los LQD, con los que ciertas técnicas del AID no pueden trabajar, por otros tipos de valores intentando no perder mucha información en el proceso de transformación. En este caso, sería deseable disponer de una herramienta software capaz de realizar esta transformación. Por otro lado, se pueden diseñar o adaptar ciertas técnicas del AID para que puedan trabajar con LQD. En este caso nos encontramos con el problema de que no hay conjuntos de datos con LQD públicos que faciliten la validación de dichas técnicas y conocer el comportamiento de los mismos cuando se trata con diferentes niveles de LQD. En este caso también sería deseable la existencia de una herramienta software capaz de crear o modificar conjuntos de datos con las características deseadas por el investigador en el ámbito de trabajar con información de baja calidad.

Actualmente hay diferentes herramientas software en el área del AID. Aunque muchas de ellas son comerciales, el número de herramientas "open source" está creciendo considerablemente. Sin embargo, analizando las diferentes herramientas, podemos decir que no existe

actualmente una que sea capaz de tratar explícitamente con LQD, llevando a cabo un preproceso de estos conjuntos de datos, bien transformándolos para sustituir los LQD por otros datos, o bien modificándolos para incorporarles LQD con el fin de poder testear el diseño de nuevas técnicas que trabajan con este tipo de datos.

Por este motivo hemos diseñado una herramienta capaz de tratar explícitamente con LQD. Esta herramienta facilita el trabajo en el campo del AID permitiendo la generación y transformación de conjuntos de datos con LQD. La generación y transformación de estos conjuntos de datos puede servir como un marco de trabajo común para llevar a la cabo la comparación de diferentes técnicas desarrolladas en el marco del AID para el tratamiento de LQD.

La herramienta se compone de dos paquetes complementarios. Un primer paquete llamado "NIP imperfection processor" (que denotamos por NIPip), [39], [49], [34], permite llevar a cabo las transformaciones de los datos paso a paso, de forma que se puede comprobar visualmente cómo se va transformando; y un segundo paquete centrado en el entorno "Experimenter" (que denotamos por ExpNIPip), [142], enfocado a la realización de experimentos con grandes cantidad de conjuntos de datos el cual permite realizar, de manera más flexible, el preproceso de forma online desde la herramienta o de forma offline mediante un script.

El resto del capítulo está organizado de la siguiente manera: vamos a llevar a cabo una comparación de las diferentes herramientas software "open source" que podemos encontrar en el área del AID, centrándonos en las ventajas e inconvenientes que nos ofrecen desde el punto de vista del preprocesado de LQD y, por supuesto, detallaremos los dos paquetes de esta herramienta y mostraremos algunos casos de uso de ellas.

14.2 Software y preprocesamiento de datos de baja calidad

En esta sección vamos a revisar las principales características de algunas herramientas software para el AID pero desde el punto de vista del preprocesamiento de LQD. En esta sección solamente vamos a exponer las principales características de software libre, aunque hay herramientas de software privado que llevan a cabo preprocesamiento de datos como por ejemplo Knowledge Seeker[8] o Simulink Design Optimization[144], entre otros.

- Sodas2 [66]: Es una herramienta que soporta el análisis simbólico de datos. La ejecución de Sodas2 nos permite construir un conjunto de datos simbólicos que resume la información de un conjunto de datos inicial y después de esto, se puede llevar a cabo el análisis simbólico de ellos. El uso de datos simbólicos permite introducir diferentes tipos de LQD, como atributos multivaluados con y sin pesos e intervalos. Estos tipos de atributos pueden representar a otros tipos de atributos, como por ejemplo datos fuzzy. Sin embargo, Sodas2 no permite:

1. El uso directo de tecnología fuzzy;
2. La introducción de valores missing o ruido;
3. La modificación de los datos para introducir algún tipo de imperfección en ellos.

- Weka [91]: Proporciona un conjunto de técnicas de preprocesamiento de datos, de clasificación, regresión, clustering, reglas de asociación y visualización. Para el preprocesamiento de datos, Weka ofrece una amplia variedad de técnicas de atributos y ejemplos. Si nos centramos en el caso de tratamiento de LQD, el número de técnicas se reduce ofreciéndonos menos oportunidades. Weka solamente es capaz de manejar valores missing y proporciona técnicas para reemplazar/imputar por la moda y la media. Además también permite añadir ruido en atributos nominales.
- Keel [4, 5]: Es una herramienta software para el AID. Contiene una larga colección de técnicas clásicas basadas en la extracción de conocimiento, preproceso y aprendizaje computacional, modelos híbridos y un módulo de tests estadísticos para comparaciones. Keel permite el uso de diferentes formatos de entrada y salida como CSV, XML o ARFF. El módulo de manejo de datos incluye una amplia variedad de técnicas de selección de ejemplos, de selección de atributos, discretización, etc. Pero al igual que en Weka, hay pocas técnicas que permiten el manejo de LQD, ya que solo permite el uso de valores missing en varias técnicas de imputación como por ejemplo con K-vecinos más cercanos o la imputación mediante K-medias.
- Rapid miner [146]: Esta herramienta (formalmente llamado Yale) trabaja en un ambiente de aprendizaje computacional y en el paradigma del AID intentando dar soporte a la creación rápida de prototipos. Así, Rapid miner ofrece la posibilidad de utilizar diferentes formatos de entrada y salida. El número de técnicas de preproceso que ofrece la herramienta es amplio, pero de nuevo, si nos centramos en el manejo de LQD el número de técnicas disminuye. La herramienta puede manejar valores missing usando múltiples técnicas de imputación, tales como reemplazar por la media, la moda, el máximo, el mínimo o el valor predicho por una técnica del AID definida por el usuario. También permite la introducción de ruido en atributos numéricos y nominales.
- MiningMart [153]: Se desarrolló con el propósito de reusar técnicas de preprocesamiento de grandes conjuntos de datos. MiningMart no está centrado en el proceso de descubrimiento de conocimiento, solamente se centra en la parte de preproceso. La herramienta ofrece agregación de técnicas de discretización, limpieza de datos, tratamiento con valores missing y selección de atributos relevantes. El único tipo de datos que permite manejar son los valores missing, permitiendo borrar ejemplos que los contengan o reemplazarlos por la moda, media o por una técnica de imputación.

Existen otras herramientas centradas en el AID como Adam [175], E2K [135], Knime [16], Orange [63] o Tanagra [171], ..., pero éstas no ponen atención en el tratamiento de LQD.

La Tabla 14.1 muestra un resumen de las principales características respecto al preprocesamiento de LQD de las herramientas de software que se han presentado. Además, en la última columna se muestran las caracterśticas asociadas a la herramienta diseñada. En esta tabla se refleja con "Y" cuando la herramienta ha desarrollado completamente la característica descrita en la fila correspondiente, se denota con "P" cuando la herramienta tiene parcialmente desarrollada la característica y se denota con "N" cuando la herramienta no desarrolla la característica.

Tabla 14.1: Características de herramientas software con respecto al preprocesamiento de datos

Característica	**S**odas2	**W**eka	**K**eel	**R**apid Miner	**M**iningMart	**H**erramienta **d**iseñada
Formato:						
Estándar	P	Y	Y	Y	P	Y
Custom	N	N	N	N	N	Y
Datos Baja Calidad:						
Valor Missing	N	Y	Y	P	N	Y
Valor Intervalar	N	N	N	N	N	Y
Valor Fuzzy	N	N	N	N	N	Y
Partición Crisp	Y	Y	Y	Y	Y	Y
Partición Fuzzy	N	N	N	N	N	Y
Subconjunto Crisp	N	N	N	N	N	Y
Subconjunto Fuzzy	N	N	N	N	N	Y
Imputación/Remplazamiento:						
Valor Missing	Y	Y	Y	P	P	Y
Valor Intervalar	N	N	N	N	N	Y
Valor Fuzzy	N	N	N	N	N	Y
Ruido en Atributos:						
Nominal	N	Y	N	Y	N	Y
Numéricos	N	N	N	Y	N	Y
Selección:						
Atributos	Y	Y	Y	Y	Y	N
Ejemplos	Y	Y	Y	Y	Y	N

14.3 El paquete NIPip para manejar datos de baja calidad

NIPip, [39], es una herramienta software para manejar y generar conjuntos de datos con LQD. El principal propósito de este paquete es establecer un marco común para este tipo de datos

en la investigación en el área del AID (debido a la ausencia de un software similar). NIPip está compuesto de los tres módulos siguientes:

- Módulo de Importación: Este módulo se compone de un conjunto de elementos funcionales para importar e iniciar el preproceso de un conjunto de datos. NIPip puede importar conjuntos de datos con un formato estándar (ARFF, KEEL, UCI, CSV) o con un formato definido por el usuario (formato "custom"). El preproceso inicial incluye borrar, modificar el tipo y ordenar los atributos, además de poder normalizar los atributos numéricos, entre otras cosas. Adicionalmente, este módulo incluye elementos de extracción de información del conjunto de datos, por ejemplo, número de ejemplos y de atributos, tipos de cada uno de los atributos, la media, la moda, el porcentaje de LQD que contiene el conjuntos de datos importado, etc.
- Módulo de manejo de LQD: El objetivo de este módulo es añadir diferentes tipos de LQD en los distintos atributos. Sobre los atributos numéricos, NIPip permite añadir desde datos missing hasta valores fuzzy, definidos por el usuario o bien obtenidos a través de técnicas de discretización. En el caso de los atributos nominales, se pueden añadir desde valores missing hasta subconjuntos crisp/fuzzy. Este módulo incluye una librería interna de técnicas de discretización de atributos numéricos.
- Módulo de Exportación: Este módulo es una de las partes fundamentales de NIPip. Los conjuntos de datos de salida se construyen en este módulo y está compuesto de los elementos que permiten exportar y reemplazar/imputar. Al igual que el módulo de importación, el conjunto de datos exportado podrá tener un formato estándar o un formato "custom" definido por el usuario. Además el módulo incluye una librería interna con técnicas de reemplazamiento/imputación, [143]. Con estas técnicas el usuario puede crear diferentes muestras de un conjunto de LQD. También se permite exportar un conjunto de datos mediante una validación cruzada de tamaño k. Por último todo el proceso seguido con el conjunto de datos queda recogido en un fichero.

El esquema general del proceso seguido por NIPip a través de los diferentes módulos se refleja en la Figura 14.1.

La Figura 14.2 muestra el diagrama de clases simplificado sobre el que se construye el paquete de NIPip donde se puede apreciar la simplicidad de su diseño y la facilidad y flexibilidad para ampliar su funcionamiento. El diagrama muestra solamente el conjunto de clases principales omitiendo los atributos y métodos.

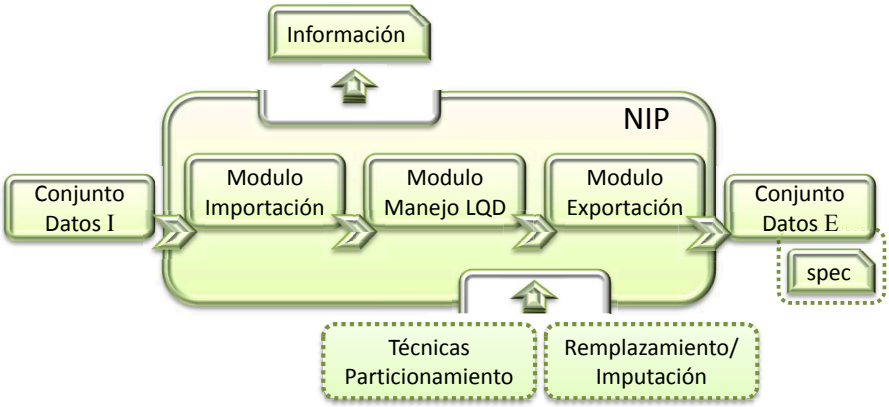


Figura 14.1: NIPip para manejar datos de baja calidad

14.3.1 Funcionalidades

El usuario dispone de un conjunto de funcionalidades propias y específicas en NIPip. Las principales se describen a continuación:

- NIPip proporciona una interfaz fácil de utilizar, orientada al manejo y construcción de conjuntos de LQD.
- Los conjuntos de datos pueden ser importados en formato estándar (ARFF, KEEL, UCI, CSV) o en un formato definido por el usuario.
- Los conjuntos de datos importados pueden contener LQD explícitos.
- Se permite borrar, normalizar, cambiar el tipo de los atributos, ...
- Se permite manejar y añadir diferentes tipos de LQD (missing, valores fuzzy, intervalos, subconjuntos crisp/fuzzy, ruido, etc).
- Contiene una librería interna de técnicas de discretización para atributos numéricos.
- Se proporciona información útil y descriptiva acerca de las características de los conjuntos de datos importados.
- Contiene una librería interna de técnicas de reemplazamiento/imputación de LQD.
- Un conjunto de datos puede ser exportado en formato estándar (ARFF, KEEL, UCI, CSV) y en un formato definido por el usuario.
- Al exportar un conjunto de datos se crea un fichero de información útil y descriptivo que recoge las características del conjunto de datos.
- Es posible exportar los conjuntos de datos en diferentes formatos de salida creando diferentes muestras de los mismos mediante el remplazamiento/imputación de los valores de baja calidad.
- Los conjuntos de datos pueden ser exportados generando una validación cruzada.

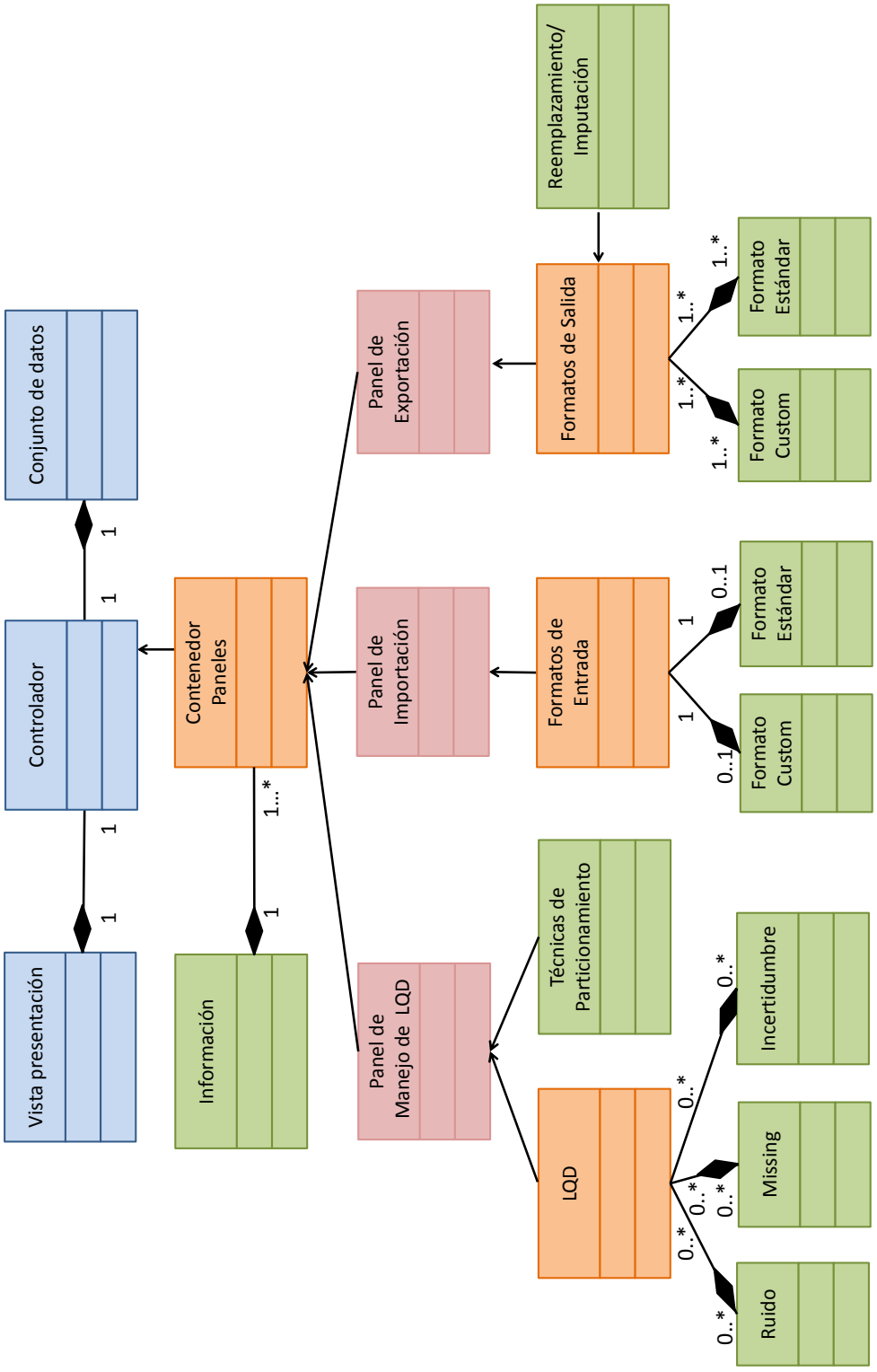


Figura 14.2: Diagrama de clases simplificado de NIPip

Puesto que la característica más diferenciadora de la funcionalidad de NIPip con respecto a otras herramientas software (ver Tabla 14.1) es el manejo de LQD, la Figura 14.3 muestra los tipos de LQD con los cuales puede tratar. Esta figura destaca las operaciones que puede realizar con cada uno: leer (módulo de importación), escribir (módulo de exportación), añadir (módulo de manejo) y reemplazar/imputar (módulo de exportación).

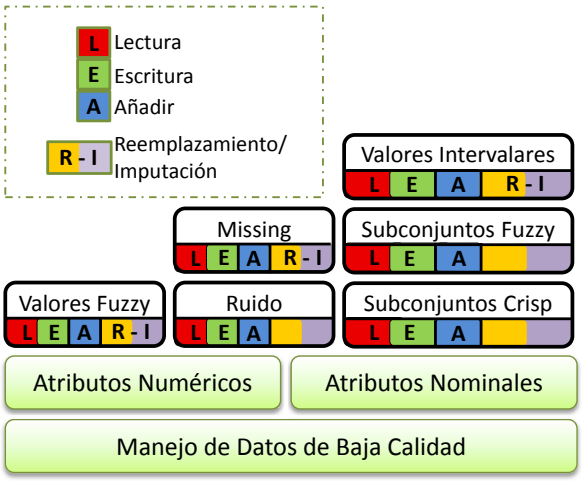


Figura 14.3: Datos de baja calidad manejados por NIPip

Los diferentes módulos de NIPip se encuentran reflejados en el software en todo momento, tal y como se muestra en la Figura 14.4 de seguimiento de los mismos.

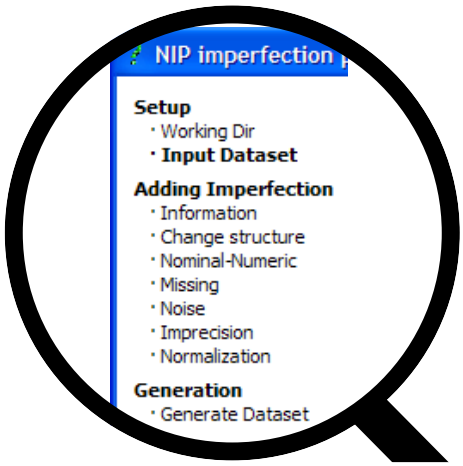


Figura 14.4: Proceso de seguimiento

Las siguientes subsecciones describen con más detalle los módulos presentados, centrándonos en las posibilidades que ofrece NIPip y las metodologías seguidas en cada uno de ellos.

14.3.2 Módulo de importación

La principal función de este módulo es importar conjuntos de datos. En el esquema de funcionamiento general hay que distinguir dos casos:

1. El uso de un formato estándar ARFF, KEEL, UCI, CSV cuando el conjunto de datos a importar incluye LQD permitidos por las técnicas del AID que utilizan tales formatos.
2. El uso de un formato predefinido por el usuario ("custom") cuando el conjunto de datos tiene un formato de entrada estándar pero este incluye LQD los cuales no permiten las distintas técnicas que utilizan este formato predefinido, o cuando el conjunto de datos tiene un formato particular.

En el primer caso, la metodología seguida por NIPip es clara, llevando a cabo un simple análisis interno de acuerdo al formato seleccionado. Por el contrario, en el segundo caso, aunque también se lleva a cabo internamente un análisis, el usuario necesita cooperar con el software a través de la interfaz para definir cómo es el formato. Esta interfaz se muestra en la Figura 14.5, donde se debe de definir cómo se representan los diferentes tipos de LQD.

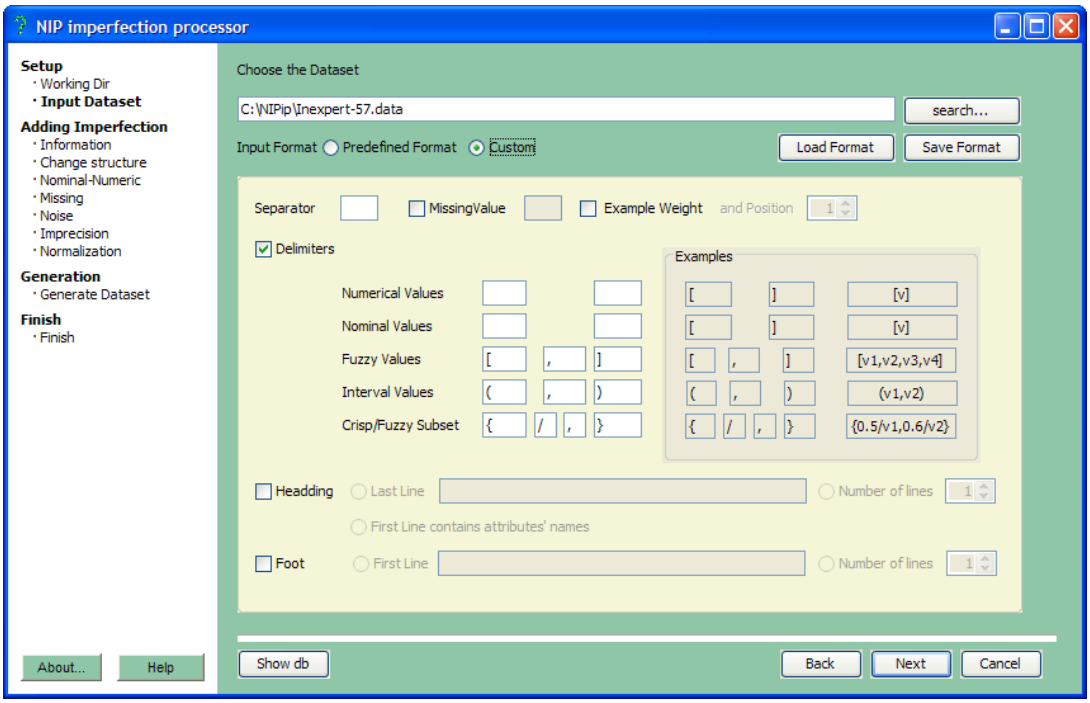


Figura 14.5: Formato de entrada "Custom"

Los valores permitidos son los siguientes:

- Un valor missing debe estar representado por un carácter o conjunto de caracteres.
- Un valor intervalar debe estar descrito por el valor mínimo y máximo del intervalo.

- Un valor fuzzy debe de estar representado por cuatro valores que describen la función de pertenencia trapezoidal asociada.
- Un subconjunto crisp se representa mediante valores nominales que no tienen asociado ningún grado de pertenencia.
- Un subconjunto fuzzy se representa por un conjunto de valores que tienen asociado un grado de pertenencia.

Cuando se importa un conjunto de datos, se calcula toda la información asociada al mismo. Esta información se divide en características generales, información estadística y porcentajes de LQD. Las características generales son, por ejemplo, número de ejemplos, número de atributos y naturaleza de los atributos (nominales o numéricos). Estas características son necesarias para calcular la información estadística que depende de la naturaleza de los atributos del conjunto de datos.

Para los atributos numéricos, la información estadística obtenida está compuesta por la media y el valor máximo y mínimo de cada atributo. La media de un atributo puede expresarse por un valor crisp, un valor intervalar o un valor fuzzy. Este valor será crisp cuando el valor del atributo para todos los ejemplos sea crisp. En este caso también se calcula la desviación estándar como información estadística. El valor de la media será intervalar cuando entre todos los valores del atributo en cuestión exista al menos un valor que sea intervalar. Por último, el valor será fuzzy cuando al menos exista un valor fuzzy en el atributo. La media es calculada de acuerdo al método aritmético.

Para los atributos nominales, la información estadística es la moda y todos los valores que componen el dominio del atributo con la proporción de cada uno de ellos en el conjunto de datos. Puesto que un atributo nominal puede contener valores crisp o valores de subconjuntos crisp/fuzzy, el cálculo de estas proporciones se lleva a cabo de la siguiente forma: para cada valor del dominio, se obtiene su proporción teniendo en cuenta el grado de pertenencia de ese valor en todos los ejemplos del conjunto de datos para el atributo en cuestión. Por ejemplo, si el valor es $\{0.1/a,0.3/b,0.6/c\}$, se considera 0.1 como la proporción para el valor de a, 0.3 para el valor b y 0.6 para el valor c.

Finalmente, si el conjunto de datos contiene LQD, se calcula el porcentaje correspondiente a cada uno de ellos. Así, para cada atributo se calcula el porcentaje de valores fuzzy, intervalares, missing y subconjuntos crisp/fuzzy.

La última funcionalidad de este módulo es la posibilidad de llevar a cabo un preproceso del conjunto de datos si fuera necesario. Este preproceso se centra principalmente en:

- Cambiar la naturaleza de un atributo, el cual puede cambiar de nominal a numérico (siempre que sea posible) o de numérico a nominal.

- Cambiar el orden de los atributos en el conjunto de datos. Además, es posible eliminar aquellos atributos que no se consideren necesarios.
- Normalizar los atributos numéricos para que sus dominios estén entre 0 y 1.

14.3.3 **Módulo de manejo de datos de baja calidad**

Uno de los aspectos más importantes es la capacidad de manejar valores de baja calidad en los conjuntos de datos. Esta funcionalidad permite la generación de experimentos sintéticos los cuales nos permiten medir la robustez de las diferentes técnicas del AID que trabajan con LQD. Con NIPip intentamos generar un repositorio que ayude a los investigadores que también trabajan en esta área.

En este apartado nos centramos en las principales funcionalidades que ofrece NIPip para manejar LQD en conjuntos de datos. Los diferentes tipos de LQD que permite son:

- Valores imprecisos: valores intervalares, valores fuzzy y subconjuntos fuzzy en atributos numéricos; subconjuntos fuzzy y subconjuntos crisp en atributos nominales.
- Valores con ruido: ruido gaussiano en atributos numéricos y ruido aleatorio en atributos nominales.
- Valores missing: tanto en atributo numéricos como en atributos nominales.

La Figura 14.6 muestra la pantalla de NIPip que permite el acceso a las diferentes opciones para añadir LQD.

En esta pantalla, y a través del proceso de modificación del conjunto de datos, se obtiene la información disponible sobre el mismo y la información sobre los porcentajes de LQD que afectan a los diferentes atributos.

Cuando queremos añadir LQD en un conjunto de datos, siempre es posible seleccionar tanto el atributo o atributos a modificar como el porcentaje de valores de cada uno de ellos, excepto para añadir ruido gaussiano, ya que éste afecta al 100 % de los valores crisp del mismo. El software controla el porcentaje total de imperfección de cada atributo en todo momento, para que el porcentaje nunca exceda el 100 % (si esto sucede, se informa al usuario del error). Además, puesto que el conjunto de datos importado puede incluir LQD, los correspondientes porcentajes iniciales también son tenidos en cuenta a la hora de calcular el total de imperfección. Por ejemplo, dado un conjunto de datos con un 50 % de intervalos en un atributo, al ser importado, solamente admite el poder añadirle un 50 % de valores de baja calidad en ese atributo.

En ciertas opciones al añadir LQD se necesitan discretizar los atributos numéricos. En este caso, se permite leer las particiones desde un fichero preparado por el usuario o bien ejecutar una técnica de discretización incluido en la librería interna.

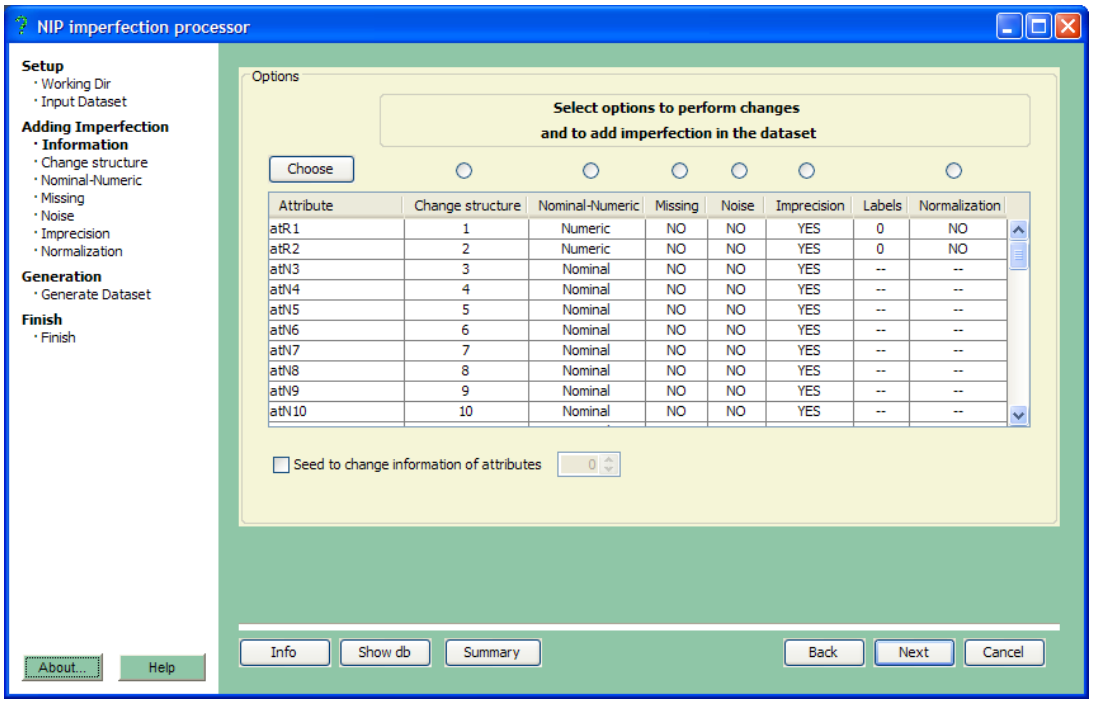


Figura 14.6: Información del conjunto de datos

La primera opción permite el uso de una partición que puede haber sido proporcionada por un experto (en este caso estaríamos introduciendo información de un experto en el conjunto de datos) o mediante una técnica de discretización propia del usuario o proporcionada por el paquete. La segunda opción permite crear un partición automática crisp/fuzzy desde una de las técnicas disponibles en el paquete. Entre las técnicas de discretización fuzzy disponibles se encuentran OFP_CLASS$_{\text{LQD}}$, [43], comentado en capítulos anteriores y una versión del método propuesto en [58]. Además, para obtener particiones crisp también se proporciona una técnica basada en un árbol de decisión, concretamente en el árbol de decisión utilizado en la primera etapa de la técnica OFP_CLASS.

A continuación vamos a detallar las diferentes funcionalidades que ofrece NIPip a la hora de añadir LQD a un conjunto de datos.

14.3.3.1 Valores imprecisos

NIPip puede añadir información imprecisa a un conjunto de datos de las siguientes maneras:

1. Cambiando un valor crisp en un atributo numérico por un valor fuzzy. En este caso es necesario disponer de una partición fuzzy para el atributo que queremos modificar. Una vez que tengamos la partición, el valor fuzzy será aquel de la partición al que el valor crisp pertenezca con mayor grado. Por ejemplo, supongamos que tenemos el valor 0.25

en un atributo de un ejemplo y las particiones asociadas a ese atributo son la siguientes [0.0,0.0,0.30,0.50] y [0.30,0.50,1.0,1.0]. El valor crisp será substituido por el valor fuzzy [0.0,0.0,0.30,0.50] que es al que pertenece con mayor grado.

2. Cambiando un valor crisp de un atributo numérico por un valor intervalar que contenga el valor del atributo. En este caso, se proporciona tres posibilidades: La primera es generar un intervalo añadiendo y restando una cantidad fija al del valor crisp que se quiere modificar (esta opción preserva el valor de la media del atributo); La segunda posibilidad es generar un intervalo añadiendo y restando una cantidad aleatoria por debajo de un valor máximo especificado por el usuario. En ambos casos los valores que forman el intervalo estarán ajustados a los límites del dominio del atributo; La última opción es añadir un intervalo desde una partición crisp usando la librería interna de discretización o usando una partición externa proporcionada por el usuario.
3. Reemplazando un valor numérico crisp por un subconjunto fuzzy de etiquetas (lingüísticas) de una partición del atributo. Dada una partición del atributo, el valor numérico se reemplaza por un subconjunto de etiquetas de la partición a las cuales este valor pertenece con un grado mayor que 0. Por ejemplo, un valor crisp de 45 que pertenece al atributo temperatura podría ser reemplazado por un subconjunto fuzzy $\{0.1/templado, 0.9/caliente\}$ si el valor 45 pertenece a la etiqueta "templado" con un grado de 0.1 y a la etiqueta "caliente" con un grado de 0.9. Por lo tanto, para añadir este tipo de valor también se necesita una partición fuzzy de los atributos numéricos.
4. Reemplazando un valor nominal crisp por un subconjunto crisp/fuzzy del conjunto de valores del dominio de ese atributo. Si el valor nominal se reemplaza por un subconjunto crisp, todos los demás valores del dominio se añaden con una probabilidad igual a la proporción de cada valor en el conjunto de datos. Si el valor se reemplaza por un subconjunto fuzzy, el grado de pertenencia del actual valor será su proporción en el conjunto de datos y los otros valores se añadirán de la misma forma que el caso de un subconjunto crisp pero incluyendo adicionalmente en este caso la proporción del mismo en el conjunto de datos como grado de pertenencia.

Además, en este módulo también se puede crear una partición crisp o fuzzy de los atributos numéricos sin tener que utilizarla para añadir imprecisión al conjunto de datos. La Figura 14.7 muestra un ejemplo de la pantalla para añadir LQD.

14.3.3.2 **Valores con ruido**

NIPip también permite introducir ruido tanto en atributos numéricos como en nominales. En el caso de los atributos numéricos es necesario indicar para cada atributo (al que se le quiere

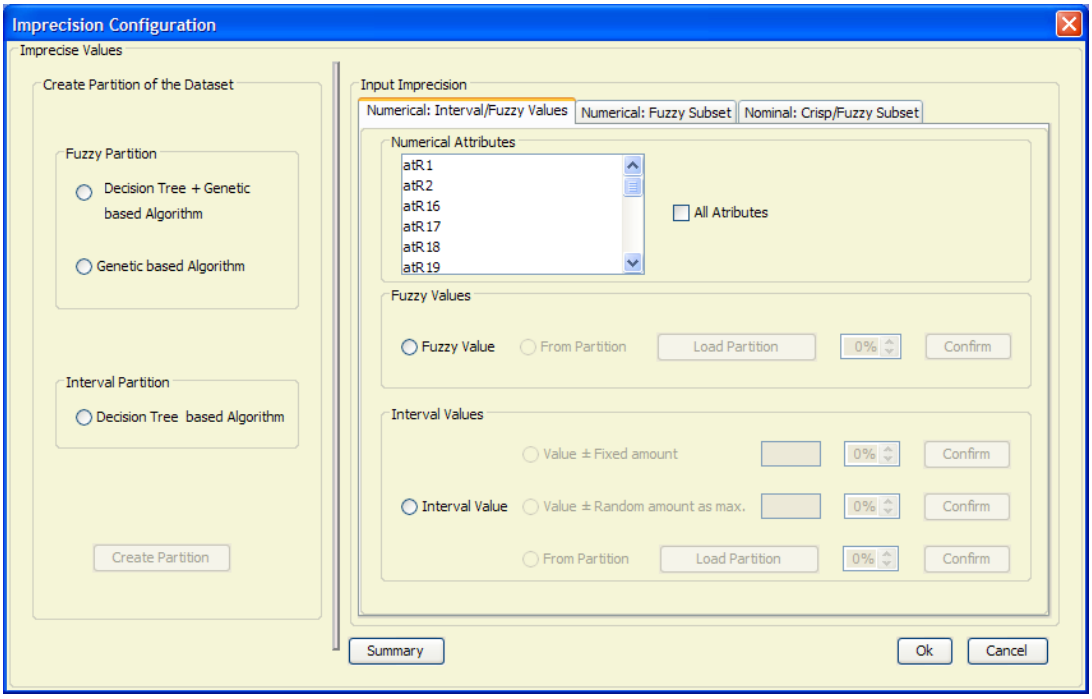


Figura 14.7: Añadiendo LQD

introducir ruido) la media y la desviación estándar para simular el error cometido al medir el valor del atributo. Después de especificar estos valores, el ruido afectará al 100 % de los valores crisp del atributo, y por tanto, no será posible añadir otro tipo de valor.

En el caso de los atributos nominales, NIPip sigue el esquema de añadir ruido utilizado por algunos autores como [219]. Para ello, solamente es necesario expresar el porcentaje de ruido para indicar el número de valores del atributo que serán modificados aleatoriamente por otro valor del dominio. Con el fin de introducir un nivel de ruido del x % en el atributo a, se eligen aproximadamente el x % de los ejemplos y al valor de a de cada uno de estos ejemplos se le asigna un valor aleatorio perteneciente al dominio del atributo siguiendo una distribución uniforme. El porcentaje de ruido puede ser menor del deseado debido a que la asignación aleatoria puede elegir el mismo valor original.

14.3.3.3 **Valores missing**

Otra posibilidad de añadir LQD en el conjunto de datos es mediante la introducción de valores missing. El añadir valores missing se realiza siguiendo una distribución aleatoria.

14.3.4 **Módulo de exportación**

Después de añadir LQD a un conjunto de datos, se debe de trabajar con el módulo de exportación. La principal funcionalidad de este módulo es exportar el conjunto de datos en el formato deseado. Se permiten múltiples salidas y cada una de ellas en un formato diferente. Además, existe la posibilidad de generar una partición del conjunto de datos resultante para validación cruzada. En este último caso, se generará el conjunto de datos completo y las particiones correspondientes a los ficheros test y train para una validación cruzada de tamaño k, donde k es un parámetro configurable. Todos los ficheros de salida se organizan en una estructura de directorios desde el directorio inicial (seleccionado por el usuario). Diferentes formatos de exportación están disponibles, desde formatos estándares (ARFF, KEEL, UCI y CSV) hasta la posibilidad de formatos definidos por el usuario. La Figura 14.8 muestra la pantalla de un formato de salida.

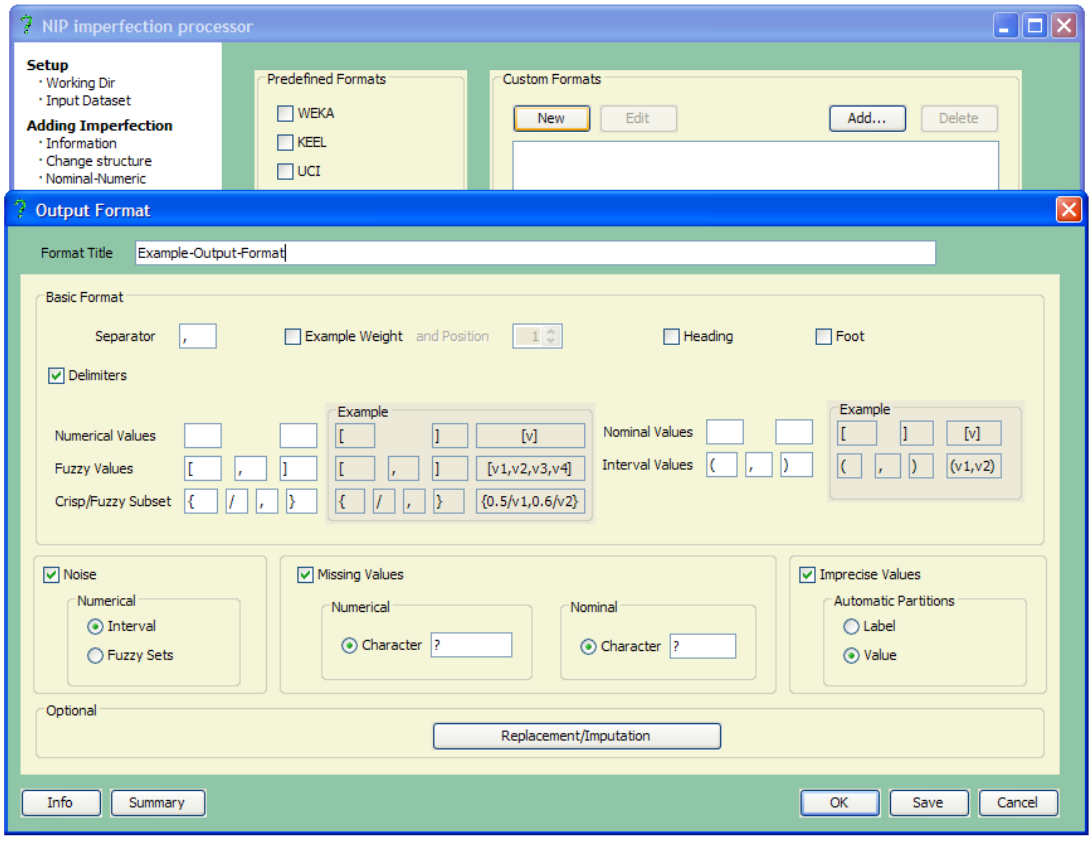


Figura 14.8: Formato de salida

Un aspecto importante y común en la definición de cualquier formato de salida es que, si los valores fuzzy y/o intervalares han sido añadidos desde una partición, éstos pueden representarse en la salida a través de la etiqueta (lingüística) asociada o por el valor que la define.

Cuando el formato de salida seleccionado es un formato estándar, es posible que el conjunto de datos a exportar contenga ciertos tipos de valores de baja calidad con los cuales la técnica del AID que use tal formato no puede trabajar. Estos valores pueden haber sido añadidos por el módulo de manejo de LQD o ya podrían existir en el conjunto de datos original. En este caso el usuario tiene dos alternativas: La primera de ellas es especificar un formado para estos valores de baja calidad, si la técnica del AID ha sido adaptada para manejar estos tipos de datos; y la segunda opción es reemplazar/imputar los valores de baja calidad por otros más apropiados para la técnica. Esta última opción soluciona el posible conflicto entre los valores de baja calidad en los formatos estándares y además nos permite trabajar con un conjunto de datos que originalmente tiene LQD en la técnica del AID.

Para llevar a cabo la acción de reemplazar/imputar LQD, NIPip proporciona una librería interna de técnicas de reemplazamiento/imputación (Figura 14.9).

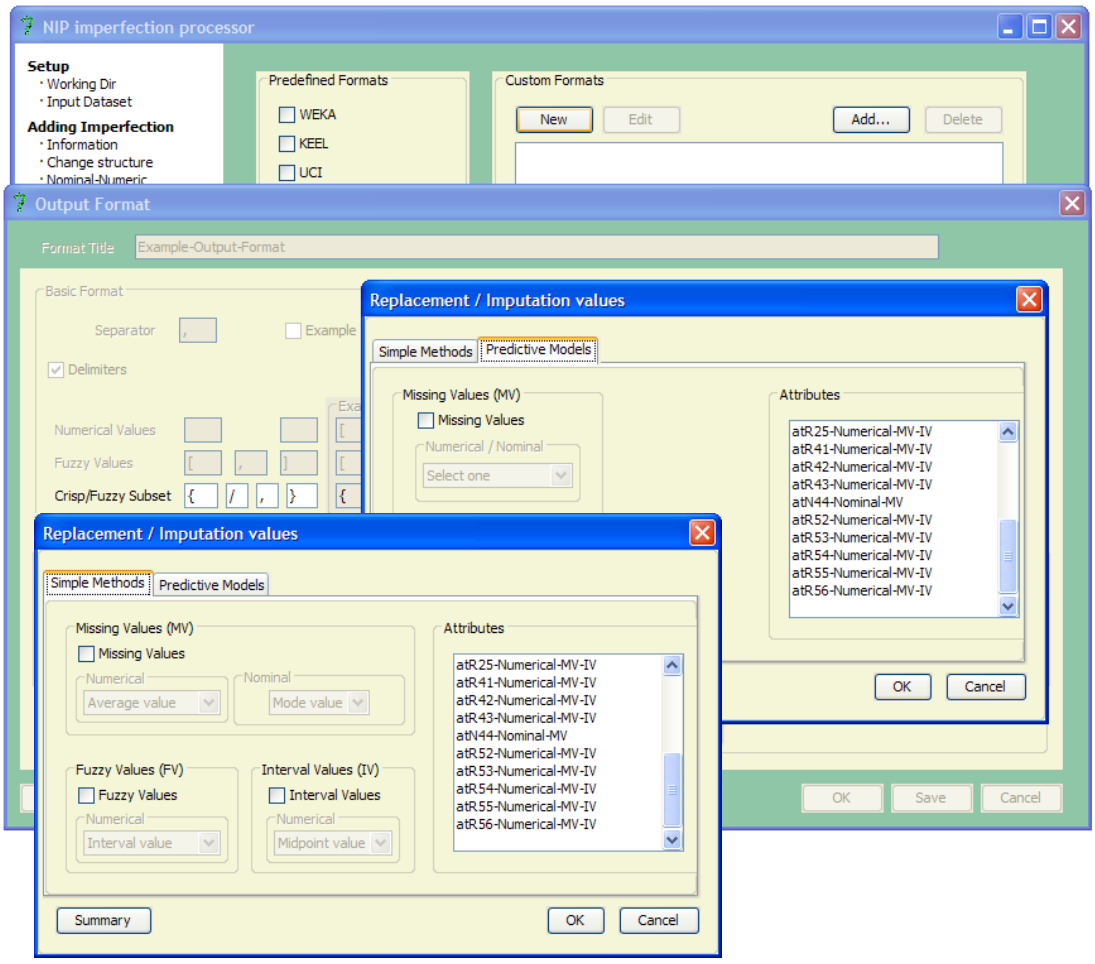


Figura 14.9: Reemplazamiento/Imputación de valores

Esta librería tiene varias técnicas simples frecuentemente utilizadas en la literatura, [13,

137]. Las técnicas para reemplazar/imputar que se proporcionan son los siguientes:

- Reemplazar/imputar valores missing de un atributo numérico por la media del atributo, por la media del atributo tomando su valor de clase como referencia, por un intervalo que incluye todo el dominio del atributo, por un valor fuzzy construido como $(\mu - 2\sigma, \mu - \sigma, \mu + \sigma, \mu + 2\sigma)$ donde μ y σ son la media y la desviación estándar del atributo. Esta última imputación solo puede llevarse a cabo cuando el atributo solamente toma valores crisp en los ejemplos conocidos.
- Reemplazar/imputar un valor missing de un atributo nominal por la moda, por un subconjunto fuzzy formado por los valores del dominio del atributo junto con la proporción de aparición de estos valores en el atributo, o por un subconjunto crisp compuesto por todos los valores del dominio del atributo sin ninguna proporción asociada.
- Reemplazar/imputar valores fuzzy por un valor crisp mediante su centro de gravedad, o por un valor intervalar el cual incluye el soporte para todas las funciones de pertenencias asociadas.
- Reemplazar/imputar un valor intervalar por un valor crisp mediante la utilización del punto medio, o el valor máximo o mínimo del intervalo.

Además de las técnicas simples de reemplazamiento/imputación, se incorporan técnicas predictivas (Figura 14.9). En la actualidad dentro de estas últimas técnicas solo se encuentra disponible el método KNN_{LQD}, descrito en el Capítulo 12. Nuestro objetivo es incorporar nuevas técnicas que permitan lleva a cabo el reemplazamiento/imputación desde LQD. En la Figura 14.9 aparece la pestaña para seleccionar la técnica predictiva de reemplazamiento/imputación KNN_{LQD}, [143].

NIPip proporciona la flexibilidad de seleccionar diferentes opciones de reemplazamiento/imputación para un mismo formato de salida, permitiendo hacer una cadena de reemplazamientos/imputaciones. Esta cadena se lleva a cabo en el siguiente orden de precedencia: primero se reemplazan/imputan los valores missing, posteriormente se reemplazan/imputan los valores fuzzy y finalmente se reemplazan/imputan los valores intervalares. Por ejemplo, si queremos reemplazar/imputar un valor fuzzy por el punto medio del intervalo que define su soporte, debemos de seleccionar la opción de reemplazar/imputar el valor fuzzy por un intervalo y además debemos de seleccionar la opción de reemplazar/imputar el intervalo por su punto medio.

Es importante tener en cuenta que cuando usamos la librería interna de reemplazamiento/imputación, se pueden generar diferentes muestras de valores de baja calidad en diferentes conjuntos de datos.

Finalmente, toda la información disponible en la interfaz se exporta a un fichero. En el directorio correspondiente, se genera un fichero .spec con cada formato de salida seleccionado. Este fichero contiene la información acerca del conjunto de datos exportado indicando para cada atributo su información estadística, el porcentaje de valores de baja calidad que han sido aplicados y si se ha utilizado o no alguna técnica de reemplazamiento/imputación, entre otra información relevante.

14.3.5 Casos de uso

En esta subsección vamos a mostrar dos casos de uso de NIPip. Para llevar a cabo estos casos, hemos seleccionado los conjuntos de datos "Inexpert-57.dat" disponible en la web de la herramienta KEEL[4] y "Hepatitis.arff" disponible en el repositorio de la UCI, [77].

14.3.5.1 Caso 1: Conjunto de datos Inexpert-57.dat

Inexpert-57.dat, [4, 159], es un conjunto de datos con LQD. Esta conjunto de datos contiene datos de niños de una escuela con dislexia. Se ha estado estimando que entre el $4\,\%$ y $5\,\%$ de estos niños sufren este desorden. La dislexia puede ser definida como una discapacidad en personas con un coeficiente intelectual normal y sin problemas físicos o psicológico que expliquen dicha discapacidad.

La presencia de dislexia en niños depende de muchos indicadores, los cuales principalmente intentan detectar si las habilidades de lectura, escritura y cálculo son adquiridas a la velocidad adecuada. Además, hay diferentes trastornos de la dislexia que comparten algunos de sus síntomas y, por lo tanto, las pruebas no solamente deben de detectar los valores anormales de los indicadores mencionados sino que además deben de separar aquellos niños que en realidad sufren de dislexia de aquellos en los cuales el problema puede estar relacionado con otras causas, por ejemplo, falta de atención o hiperactividad.

Una vez definido en NIPip un formato predefinido ("custom") adecuado para la descripción del conjunto de datos, podemos importarlo. Este conjunto de datos está compuesto de 57 atributos y 52 ejemplos. Los atributos son nominales y numéricos y tienen las siguientes características: los atributos 1, 2, 16-25, 41-43 y 52-56 son numéricos y están definidos por intervalos. Todos ellos contiene valores missing excepto el atributo 1. Los otros atributos son nominales y están definidos por subconjuntos crisp/fuzzy. Estos atributos nominales no contienen valores missing. La Figura 14.10 muestra algunos ejemplos del conjunto de datos.

La Figura 14.11 muestra una parte de la pantalla de NIPip con la información sobre la cantidad de LQD que contiene el conjuntos de datos.

cionado,0.4/Regular},{Regular},{Regular},{?},[5.1,7.3],{?},[2,4],[1.3,3.4],{?},[2,3.6],{?
llado},{Basico},{Desproporcionado},{Regular},{Regular},[5.1,7.3],[8,10],[4,6],[4,6],[4,6
proporcionado},{0.4/Regular,0.6/Bajo},{Incompleto},{?},[5.1,7.3],{?},{?},[4,6],{?},[0,2]
proporcionado},{Regular},{Regular},{?},[6,8],[6,8],[2,3.4],[2.9,4.1],{?},[1.3,3.4],{?},[1.3
.8/Regular,0.4/Basico},{Desproporcionado},{0.7/Bajo,0.4/Regular},{Incompleto},[5.2,7.0
,{Basico},{Elevado},{Normal},{Regular},{Regular},[1.3,3.4],[6.0,7.5],[6.9,8.5],[2.9,4.6]
{Bajo},{Desproporcionado},{Regular},{Incompleto},[1,2],{?},[3.4,5.1],[1.3,2.6],[1.3,2.6
Detallado,Aceptable},{Normal},[4,6],[8,10],[4,6],[4.4,6.3],[4.4,6.3],{?},[7.3,9],{?},{?},{
o,0.5/Regular},{Normal},{Aceptable},{Normal},[4,6],{?},[5.8,7.9],[5.5,7.6],[5.1,6.9],{?}
able},{Normal},[3.4,5.1],[3.8,6],[6.3,7.9],[3.8,5.1],[3,4.7],[8,10],[8,9.7],{?},[1.3,3.4],{?},{
sproporcionado},{Regular},{0.82/Regular,0.3/Bajo},[0.2,2],{?},[4.6,6.9],[2.2,4],[1.2,2.5],
{0.4/Regular,0.7/Bajo},{Bajo},[1.3,3.4],{?},{?},[0,0],{?},{?},{?},[3.4,5.1],[0,2],{?},{Imp
Regular,0.4/Elevado},{Desproporcionado},{0.5/Regular,0.9/Aceptable},{Regular},[2,4],[1
},{0.7/Completo,0.3/Normal},[3.4,5.1],[6,8],[7.3,9],[6,8],[3.4,5.1],{?},[7.3,9],{?},[6,8],{?
proporcionado},{Aceptable},{Normal},[4,6],[4,6],[6,8],[2,3.8],[4,5.3],{?},[5.1,7.3],{?},[4
evado,0.4/Basico},{Normal},{Aceptable},{Normal},[2,4],[4,6],[7.3,9],[3.1,4.9],[4,6],{?}
roporcionado},{0.5/Regular,0.5/Elevado},{Regular},[4,6],{?},[6,8],[4,5.8],[3.1,5.0],{?},[1
egular},[4,6],{?},[6,8],[4,6],[3.4,5.1],{?},[4.4,6.1],{?},[2.2,4.3],[4.8,6.5],{Regular},{Si},{N
ular,Bajo},{Basico,Elevado,Detallado,Incoherente,Regular,Bajo},{Incoherente,Desproporc
Normal,Regular},{Regular},{Regular,Bajo},[5.1,6.9],{?},[6,8],[4,6],[2,4],{?},[1.2,2.4],{?

Figura 14.10: Una parte del conjunto de datos "Inexpert-57.dat"

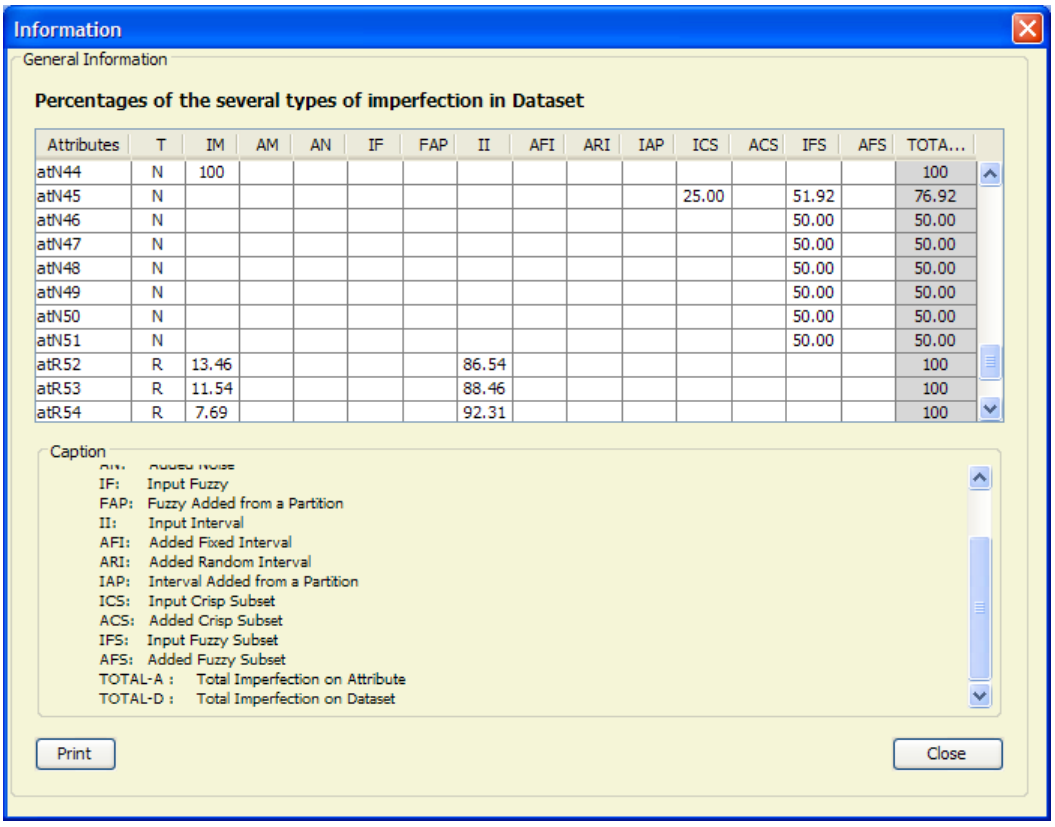


Figura 14.11: Resumen de LQD en Inexpert-57.dat

Como podemos observar en esta figura, el atributo 44 contiene un 100 % de valores missing. En esta situación hemos decidido borrar este atributo utilizando la opción de preproceso previa para borrar atributos. La Figura 14.12 nos muestra esta opción.

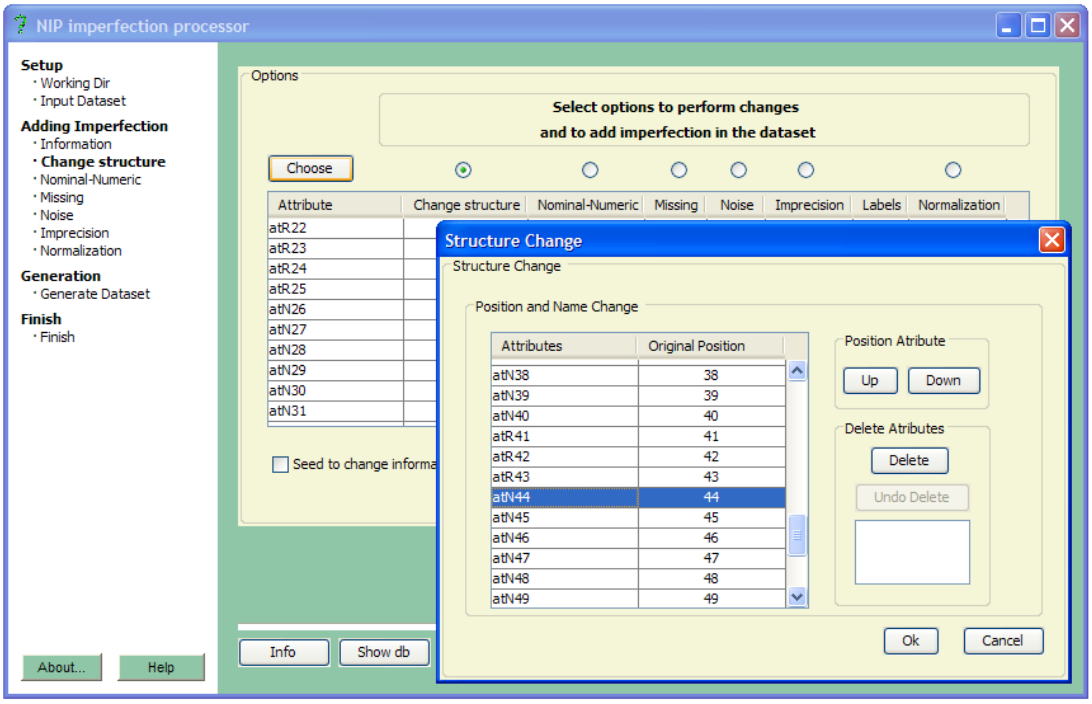


Figura 14.12: Borrando el atributo 44 de Inexpert-57.dat

Con la ayuda de NIPip queremos eliminar todos los valores missing del conjunto de datos llevando a cabo reemplazamiento/imputación. Para ello, en el módulo de exportación definimos un formato de salida e indicamos que se lleve a cabo un "Reemplazamiento/Imputación". Utilizando esta funcionalidad podemos reemplazar valores missing por valores intervalares. En este caso podemos considerar dos opciones:

- Reemplazarlos por la media del atributo expresada como intervalo, o
- reemplazarlos por un intervalo que represente todo el dominio del atributo al cual pertenezcan los valores missing. En este caso, un valor missing se interpreta como la ausencia total de información.

Por ejemplo, el dominio del atributo 2 es el intervalo [0,10] y la media es [5.4229,7.3549]. Por lo tanto, en la primera opción un valor missing sería reemplazado por la media, es decir, el intervalo [5.4229,7.3549], mientras que con la segunda opción se reemplazaría por [0,10].

En este caso, hemos elegido la opción 2 y en la Figura 14.13 se muestra parte del conjunto de datos transformado. En esta figura, destacamos algunos de los valores transformados con respecto al conjunto de datos original mostrado en la Figura 14.10.

rcionado,0.4/Regular},{Regular},{Regular},[0.2,10.0],[5.1,7.3],[0.0,10.0],[2,4],[1.3,3.4],[
llado},{Basico},{Desproporcionado},{Regular},{Regular},[5.1,7.3],[8,10],[4,6],[4,6],[4,6
proporcionado},{0.4/Regular,0.6/Bajo},{Incompleto},[0.2,10.0],[5.1,7.3],[0.0,10.0],[0.0,8
proporcionado},{Regular},{Regular},[0.2,10.0],[6,8],[6,8],[2,3.4],[2.9,4.1],[8.0,10.0],[1.3
.8/Regular,0.4/Basico},{Desproporcionado},{0.7/Bajo,0.4/Regular},{Incompleto},[5.2,7.
},{Basico},{Elevado},{Normal},{Regular},{Regular},[1.3,3.4],[6.0,7.5],[6.9,8.5],[2.9,4.6
,{Bajo},{Desproporcionado},{Regular},{Incompleto},[1,2],[1.3,10.0],[3.4,5.1],[1.3,2.6],[
{Detallado,Aceptable},{Normal},[4,6],[8,10],[4,6],[4.4,6.3],[4.4,6.3],[8.0,10.0],[7.3,9],[3.4
o,0.5/Regular},{Normal},{Aceptable},{Normal},[4,6],[1.3,10.0],[5.8,7.9],[5.5,7.6],[5.1,6.
able},{Normal},[3.4,5.1],[3.8,6],[6.3,7.9],[3.8,5.1],[3,4.7],[8,10],[8,9.7],[3.4,5.1],[1.3,3.4]
sproporcionado},{Regular},{0.82/Regular,0.3/Bajo},[0.2,2],[1.3,10.0],[4.6,6.9],[2.2,4],[1
{0.4/Regular,0.7/Bajo},{Bajo},[1.3,3.4],[1.3,10.0],[0.0,10.0],[0,0],[1.2,9.0],[8.0,10.0],[0.0
Regular,0.4/Elevado},{Desproporcionado},{0.5/Regular,0.9/Aceptable},{Regular},[2,4],[
},{0.7/Completo,0.3/Normal},[3.4,5.1],[6,8],[7.3,9],[6,8],[3.4,5.1],[8.0,10.0],[7.3,9],[3.4,5
proporcionado},{Aceptable},{Normal},[4,6],[4,6],[6,8],[2,3.8],[4,5.3],[8.0,10.0],[5.1,7.3],
levado,0.4/Basico},{Normal},{Aceptable},{Normal},[2,4],[4,6],[7.3,9],[3.1,4.9],[4,6],[8.0
roporcionado},{0.5/Regular,0.5/Elevado},{Regular},[4,6],[1.3,10.0],[6,8],[4,5.8],[3.1,5.0]
egular},[4,6],[1.3,10.0],[6,8],[4,6],[3.4,5.1],[8.0,10.0],[4.4,6.1],[3.4,5.1],[2.2,4.3],[4.8,6.5]
ular,Bajo},{Basico,Elevado,Detallado,Incoherente,Regular,Bajo},{Incoherente,Despropor
Normal,Regular},{Regular},{Regular,Bajo},[5.1,6.9],[1.3,10.0],[6,8],[4,6],[2,4],[8.0,10.0

Figura 14.13: Una parte del conjunto de datos “Inexpert-57.dat” transformada

Para este caso, la salida de NIPip es la siguiente: 1) “Inexpert-57.custom” que contiene los datos de salida, y 2) el fichero de especificación “Inexpert-57.custom.spec” con información de las transformaciones llevadas a cabo en el conjunto de datos.

14.3.5.2 **Caso 2: Conjunto de datos Hepatitis.arff**

Hepatitis.arff es un conjunto de datos que contiene información médica acerca de la esperanza de vida de los pacientes con hepatitis. Para importar este conjunto de datos debemos de elegir en NIPip el formato estándar ARFF. El conjunto de datos se compone de 20 atributos incluyendo el atributo de clase (el último) y 155 ejemplos. Los atributos son nominales, enteros y los atributos 14 y 17 son reales. Todos los atributos tienes valores missing excepto los atributos 1, 2, 4, 19 y el atributo clase. En la Figura 14.14 se pueden observar algunos ejemplos de este conjunto de datos.

Supongamos que en este caso queremos añadir etiquetas (lingüísticas) desde una partición automática a los dos atributos reales del conjunto de datos. Para llevar a cabo esto, utilizamos una partición generada por la primera técnica de discretización proporcionada por la librería interna de NIPip (Figura 14.7). Una vez generada la partición, ésta se guarda en un fichero donde para cada atributo se indica en la primera línea el nombre del atributo seguido del número de conjuntos fuzzy generados para ese atributo. Después, las siguientes líneas del fichero muestran los diferentes conjuntos fuzzy generados. En la Figura 14.15 se muestra la partición fuzzy de los atributos.

```
30,male,no,no,no,no,no,no,no,no,no,no,no,1,85,18,4,?,no,LIVE
50,female,no,no,yes,no,no,no,no,no,no,no,no,0.9,135,42,3.5,?,no,LIVE
78,female,yes,no,yes,no,no,yes,no,no,no,no,no,0.7,96,32,4,?,no,LIVE
31,female,?,yes,no,no,no,yes,no,no,no,no,no,0.7,46,52,4,80,no,LIVE
34,female,yes,no,no,no,no,yes,no,no,no,no,no,1,?,200,4,?,no,LIVE
34,female,yes,no,no,no,no,yes,no,no,no,no,no,0.9,95,28,4,75,no,LIVE
51,female,no,no,yes,no,yes,yes,no,yes,yes,no,no,?,?,?,?,?,no,DIE
23,female,yes,no,no,no,no,yes,no,no,no,no,no,1,?,?,?,?,no,LIVE
39,female,yes,no,yes,no,no,yes,yes,no,no,no,no,0.7,?,48,4.4,?,no,LIVE
30,female,yes,no,no,no,no,yes,no,no,no,no,no,1,?,120,3.9,?,no,LIVE
39,female,no,yes,no,no,no,no,yes,no,no,no,no,1.3,78,30,4.4,85,no,LIVE
32,female,yes,yes,yes,no,no,yes,yes,no,yes,no,no,1,59,249,3.7,54,no,LIVE
41,female,yes,yes,yes,no,no,yes,yes,no,no,no,no,0.9,81,60,3.9,52,no,LIVE
30,female,yes,no,yes,no,no,yes,yes,no,no,no,no,2.2,57,144,4.9,78,no,LIVE
47,female,no,yes,no,no,no,yes,no,no,no,no,no,?,?,60,?,?,no,LIVE
38,female,no,no,yes,yes,yes,yes,no,no,no,yes,no,2,72,89,2.9,46,no,LIVE
66,female,yes,no,yes,no,no,yes,no,no,no,no,no,1.2,102,53,4.3,?,no,LIVE
40,female,no,no,yes,no,no,yes,yes,no,no,no,no,0.6,62,166,4,63,no,LIVE
38,female,yes,no,no,no,no,yes,no,no,no,no,no,0.7,53,42,4.1,85,yes,LIVE
```

Figura 14.14: Una parte del conjunto de datos “Hepatitis.arff”

```
BILIRUBIN 3
BILIR_low,0.3,0.3,1.29099,2.00863
BILIR_medium,1.29099,2.00863,2.15801,2.74167
BILIR_high,2.15801,2.74167,8.0,8.0
ALBUMIN 4
ALBU_low,2.1,2.1,3.26057,3.6394
ALBU_mediumlow,3.26057,3.6394,3.8397799,3.8599901
ALBU_mediumhigh,3.8397799,3.8599901,3.8849301,3.91503
ALBU_high,3.8849301,3.91503,6.4,6.4
```

Figura 14.15: Partición fuzzy de los atributos 14 y 17 de Hepatitis.arff

Utilizando esta partición, obtenemos un nuevo conjunto de datos con todos los valores de los atributos 14 y 17 como valores fuzzy, expresados mediante su etiqueta (lingüística). Tal y como hemos explicado en el subsección 14.3.3, cada valor se reemplaza por el valor fuzzy de la partición a la que pertenezca con mayor grado. También podríamos haber seleccionado reemplazar solamente un porcentaje de los valores en lugar de todos.

Finalmente, obtenemos el conjunto de datos transformado en el formato de salida seleccionado. La Figura 14.16 muestra una parte del conjunto de datos transformado en el formato ARFF. En esta figura destacamos algunos de los valores transformados con respecto al conjunto de datos original de la Figura 14.14

En este caso, la salida de NIPip son tres ficheros: 1) el fichero “hepatitis-m.arff” que contiene el conjunto de datos transformado, 2) el fichero “hepatitis-m.arff.spec” y 3) el fichero con las particiones fuzzy de los atributos numéricos.

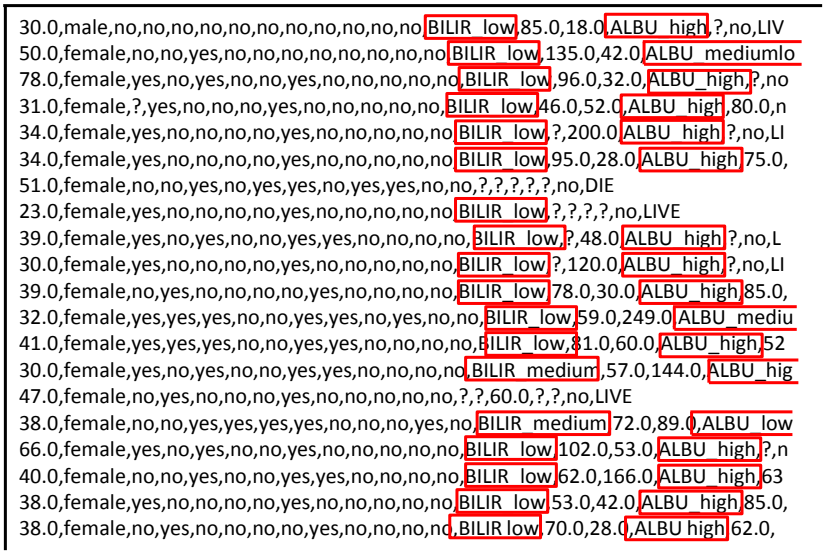


Figura 14.16: Una parte del conjunto de datos “hepatitis.arff” transformado

14.4 El entorno experimenter de NIPip

ExpNIPip, [142], está compuesto de dos bloques generales: un framework para la construcción de un experimento o grupos de experimentos y una simple e intuitiva interfaz gráfica desarrollada con la tecnología Java y la API Swing, para controlar la construcción de tales experimentos. Es importante aclarar antes de seguir con la explicación que cuando hablamos de experimentos en el entorno de ExpNIPip nos referimos a la preparación de todos los conjuntos de datos (con o sin LQD) que posteriormente utilizará el investigador para llevar a cabo sus tests en el campo del AID.

ExpNIPip permite diseñar gráficamente un experimento o grupo de experimentos con conjuntos de datos. La interfaz del entorno de ExpNIPip está basada en el flujo de datos donde se pueden seleccionar los distintos elementos funcionales (que llamaremos componentes) desde una barra de herramientas y puede conectarlos mediante links con el fin de reflejar la secuencia que siguen dichos elementos. Esta estructura en forma de grafo representa el flujo de conocimiento en el preproceso de datos (hay que tener en cuenta que el grafo construido debe de ser acíclico).

Este entorno facilita el uso de este paquete a nivel de investigación para llevar a cabo la preparación de experimentos a gran escala. Sus principales ventajas y diferencias respecto a NIPip son las siguientes:

- Permite llevar a cabo las principales funcionalidades del paquete NIPip para un grupo grande de conjuntos de datos en un solo paso.

- Incluye nuevas funcionalidades que no contiene el paquete NIPip, tales como nuevas técnicas de imputación y/o discretización.
- Los experimentos se definen gráficamente basándose en el flujo de datos.
- Permite definir varias secuencias funcionales sobre los conjuntos de datos (diferentes ramas de un mismo grafo), siendo capaz de llevar a cabo experimentos similares sobre muchos conjuntos de datos de forma fácil y rápida.
- Una vez diseñado un experimento, éste puede ser ejecutado en modo online. ExpNIPip permite la ejecución en paralelo en modo online de varios experimentos, así el usuario puede continuar definiendo otros experimentos. Un log indica el estado de las ejecuciones de los experimentos en progreso (este log puede ser visualizado por el usuario en todo momento).
- Un experimento puede ser guardado y ejecutado en cualquier momento mediante la consola de comandos (modo batch) usando el script generado al configurar y guardar el experimento.
- Un experimento puede recuperarse desde el entorno ExpNIPip para ser modificado. Esto permite obtener diferentes versiones del mismo experimento de una forma rápida y sencilla.

14.4.1 **Principales acciones y componentes del entorno ExpNIPip**

la pantalla principal cuando accedemos al entorno ExpNIPip se muestra en la Figura 14.17.

En la parte superior de la pantalla se encuentran una serie de opciones que permiten incorporar componentes a un experimento. Estas opciones despliegan una barra de herramientas que permite configurar el experimento de una forma gráfica e interactiva. El panel central de la pantalla es el panel de dibujo donde se muestran los componentes del experimento actual y el flujo y las relaciones entre ellos. Las principales acciones para llevar a cabo la creación de un experimento están agrupadas en los menús “File” y “Actions”:

- *New*: Crear un nuevo experimento.
- *Open*: Seleccionar un experimento previamente creado y cargarlo para editarlo.
- *Save*: Guardar el experimento actual con el nombre actual o con otro nombre (*Save as*). Al guardar el experimento, se generan los ficheros “.exp” y “.scr”. Así el experimento puede ser ejecutado en modo batch en cualquier momento usando el script del fichero “.scr” o puede ser recuperado en cualquier otro momento para completarlo o modificar su estructura.

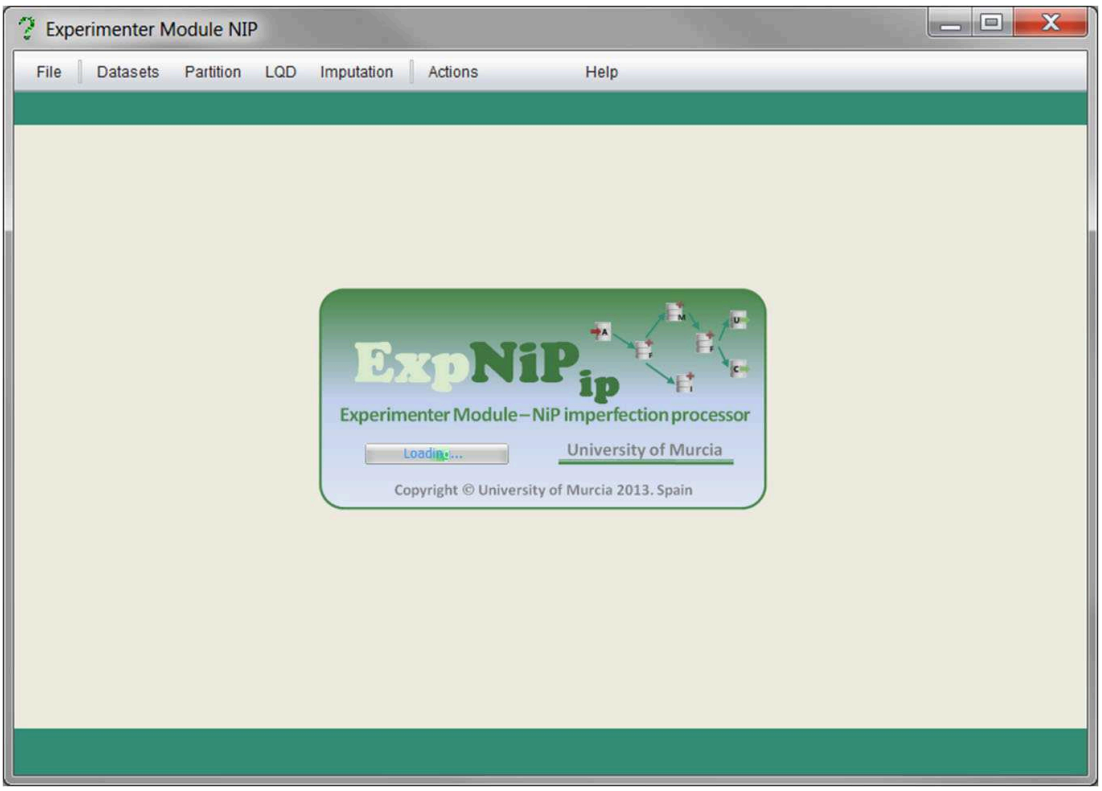


Figura 14.17: Interfaz del entorno ExpNIPip

- *Run*: Ejecutar el experimento actual en modo online y generar los conjuntos de datos correspondientes. ExpNIPip permite ejecutar en modo online más de un experimento mientras el usuario continúa con la definición de otros experimentos.
- *Links*: Al añadir links entre los nodos del grafo, definimos las relaciones entre ellos.
- *Visualize*: Muestra el estado de la ejecución de los experimentos que están siendo ejecutados en modo online. El resultado se guarda en un fichero que toma el nombre del experimento y con extensión ".log".

Los posibles componentes (en un segundo nivel) del grafo que configura un experimento son:

- *Datasets->Import*: Permite definir el conjunto de conjuntos de datos de entrada de un experimento.
- *Partition->Crisp*: Permite aplicar diferentes técnicas de discretización crisp sobre los atributos numéricos de los conjuntos de datos.
- *Partition->Fuzzy*: Permite aplicar diferentes técnicas de discretización fuzzy sobre los atributos numéricos de los conjuntos de datos.
- *LQD->Add*: Permite añadir diferentes tipos de LQD a los conjuntos de datos.

- *Imputation−>Missing*: Permite aplicar diferentes técnicas de imputación sobre valores missing.
- *Imputation−>Interval*: Permite aplicar diferentes técnicas de imputación sobre valores intervalares.
- *Imputation−>Fuzzy*: Permite aplicar diferentes técnicas de imputación sobre valores fuzzy.
- *LQD−>formats*: Permite definir los formatos de salida de los LQD.
- *Datasets−>Export*: Especifica los formatos de salida para los conjuntos de datos de un experimento.

La Figura 14.18 muestra un ejemplo de la dinámica de trabajo con el entorno ExpNIPip y sus principales elementos. Cuando introducimos un componente de cualquier tipo, activamos el botón derecho del ratón sobre él y seleccionando la opción "Properties" podemos acceder a sus propiedades particulares.

La verificación semántica en la construcción de un grafo se lleva a cabo mediante el módulo de control semántico. Cuando hemos completado la definición de un experimento y en cualquier momento de la ejecución, el módulo de control semántico examina que no existan inconsistencias en la definición del experimento, mostrando un mensaje de error en el caso de encontrarlo. Como regla general, un grafo de un experimento debe empezar en el nodo raíz el cual debe de ser un componente del tipo *Datasets−>Import* y debe terminar en un nodo hoja que debe ser del tipo *Partition* o *Datasets−>Export*, correspondientes a la generación de particiones o la definición de un formato de salida, respectivamente.

La Figura 14.19 muestra el diagrama de clases simplificado sobre el que se construye ExpNIPip. El diagrama muestra solamente el conjunto de clases principales omitiendo los atributos y métodos, de forma que se pueda identificar fácilmente la estructura de ramas del grafo de un experimento.

14.4.2 **Importación de conjuntos de datos**

El componente *Datasets−>Import* permite incorporar conjuntos de datos en varios formatos estándares y predefinidos por el usuario ("custom"). Los formatos de entrada disponibles son ARFF, KEEL, UCI, CSV y para introducir un conjunto de datos con formato "custom" debemos de crear previamente un fichero ".in" mediante NIPip el cual define el formato particular del conjunto de datos.

ExpNIPip permite incorporar tantos componentes *Datasets−>Import* como el usuario necesite para un experimento. Cuando se añade este tipo de componente a un experimento hay

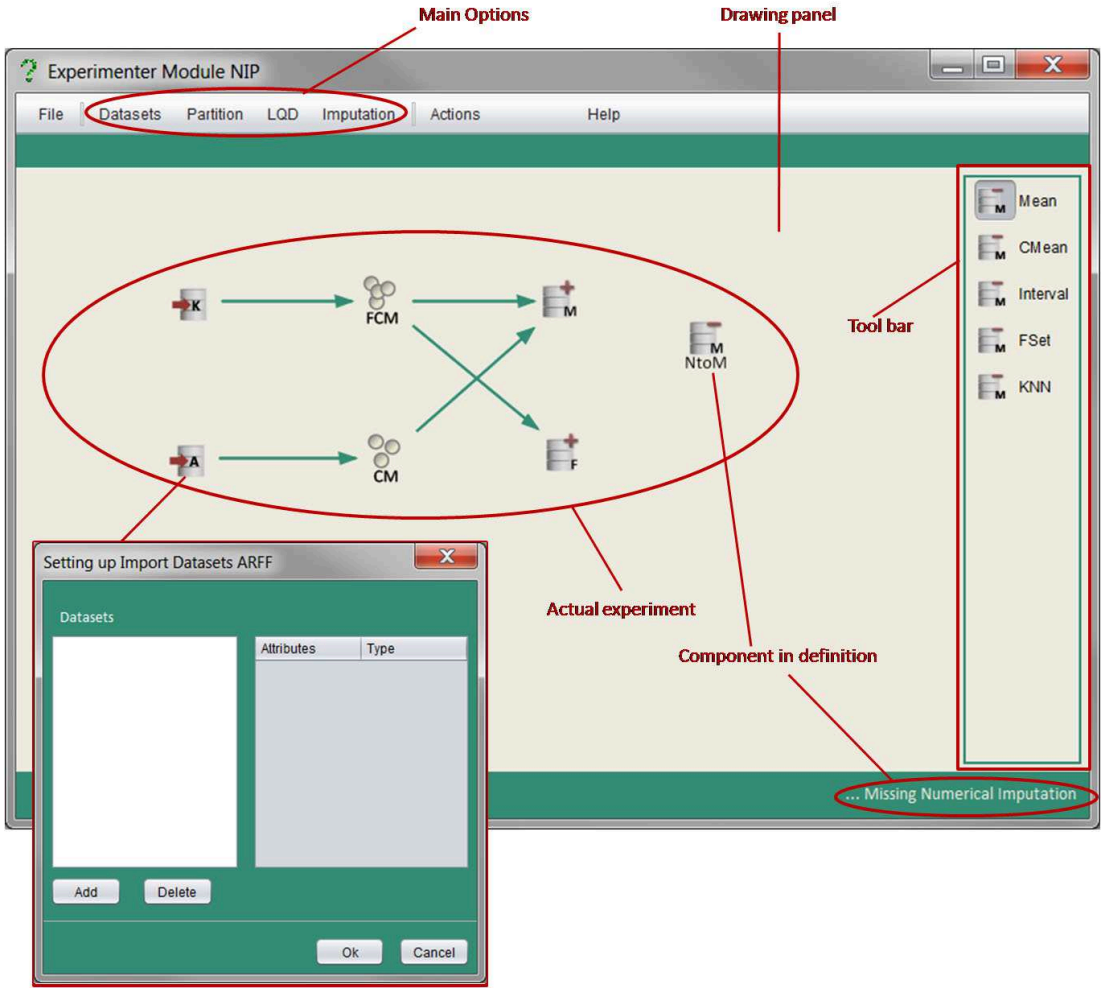


Figura 14.18: Diferentes elementos de ExpNIPip

que configurarlo para incorporar tantos conjuntos de datos como se desee en el formato especificado por el componente. Para incorporar a un experimento conjuntos de datos con diferentes formatos predefinidos por el usuario, solamente hay que añadir tantos componentes *Datasets−>Import* como formatos "Custom" diferentes existan y añadirle el formato "Custom" definido con NIPip.

En cualquier momento durante la definición de un experimento, el usuario puede modificar la configuración de cualquier componente. Tales cambios se transfieren a los otros componentes que estén conectados con él de forma directa o indirecta. El módulo de control semántico es el responsable de chequear que los componentes *Datasets−>Import* sean siempre los nodos raíz de las diferentes ramas del grafo y que al menos haya un componente de este tipo al inicio de cada rama.

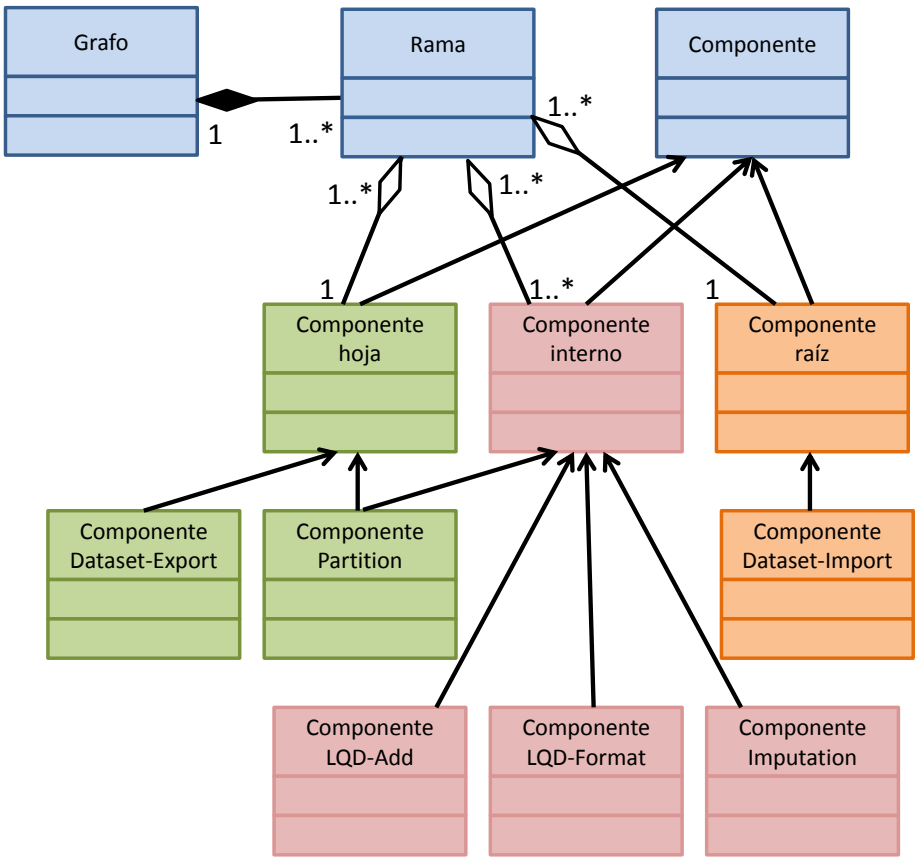


Figura 14.19: Diagrama de clases simplificado de ExpNIPip

14.4.3 Técnicas de discretización

El componente *Partition* permite la incorporación a un experimento de la discretización de los atributos numéricos de un grupo de conjuntos de datos. Esta discretización o particionamiento puede llevarse a cabo a través de un conjunto de técnicas con el fin de proporcionarle al investigador la generación automática de tales particiones para que sean utilizadas en técnicas del AID que requieran de dicha discretización o bien para reemplazar el valor de los atributos numéricos por sus correspondientes etiquetas (lingüísticas). ExpNIPip facilita, por lo tanto, la tarea de testear qué tipo de discretización es más adecuada para una técnica particular.

Las Tablas 14.2 y 14.3 muestran las diferentes técnicas de discretización disponibles y se clasifican de acuerdo al tipo de partición generada. Para cada uno de ellos se muestra el acrónimo usado en el paquete, una referencia y una pequeña descripción.

Además de las técnicas de discretización mencionadas, disponemos de la posibilidad de usar una partición generada por una técnica de discretización externa usando el componente *Partition−>Crisp−>Extern* para las particiones externas crisp y el componente *Partition−>Fuzzy−>Extern* para las particiones externas fuzzy. Es importante tener en cuenta

Tabla 14.2: Técnicas de partición Crisp

Acronim	Ref.	Descripción
DT	[43]	Basado en un árbol de decición y un algoritmo genético
CM	[76]	Basado en el algoritmo C-Means
Extern	–	Cualquier partición crisp externa en el formato apropiado

Tabla 14.3: Técnicas de partición Fuzzy

Acronim	Ref.	Descripción
OFP	[43]	Basado en un árbol de decición y un algoritmo genético
BAGOFP	[38]	Basado en un árbol de decisión, un algoritmo genético y bagging
GA	[58]	Basado en un algoritmo genético
FCM	[18]	Fuzzy C-Means
Extern	–	Cualquier partición fuzzy externa en el formato apropiado

que las particiones deben de proporcionarse en el formato adecuado y que la discretización fuzzy permitida es aquella que está compuesta por conjuntos fuzzy trapezoidales. El formato del fichero que contiene las particiones debe de ser similar al mostrado en la Figura 14.20. Este fichero incluye para cada atributo numérico, diferentes etiquetas (lingüísticas) que se corresponden con la discretización, indicando para cada uno los cuatro valores que definen la función de pertenencia trapezoidal. En el caso de la partición crisp, el formato del fichero es el mismo, solo que en este caso los dos primeros valores de la etiqueta (lingüística) son iguales y los dos últimos también.

```
AGE 6
  AGE1,7.0,7.0,24.8849,28.1154
  AGE2,24.8849,28.1154,37.4022,37.593903
  AGE3,37.4022,37.593903,40.3984,40.6043
  AGE4,40.3984,40.6043,41.9675,44.0265
  AGE5,41.9675,44.0265,52.9370,70.062195
  AGE6,52.937,70.062195,78.0,78.0
BILIRUBIN 1
  BILIRUBIN0,0.3,0.3,8.0,8.0
ALK_PHOSPHATE 2
  ALK_PHOSPHATE1,26.0,26.0,37.3249,201.657
  ALK_PHOSPHATE2,37.3249,201.657,295.0,295.0
SGOT 1
  SGOT0,14.0,14.0,648.0,648.0
```

Figura 14.20: Fichero con una partición fuzzy

El módulo de control semántico de ExpNIPip es el responsable de detectar que al menos se especifique una partición cuando sea necesario para llevar a cabo la ejecución del experimento. Esto ocurre cuando la opción de añadir imprecisión en atributos numéricos se selecciona (fuzzy, intervalar o subconjuntos fuzzy) o si se quiere expresar los valores fuzzy e intervalares usando su correspondiente etiqueta (lingüística) en el formato de salida de los conjuntos de datos.

Cuando hay varios componentes *Partition* en una secuencia de flujo, es el último componente el que prevalece sobre los otros.

14.4.4 **Añadiendo LQD**

El componente *LQD->Add* de ExpNIPip encapsula la funcionalidad de introducir ciertos porcentajes de LQD en los conjuntos de datos. Esta funcionalidad permite generar el mismo conjunto de datos con diferentes porcentajes de LQD, lo cual facilita la tarea de generar conjuntos de datos para evaluar la robustez de las técnicas que pueden manejar LQD. Las posibilidades que se proporcionan son las siguientes:

- Añadir ciertos porcentajes de valores missing tanto en atributos nominales como numéricos. La introducción de estos valores se lleva a cabo aleatoriamente, [86]. En este caso la distribución de los ejemplos con valores missing en un atributo no depende de los datos observados y, por lo tanto, la distribución de los datos completos y de los datos con missing es la misma.
- Añadir cierto porcentaje de valores fuzzy en atributos numéricos. En este caso es necesario tener discretizados los atributos numéricos, bien utilizando un componente de partición automática o bien indicando una partición en el formato correcto con el componente de partición externa. Dada una partición, cuando un valor crisp debe de ser transformado a un valor fuzzy, el valor fuzzy de la partición será aquel al que el valor numérico pertenezca con un mayor grado.
- Añadir un cierto porcentaje de valores intervalares en atributos numéricos. En este caso se proporcionan tres opciones: La primera opción es utilizar una partición crisp generada automáticamente o proporcionada por el usuario; La segunda es sumar y restar una cantidad fija especificada por el usuario al valor numérico del atributo; y la tercera es sumar y restar una cantidad aleatoria dentro de un rango especificado por el usuario al valor numérico del atributo.
- Añadir subconjuntos crisp en atributos nominales. Si un atributo nominal se reemplaza por un subconjunto crisp, todos los demás valores posibles del dominio pueden ser añadidos al actual valor con una probabilidad igual a la proporción de cada valor en el conjunto de datos.
- Añadir subconjuntos fuzzy tanto en atributos numéricos como en atributos nominales. En el caso de los atributos numéricos, dada una partición fuzzy de un atributo, el valor numérico será reemplazado por el subconjunto de etiquetas de la partición a los cuales este valor pertenece con un grado de pertenencia mayor de cero. Para el caso de los atributos nominales, si el valor se reemplaza por un subconjunto fuzzy, como grado de

pertenencia asociado al valor actual se añade la proporción en el conjunto de datos del mismo, y los otros valores que pertenecen al dominio se añaden de la misma forma que un subconjunto crisp, pero incluyendo como grado de pertenencia la proporción de cada valor en el conjunto de datos.

- Añadir cierto porcentaje de ruido en atributos numéricos y nominales. En el caso de los atributos numéricos, el ruido que se añade es ruido gaussiano. En este tipo de ruido, es necesario indicar en el atributo el cual se quiere modificar, la media y la desviación estándar. Este tipo de ruido se utiliza para simular la existencia de un error en los datos con distribución conocida, por ejemplo, un sensor del cual se conoce su medida de error. Para el caso de los atributos nominales, se cambiará aleatoriamente un valor por otro valor del dominio.

Cuando se definen las propiedades de los componentes *LQD*−>*Add*, se muestra en la ventana de configuración una lista de todos los conjuntos de datos asociados mediante links a este componente. Si seleccionamos uno de estos conjunto de datos, se expande a la derecha la lista de atributos que lo componen. En esta lista es posible seleccionar todos los atributos, sólo los nominales o sólo los numéricos Esta opción de selección depende del tipo de LQD que queramos añadir. La Figura 14.21 muestra un ejemplo.

Además, el módulo de control semántico detecta que en el flujo de datos de un experimento exista una partición cuando se vayan a añadir valores fuzzy, intervalares o subconjuntos fuzzy en atributos numéricos.

14.4.5 **Técnicas de Imputación**

En la literatura, la mayoría de técnicas de imputación existentes consisten en imputar los valores missing por un valor crisp. Sin embargo, ExpNIPip permite la posibilidad de llevar a cabo la imputación no solo de valores missing, sino también de otros tipos de valores de baja calidad, por ejemplo, valores fuzzy e intervalares. Además, dependiendo de la técnica de imputación seleccionada el valor imputado puede ser un valor crisp u otro valor de baja calidad, por ejemplo, un valor fuzzy puede ser imputado por un intervalo. Por lo tanto, cuando se vaya a seleccionar la técnica de imputación hay de tener en cuenta el tipo de LQD con el cual la técnica del AID pueda trabajar.

Una importante ventaja de las técnicas de imputación es que son independientes de la técnica del AID que se aplique posteriormente, pero algunos de ellos son más apropiados que otros cuando trabajan con una técnica en particular. Por eso, ExpNIPip proporciona la funcionalidad de generar de una forma fácil y rápida un mismo conjunto de datos con los valores imputados con diferentes técnicas de imputación, [142].

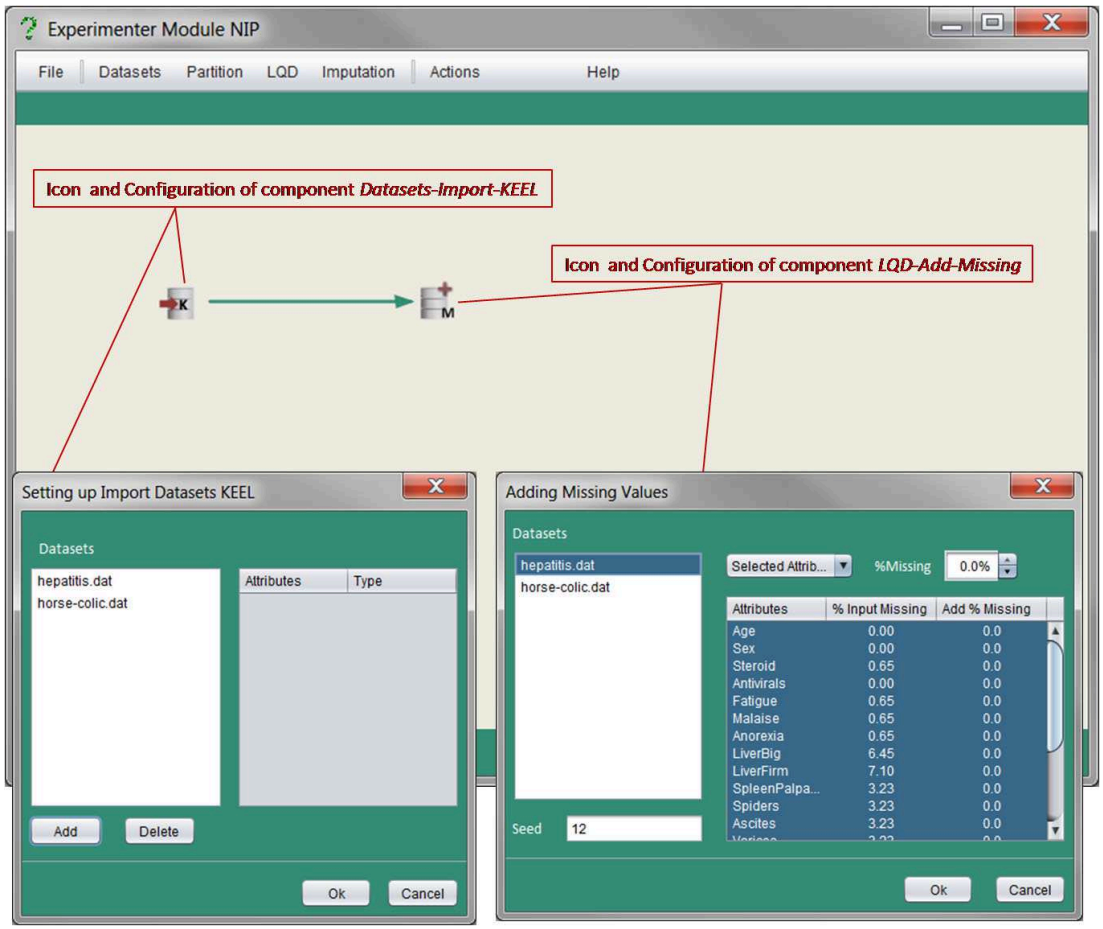


Figura 14.21: Configuración de los componentes *LQD-Add*

ExpNIPip incluye las técnicas de reemplazamiento de NIPip y también proporciona nueva funcionalidad incorporando técnicas que usan un modelo predictivo para imputar los valores. Las técnicas de imputación disponibles y el tipo de valor que proporciona tras la imputación se describen a continuación (entre paréntesis se muestra el acrónimo usado en la herramienta para cada técnica):

- Para valores missing en atributos nominales:

NotoM El valor missing se imputa por la moda o valor más común del atributo. Se proporciona un valor crisp.

NotoCM El valor missing se imputa por la moda conceptual, es decir, el valor más común del atributo considerando solamente los ejemplos que pertenecen a la misma clase del ejemplo a imputar. El valor proporcionado es crisp.

NotoCS El valor missing se imputa por todo el conjunto de valores posibles. El valor generado es un conjunto que contiene todos los posibles valores del atributo. El conjunto

es crisp porque los valores no tienen asignado un peso o valor de probabilidad.

NotoFS El valor missing se imputa por un subconjunto fuzzy que asigna a cada posible valor del atributo la proporción con la cual aparece en el conjunto de datos. En este caso el valor proporcionado es un conjunto que se compone de posibles valores del atributo con un peso asociado a cada valor en función de la proporción de ocurrencias de ese valor en el conjunto de datos.

NoKNN El valor missing se imputa por un valor crisp o por un subconjunto crisp/fuzzy dependiendo del valor obtenido por la técnica de imputación KNN_{LQD}, [143].

- Para valores missing en atributos numéricos:

NtoM El valor missing se imputa por la media del atributo. El valor devuelto es crisp.

NtoCM El valor missing se imputa por la media conceptual, es decir, por la media del atributo considerando solamente los ejemplos que pertenezcan a la misma clase que el ejemplo a imputar. El valor proporcionado es crisp.

NtoI El valor missing se imputa por un intervalo que recoge todos los valores del dominio. El valor generado es un intervalo crisp definido como el valor mínimo y máximo del atributo en el conjunto de datos.

NtoF El valor missing se imputa por un conjunto fuzzy $(\mu - 2\sigma, \mu - \sigma, \mu + \sigma, \mu + 2\sigma)$ donde μ donde σ son la media y la desviación estandar del atributo. En este caso el valor generado es un conjunto fuzzy trapezoidal.

NKNN El valor missing se imputa por un valor crisp, fuzzy o intervalar dependiendo del valor obtenido por la técnica de imputación KNN_{LQD} [143].

- Para valores intervalares:

ItoC Se imputa el intervalo por el punto medio. En este caso el valor proporcionado es crisp.

ItoMin Se imputa el intervalo por su valor mínimo. En este caso el valor devuelto también es crisp.

ItoMax Se imputa el intervalo por su valor máximo. El valor devuelto también es crisp.

- Para valores fuzzy:

FtoI Se imputa el valor fuzzy por el intervalo asociado al α-corte 1. El valor devuelto es un intervalo.

FtoCG Se imputa el valor fuzzy por el valor crisp correspondiente a su centro de gravedad. El valor devuelto es crisp.

La posibilidad de imputar el mismo tipo de valor desde diferente técnicas, facilita la investigación en AID en dos direcciones: Por un lado facilita el análisis sobre el tipo de valor que ofrece la mejor alternativa a la naturaleza de la imperfección que aparece en los datos cuando se trabaja con técnicas que permiten procesar datos LQD de formas diferentes; por otro lado, facilita el análisis sobre que técnica de imputación genera el valor crisp más adecuado para el método a utilizar (este caso es para las técnicas que no permitan preprocesar datos LQD).

Debido a la facilidad en el diseño de las conexiones en un grafo, el usuario puede enlazar distintas técnicas de imputación. El módulo de control no verifica la presencia del tipo particular de LQD el cual el usuario ha indicado imputar. En el caso de no existir este tipo de LQD, la imputación no se ejecuta.

14.4.6 **Formatos de salida y formato de los datos de baja calidad**

Los componentes *LQD−>Format* y *Datasets−>Export* permiten incorporar al experimento la configuración de los formatos de salida, generando varios formatos rápida y fácilmente. Cada rama del grafo de un experimento genera una salida la cual puede ser una partición de los atributos cuando el nodo hoja de la rama es un componente *Partition* o puede ser un conjunto de conjuntos de datos con un particular formato si el nodo hoja de la rama es un componente *Datasets−>Export*.

El componente *Datasets−>Expor* establece un formato de salida para los conjuntos de datos generados por el experimento. Los formatos de salida pueden ser ARFF, KEEL, UCI, CSV y “Custom”. Para el caso de los formatos “Custom”, estos deben de ser definidos previamente en NIPip y cargados en la ventana de configuración del componente *Datasets−>Export*. Se puede especificar tantos formatos de salida como se quiera, indicando para cada uno de ellos las siguientes opciones:

- La semilla a utilizar (Seed).
- Si el experimento es una validación cruzada, hay que indicar el número de validaciones y el número de repeticiones de la misma.
- Si la salida requiere un fichero de train y otro de test, hay que especificar el porcentaje de ejemplos que pertenecerán al fichero train (el porcentaje para el fichero test se calcula automáticamente). También hay que indicar el número de repeticiones.

Además, la sintaxis para cada uno de los diferentes valores de LQD introducidos será por defecto según el formato. Si el usuario quiere modifcar el formato por defecto, deberá de indicarlo añadiendo el componente *LQD−>Format* para cada tipo de dato.

Los componentes *LQD->Format* permiten especificar cada uno de los formatos de salida de los valores de LQD en los diferentes conjuntos de datos. Particulamente, si es seleccionado la salida con etiqueta en el caso de los valores intervalares o fuzzy, el módulo de control semántico chequea que existe por lo menos una partición crisp o fuzzy definida previamente. Para el resto de valores, se mostrará el formato de salida por defecto.

14.4.7 **Caso de uso**

En esta sección vamos a presentar un caso de uso como un ejemplo del funcionamiento y proceso que hay que llevar a cabo para crear un experimento utilizando ExpNIPip.

Supongamos que queremos llevar a cabo un análisis experimental de la técnica de clasificación FRF_{LQD}, [42], con los siguientes objetivos: 1) analizar la robustez de la técnica FRF_{LQD} frente a la existencia de diferentes tipos de LQD, en particular a la existencia de datos missing, de valores intervalares y de valores fuzzy. Para realizar esto, queremos analizar si esta técnica funciona mejor con un conjunto de datos con ciertos porcentajes de valores missing o con los mismos porcentajes de valores intervalares o con los mismos porcentajes de valores fuzzy; 2) Analizar el comportamiento de FRF_{LQD} frente a diferentes técnicas de imputación de valores missing, es decir, imputamos lo valores missing añadidos en los atributos numéricos utilizando diferentes técnicas de imputación para ver qué técnica produce un mejor rendimiento de la técnica de clasificación.

En este ejemplo queremos obtener 10 conjuntos de datos en formato KEEL, [4], y en formato UCI, [77] de la siguiente forma:

1) Conjuntos de datos que contengan valores missing,

2) conjuntos de datos con valores missing imputados con diferentes técnicas,

3) conjuntos de datos con intervalos, y finalmente

4) conjuntos de datos con valores fuzzy.

Estos conjuntos de datos deben de ser adecuados para llevar a cabo una validación cruzada de tamaño 3 repetida 2 veces. El formato de salida debe de ser el específico que utiliza FRF_{LQD} para todos los conjunto de datos, y para ello, tenemos el formato definido previamente “frf.out”.

La Tabla 14.4 muestra los conjuntos de datos utilizados en los experimentos, indicando para cada uno de ellos el nombre, el número de atributos, el número de ejemplos, el número de valores de clases, y si el conjunto de datos contiene o no datos de baja calidad y el formanto de entrada.

Tabla 14.4: Conjuntos de datos empleados en el estudio

Conjunto de datos	$\|A\|$	$\|E\|$	$\|C\|$	**LQD**	**Formato**	**Conjunto de datos**	$\|A\|$	$\|E\|$	$\|C\|$	**LQD**	**Formato**
Apendicitis	7	106	2	No	KEEL	Wine	13	178	3	No	UCI
Pima Indian Diabetes	8	768	2	No	KEEL	Breast Cancer W.	32	569	2	No	UCI
Glass identification	9	214	7	No	KEEL	Iris Plant	4	150	3	No	UCI
Hepatitis	19	155	2	Yes	KEEL	Ionosphere	34	351	2	No	UCI
Horse-colic	2	368	2	Yes	KEEL	Zoo	17	101	7	No	UCI

14.4.7.1 **Diseñando el experimento con ExpNIPip**

Empezamos construyendo el grafo del experimento añadiendo dos componentes *Datasets−>Import*, uno para cada formato de entrada de los conjuntos de datos utilizados en este ejemplo (*Datasets−>Import−>KEEL* y *Datasets−>Import−>UCI*).

Para añadir un porcentaje de valores intervalares, incorporamos en el grafo un componente *Partition−>Crisp* que debe de aparecer en el flujo del grafo antes del componente *LQD−>Add−>Interval*. De la misma forma, creamos una partición fuzzy para añadir un porcentaje de valores fuzzy en los conjuntos de datos.

Después incorporamos un componente para añadir valores missing (componente *LQD−>Add−>Missing*).

Luego estos valores missing serán imputados utilizando tres técnicas diferentes de imputación, y para ello añadimos sus respectivos componentes *Imputation−>Missing−>NtoM*, *Imputation−>Missing−>NtoCM* y *Imputation−>Missing−>NKNN*.

Finalmente, añadimos el componente *Datasets−>Export−>Custom* indicando el número de repeticiones y el tamaño de la validación cruzada, además de cargar el formato de salida "custom" que hemos definido previamente "frf.out".

El grafo de la Figura 14.22 muestra el flujo de datos y conexiones que forman el experimento (los nodos del grafo representan los distintos componentes que hemos ido detallando anteriormente).

Todos los parámetros de los diferentes componentes se configuran seleccionando propiedades sobre el correspondiente icono. La Tabla 14.5 muestra los parámetros utilizados en la configuración de cada componente del experimento.

Cuando hemos configurado el experimento, podemos guardarlo (se generan los ficheros ".exp" y ".scr") para poder ejecutarlo en modo offline. Desde la pantalla de definición del experimento ejecutamos el experimento en modo online *Actions-Run* y podemos seguir la evolución del mismo accediendo a *Actions-Visualize*. El log creado durante la ejecución se guarda en un fichero ".log", en el cual se describe la secuencia de acciones llevadas a cabo duran-

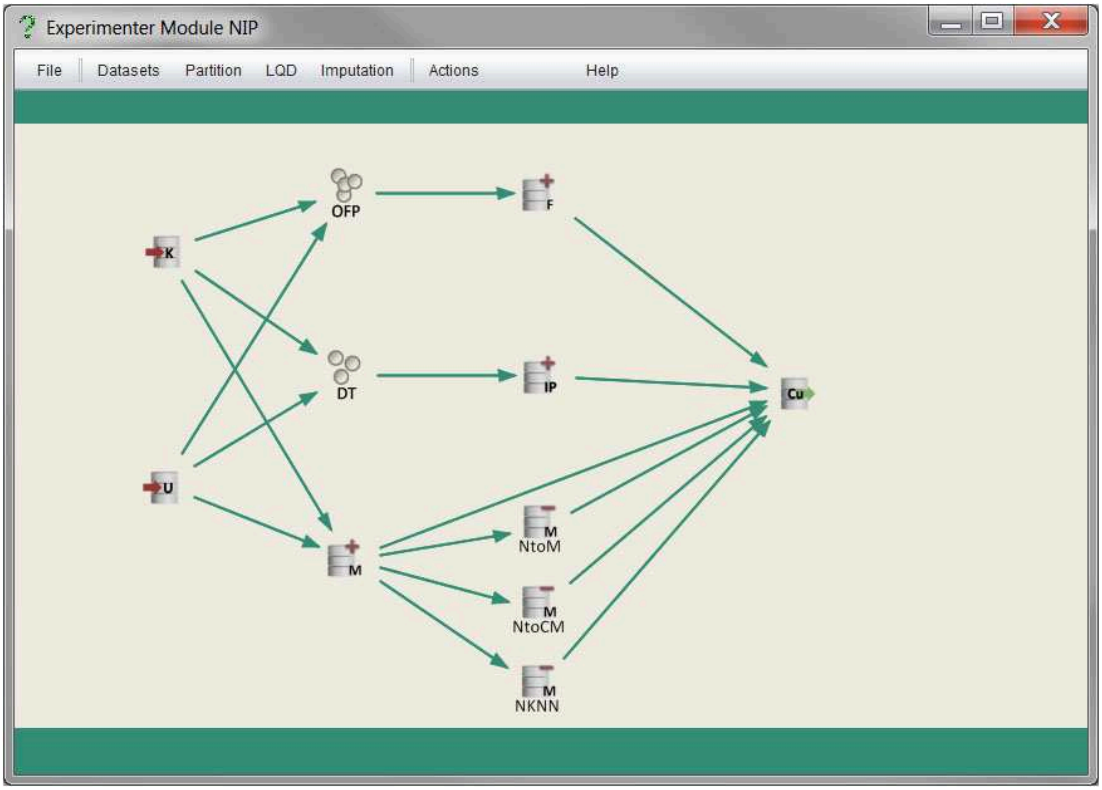


Figura 14.22: Representación gráfica del experimento

Tabla 14.5: Parámetros utilizados en los componentes

Componentes	**Parámetros**
Datasets->Import->UCI	–
Datasets->Import->KEEL	–
Partition->Crisp->DT	Seed-59, MinEx-1, %Pure-100, Thread-2, atributo clase
Partition->Fuzzy->OFP	Seed-59, MinEx-1, %Pure-100, Thread-2, Gene-100, Popu-20, PCross-0.9, PMut-0.2, atributo clase
LQD->Add->Missing	Seed-0, 5 % en todos los atributos
LQD->Add->Interval	Seed-0, 10 % en todos los atributos numéricos
LQD->Add->fuzzy	Seed-0, 5 % en todos los atributos numéricos
Imputation->Missing->NtoM	–
Imputation->Missing->NtoCM	–
Imputation->Missing->NKNN	K_value-sqrt, Thresdhold-0, Distance-F1
Datasets->Export->Custom	"frf.out", N. folds-3, Repetitions-2, Seed-0

te la ejecución y se indica si ha habido algún error y no se ha podido ejecutar una parte del experimento.

Para el caso de uso que estamos analizando, la Figura 14.23 muestra parte de la estructura de directorios y ficheros generados al ejecutar el experimento.

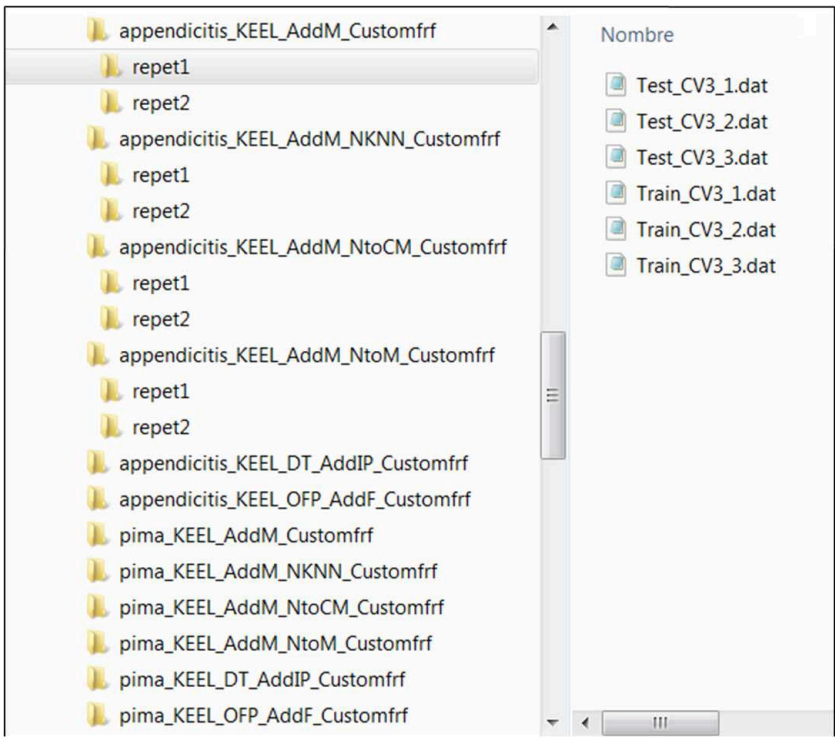


Figura 14.23: Vista parcial de la salida generada por el experimento

Los enlaces con el componente *Datasets−>Export* definen los onjuntos de datos que se van a generar. Así, para cada conjunto de datos *DS* en el experimento se generan seis directorios en el directorio especificado por el usuario en el componente *Datasets−>Export*: "*DS_KEEL_OFP_AddF_Customfrf*" contiene los conjuntos de datos con valores fuzzy de la partición generada por la técnica *OFP*, "*DS_KEEL_DT_AddIP_Customfrf*" contiene los conjuntos de datos con intervalos de una partición generada con la técnica *DT*, "*DS_KEEL_AddM_Customfrf*" contiene conjuntos de datos con valores missing, "*DS_KEEL_AddM_NtoM_Customfrf*" contiene conjuntos de datos con valores missing imputados utilizado la técnica *NtoM*, "*DS_KEEL_AddM_NtoCM_Customfrf*" contiene conjuntos de datos con valores missing imputados con la técnica *NtoCM* y "*DS_KEEL_AddM_NKNN_Customfrf*" contiene conjuntos de datos con valoress missing imputados con la técnica *NKNN*. Cada una de estos seis directorios contiene los respectivos conjuntos de datos descritos, tal y como se ha indicado en el experimento, para una validación de tamaño 3 repetidas 2 veces. El nombre de cada directorio describe la rama del grafo que lo ha generado.

Además, las particiones crisp y fuzzy también son parte del experimento por lo cual también se generan en el directorio especificado por el usuario al configurar tales componentes. Una parte del fichero ".scr" generado por el experimento se muestra en el Figura 14.24.

```
------------------------------------------------

Import Datasets Format KEEL
      glass.dat: Path-H:\keel
      hepatitis.dat: Path-H:\keel
      pima.dat: Path-H:\keel
      appendicitis.dat: Path-H:\keel
      horse-colic.dat: Path-H:\keel

Discretization Method-OFP_CLASS Path-H:\partitionf Seed-59 MinEx-1 %
Pure-100 Thread-2 Gene-100 Popu-20 PCross-0.9 PMut-0.2
      glass.dat TypeGlass
      hepatitis.dat Class
      pima.dat Class
      appendicitis.dat Class
      horse-colic.dat Surgical_lesion?

Add Fuzzy Seed-0
      glass.dat: (RI-4.9999995% Na-4.9999995% Mg-4.9999995% Al-
4.9999995% Si-4.9999995% K-4.9999995% Ca-4.9999995% Ba-4.9999995% Fe-
4.9999995%)
      hepatitis.dat: (Age-4.9999995% Sex-4.9999995% Steroid-4.9999995%
Antivirals-4.9999995% Fatigue-4.9999995% Malaise-4.9999995% Anorexia-
4.9999995% LiverBig-4.9999995% LiverFirm-4.9999995% SpleenPalpable-
4.9999995% Spiders-4.9999995% Ascites-4.9999995% Varices-4.9999995%
Bilirubin-4.9999995% AlkPhosphate-4.9999995% Sgot-4.9999995% AlbuMin-
4.9999995% ProTime-4.9999995% Histology-4.9999995%)
      pima.dat: (Preg-4.9999995% Plas-4.9999995% Pres-4.9999995% Skin-
4.9999995% Insu-4.9999995% Mass-4.9999995% Pedi-4.9999995% Age-
4.9999995%)
      appendicitis.dat: (At1-4.9999995% At2-4.9999995% At3-4.9999995%
At4-4.9999995% At5-4.9999995% At6-4.9999995% At7-4.9999995%)
      horse-colic.dat: (Hospital_Number-4.9999995% Rectal_temperature-
4.9999995% Pulse-4.9999995% Respiratory_rate-4.9999995%
Nasogastric_reflux_PH-4.9999995%  Packed_cell_volume-4.9999995%
Total_protein-4.9999995% Abdomcentesis_total_protein-4.9999995%)

Export Dataset Format Custom: FileFormat-"frf.out" Path-H:\results
FoldCv-3 Repe-2 Seed-10.0
```

Figura 14.24: Parte del fichero ".scr" del experimento

CAPÍTULO

15

Conclusiones parciales y aportaciones

15.1 Conclusiones

Durante esta última parte del trabajo hemos presentado el diseño y uso de una herramienta software, NIPip, para preprocesar conjuntos de datos que contienen datos de baja calidad explícitamente. El principal motivo para el diseño de esta herramienta, es que aunque actualmente podemos encontrar una variedad de herramientas tanto de código abierto como privadas para manejar y preprocesar conjuntos de datos, no existe una, que sea capaz de preprocesar estos conjuntos de datos cuando contienen de forma explícita LQD. Cuando diseñamos o adaptamos técnicas en el ámbito del AID para que traten con LQD, no existe un gran variedad de conjuntos de datos que permitan analizar la robustez y el comportamiento de la técnicas propuestas. Con la herramienta propuesta facilitamos el proceso de creación de un repositorio de conjuntos de datos que sirva como marco de referencia común en este campo de investigación.

La herramienta que hemos propuesto NIPip se compone de dos paquetes, un primer paquete más interactivo donde se puede trabajar con un conjunto de datos cada vez y otro paquete más orientando a la experimentación capaz de preprocesar varios conjuntos de datos simultáneamente (ExpNIPip). La característica principal de esta herramienta es la capacidad de llevar a cabo un preproceso de los datos de forma robusta tratanto LQD.

Entre todas las capacidades del primer paquete de NIPip que hemos descrito en detalle a lo largo del Capítulo 14 debemos de destacar su interfaz amigable con el usuario, fácil de utilizar y orientada a la manipulación y la construcción de conjuntos con LQD. Además los

conjuntos de datos puede ser importados bien en un formato estándar o en un formato propio predefinido por el usuario. Es importante tener en cuenta que los conjuntos de datos importados pueden contender o no LQD explícitamente. Otra característica de la herramienta es que permite eliminar, normalizar y cambiar el tipo de los atributos, además de proporcionar cierta información estadística de cada uno de los atributos y cierta información descriptiva y útil del conjunto de datos importado. También permite manejar y añadir diferentes tipos de LQD como valores missing, valores fuzzy, intervalos, subconjuntos crisp/fuzzy, ruido, etc. Una vez que se ha preprocesado todo el conjunto de datos hay que exportarlo a un formato estándar o un formato predefinido por el usuario generando un fichero único o una validación cruzada. Tras presentar toda la funcionalidad del primer paquete de NIPip, hemos mostrado mediante unos casos de estudio el funcionamiento de este paquete.

En este capítulo también hemos presentado el paquete orientado a la experimentación con grandes conjuntos de datos de NIPip (ExpNIPip). Este paquete está más orientado a la preparación de experimentos que conllevan el trabajo con grandes conjuntos de datos simultáneamente. El paquete ExpNIpip tiene las mismas características del paquete NIPip y se añaden algunas nuevas, como por ejemplo nuevas técnicas de imputación y de discretización. El preprocesamiento de los conjuntos de datos se define de forma gráfica basándose en el flujo de datos. Esto permite definir diversas secuencias funcionales sobre grandes conjuntos de datos, llevando a cabo el preprocesamiento de forma fácil y rápida. Además el paquete experimentador de NIPip permite ejecutar los experimentos diseñados en modo on-line y off-line y seguir la ejecución de un experimento mediante un log que puede ser visualizado en cualquier momento por el usuario.

Como cualquier herramienta software, NIPip está abierta a futuras ampliaciones que irán principalmente enfocadas a la incorporación de la fase de minería de datos en la misma recopilando técnicas que permitan el trabajo con LQD, incorporación de técnicas para selección de atributos y selección de ejemplos desde conjuntos de datos con LQD, ampliación de las técnicas de imputación y discretización que actualmente existen en la herramienta, etc.

Como conclusión, la herramienta NIPip está diseñada centrándose en la fase de preprocesamiento de datos del AID, con el objetivo de poder tratar con los conjuntos de datos que contienen de forma directa valores de baja calidad o bien añadir este tipo de valores para evaluar el diseño de nuevas propuestas que permitan el manejo de este tipo de datos. En definitiva, el objetivo de esta herramienta es la generación y transformación de conjuntos de datos que sirvan como un marco de trabajo común para llevar a la cabo la comparación y validación de las diferentes modelos del AID con tratamiento de LQD que se desarrollen.

15.2 Aportaciones más relevantes

[34] J.M. Cadenas, M.C. Garrido, R. Martínez. A tool to manage low quality datasets. WCCI 2012 IEEE World Congress on Computational Intelligence (in FUZZIEEE 2012), 1154-1161, Brisbane, Australia, 2012.

Congreso CORE A

[49] J.M. Cadenas, M.C. Garrido, R. Martínez, E. Serrano. "NIP imperfection processor Registro de la propiedad intelectual. 18-07-2012, nº de asiento registral 08/2012/700.

Patente

[39] J.M. Cadenas, M.C. Garrido, R. Martínez. NIP - An Imperfection Processor to Data Mining datasets. International Journal of Computational Intelligence Systems 6(Supplement 1), 3–17, 2013.

JCR 0.451

[143] R. Martínez, J.M. Cadenas, M.C. Garrido, A. Martínez. Imputing missing values from Low Quality Data by NIP tool. IEEE International Conference on Fuzzy System (FUZZIEEE 2013), 1–8, Hyderabad, India, 2013.

Congreso CORE A

[142] R. Martinez, J.M. Cadenas, M.C. Garrido. The experimenter environment of the NIP imperfection. IEEE International Conference on Fuzzy System (FUZZIEEE 2014), 1–8, Pekín, China, 2014.

Congreso CORE A

Parte V

CONCLUSIONES Y VÍAS FUTURAS

Conclusions

The methodologies offered by Softcomputing and Intelligent Data Analysis have been the focus of this doctoral thesis. Throughout the different parts and chapters in which this work is divided, we have focused on two of the main phases of intelligent data analysis, namely the preprocessing phase and the data mining phase, with the aim of incorporating the treatment of low quality data. As the first chapter of this work has described, the motivation to focus on this type of data is on the one hand, the number of real problems where the available information is imperfect and if the techniques of intelligent data analysis can not work directly with this information, we should transform it. Some valuable information may be lost in this transformation, seeing the loss of information reflected in the results. On the other hand, the number of techniques that can explicitly deal with low quality data in datasets is quite low compared to the number of techniques that we can find in literature.

In the first part of the tesis, the concepts of Softcomputing and Intelligent Data Analysis have been described. Besides, it has been presented how these two concepts are complementary and they can be used together to increase flexibility of techniques and give them a greater ability to handle different types of information. After that, an analysis, of the different types of low quality data and of how techniques work with this data, have been performed. Then, we focus on the Intelligent Data Analysis phases of data preprocessing and data mining, studying in depth, although without been exhaustive, those techniques based on some of the different methodologies offered by Softcomputing. In other words, those techniques that can somehow deal with low quality data.

The three following parts of this doctoral thesis are summarized graphically in Figure 15.1.

In Parts II and III, new techniques in the data preprocessing phase and new extensions of techniques in the data mining phase have been presented. Each technique presented is able to work with low quality data when these data appear explicitly in datasets. Related to the data mining phase, an ensemble Fuzzy Random Forest (FRF) has been extended in order that this ensemble can work with low quality data. This therefore has implied to have to extend the fuzzy decision tree, that is the base of this ensemble, to provide it with the ability to deal with

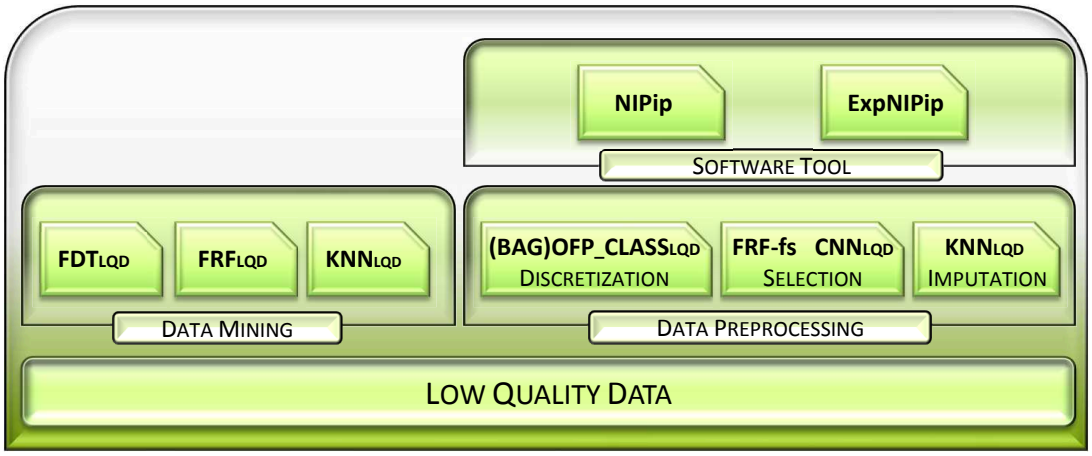


Figura 15.1: General scheme of the thesis

this type of data. To do this, a similarity measure has been proposed. This measure provides a membership degree of a low quality value to the possible nodes where this value may descend. An aspect to take into account in that extension is that having a low quality data in the input, the output is also of low quality data, in place of having a specific accuracy value, an interval is obtained. This interval includes the uncertainty which appears in the input data. The assessment, performed to validate this extension of the FRF, has shown that the results obtained by working directly with low quality data are optimistic. Furthermore, in comparison with others works, that also deal with the same data type, FRF obtains better results globally. These results have been validated by statistical tests. In addition, in the assessment, in one of the experiments, it has been verified that when the low quality data is transformed into accurate information, a loss of information occurs during this transformation. Subsequently, this loss is reflected in a loss of accuracy in the results. Therefore, we can conclude that the extensions of the proposed techniques have obtained promising results. Besides, not only do these techniques have a similar behavior, and in some situations better, when working with crisp data compared to other techniques in literature, but also these techniques have the ability to handle low quality data.

In addition to the extension of the ensemble FRF, the technique of K-nearest neighbors (KNN) has been also extended, to be able to deal with low quality data. Here, the technique has been extended with dual function. Firstly to perform classification and/or regression tasks and secondly to carry out the imputation of missing values. To achieve this extension, a series of distance measures, to calculate the distance between an example and its neighbors, have been designed. The designed measure takes into account the imprecision degree of the input data when selecting its nearest neighbors. As KNN has problems working with large datasets, a very simple and easy technique to select examples has been extended, specifically, the technique called “Condensed Nearest Neighbors” (CNN). CNN has been extended to be able to work

directly with low quality information and so to reduce the number of examples, taking the most characteristic examples. When evaluating the performance of these two techniques, the results obtained are quite optimistic and again coming to the conclusion that when working with low quality data at the input, the accuracy obtained is also of low quality, being more imprecise but more in line with reality. In conclusion, although the results obtained with the technique KNN both in imputation and in classification are quite optimistic, it is important to note that the extension of the technique of imputation KNN and of the technique of condensation CNN are initial works and are still under study and improvement.

Regarding the data preprocessing phase, apart from the imputation technique already mentioned, two techniques with their respective improvements have been designed; a technique of discretization of numerical attributes and a technique of attribute selection. The first technique is to discretize the numerical attributes by means of fuzzy partitions. This technique is categorized as a hybrid technique and consists of a fuzzy decision tree and a genetic algorithm, and is able to work with low quality data. Besides, the last improvement designed is able to obtain more global partitions when datasets have proportionately a larger number of attributes than of examples. The discretization technique and its improvements proposed have been assessed using several datasets both with low quality data and without low quality data. In addition, an experimental comparison with other works of literature, using datasets of real problems, has been performed. Also the different improvements on the discretization algorithm have been compared in order to analyze the behavior globally. The results obtained are quite good and optimistic compared to other techniques of literature that work both with crisp data and with low quality data. In conclusion we can say that the technique proposed and the improvements of this technique are robust and perform when when working with crisp data and also have the ability to work explicitly with low quality data, obtaining quite promising results.

The other technique proposed, within the data preprocessing phase, is a attribute selection technique that is able to work explicitly with low quality data. The attribute selection technique is a hybrid technique because it uses a filtering technique and a wrapper technique. This technique firstly uses the extended ensemble FRF, FRF_{LQD}, to collect relevant information about the attributes. After, the information obtained is combined to create a ranking of importance of the attributes, to subsequently obtain a subset of the most relevant attributes using a classification technique. To evaluate and analyze the behavior of the technique, a series of experiments using different datasets have been performed. A comparison of the results with other techniques proposed in literature to select attributes have been performed. Specifically, microarray datasets have been used to assess the robustness of the technique proposed when working with crisp data. The results obtained are very satisfactory, being equal to one of the techniques of literature about attributes selection and improving the behavior of others. Besides, the experiments

performed with low quality data have given also satisfactory and encouraging results. That is why, we can conclude that the technique is robust and that its behavior is stable working both with low quality data and without low quality data.

The last proposal presented has been the design and implementation of a software tool for preprocessing data of low quality. This tool consists of two environments, a first more interactive environment and a second environment that is more orientated to the experimentation. Among the main characteristics of the tool, we must emphasize the following: the flexibility in the input and output data formats, the ability to add and/or to remove low quality data of different types either turning them into crisp data or into other low quality data that is less imprecise, the capacity to discretize numerical attributes even when the dataset contains low quality data, etc.. This tool includes some of the techniques presented throughout the work, but in the future new characteristics and techniques will be incorporated.

As a final comment, we must add that all the techniques presented are easily improvable and expandable from the computational point of view, since they are designed and implemented modularly. Moreover, the techniques use parallelization functions when they are doing a lot of calculation although we have not focused at any point on the execution times.

Conclusiones

Las metodologías que nos ofrece el Softcomputing y el AID han sido el tema central de esta tesis doctoral. A lo largo de las diferentes partes y capítulos en las cuales se divide este trabajo, nos hemos centrado en dos de las principales fases del análisis inteligente de datos, la fase de preprocesamiento de los datos y la fase de minería de datos con el objetivo de incorporar el tratamiento de LQD. Tal y como describimos en el primer capítulo de este trabajo, la motivación de centrarnos en este tipo de datos es por un lado, la cantidad de problemas reales donde la información disponible es imperfecta y si las técnicas del AID no pueden trabajar con esta información directamente, debemos de transformarla, pudiendo perder información valiosa en dicha transformación, viéndose dicha pérdida de información reflejada en los resultados. Por otro lado, el número de técnicas que pueden tratar de forma explícita con datos de baja calidad en los conjuntos de datos es bastante escaso comparado con el número de técnicas que nos podemos encontrar en la literatura.

En la primera parte del trabajo, primero hemos descrito los conceptos del Softcomputing y del Análisis Inteligente de Datos y cómo estos dos conceptos son complementarios y pueden ser utilizados conjuntamente para flexibilizar las técnicas y darle una mayor capacidad de manejar diferentes tipos de información. Después hemos realizado un análisis general de los diferentes tipos de datos de baja calidad y de cómo las técnicas trabajan con ellos. Luego nos hemos centrado en las fases del AID de preprocesamiento de datos y de minería de datos, estudiando más en profundidad, aunque sin ser exhaustivos, aquellas técnicas basadas en alguna de las diferentes metodologías que nos ofrece el Softcomputing, en otras palabras, aquellas técnicas que puedan tratar de alguna forma con datos de baja calidad.

Las tres siguientes partes del trabajo, se encuentran resumidas de forma gráfica en la Figura 15.2.

En las Partes II y III, hemos presentado nuevas técnicas en el caso de la fase de preprocesamiento de los datos y nuevas extensiones de técnicas en el caso de la fase de minería de datos, pudiendo todas las técnicas trabajar con datos de baja calidad que aparecen de forma

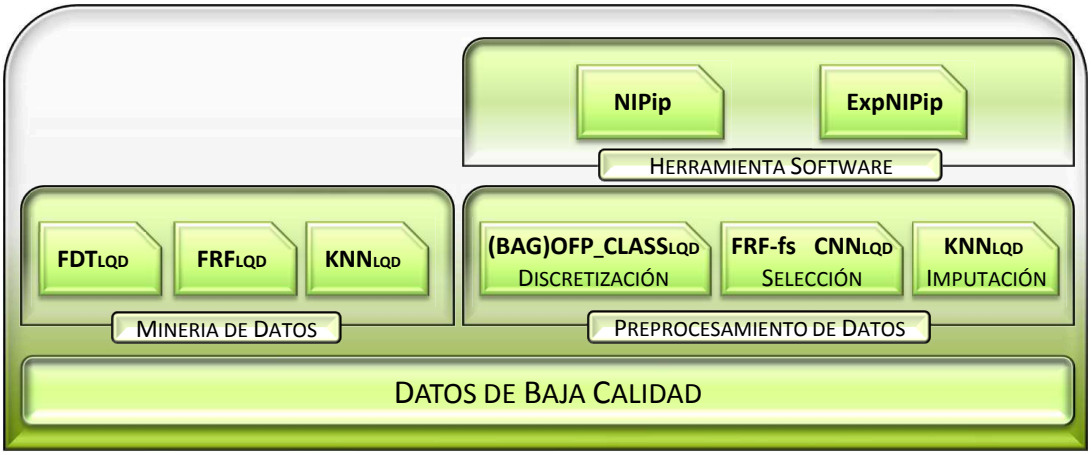


Figura 15.2: Esquema general de la tesis

explícita en los conjuntos de datos. Relacionado con la fase de minería de datos hemos extendido el ensamble Fuzzy Random Forest (FRF) para que pueda trabajar directamente con datos de baja calidad. Esto ha implicado extender el árbol de decisión fuzzy, que toma como base este ensamble, para que pueda tratar con este tipo de datos. Para ello, hemos propuesto una medida de similitud que nos proporciona un grado de pertenencia de un valor de baja calidad a los posibles de nodos donde dicho valor descenderá. Un aspecto a tener en cuenta en dicha extensión es que al disponer de una entrada de datos de baja calidad, la salida de los datos también es de baja calidad, por lo que en lugar de tener un valor concreto de precisión, obtenemos un intervalo que recoge la incertidumbre que se presenta en los datos de entrada. La evaluación que hemos llevado a cabo para validar esta extensión del FRF, nos ha mostrado que los resultados obtenidos trabajando directamente con datos de baja calidad son optimistas y que en comparación con otro trabajo que también trabaja con el mismo tipo de datos, FRF obtiene mejor resultado globalmente. Estos resultados han sido validados mediante test estadísticos. Además, en la evaluación hemos podido comprobar, en uno de los experimentos, que cuando tratamos de transformar la información de baja calidad en información precisa, se produce una perdida de información en dicha transformación que se traduce en una pérdida de precisión en los resultados. Por lo tanto, podemos concluir que las extensiones de las técnicas propuestas han obtenido unos resultados optimistas que tienen un comportamiento similar y en algunas situaciones mejor cuando trabajan con datos crisp con respecto a otras técnicas de la literatura, pero además tienen la capacidad de manejar LQD.

Además de la extensión del ensamble FRF, también hemos extendido la técnica de K-vecinos más cercanos (KNN), para que sea capaz de tratar con datos de baja calidad. En este caso, la técnica la hemos extendido con funcionalidad doble, por un lado para realizar tareas de clasificación y/o regresión y por otro para llevar a cabo la imputación de valores missing. Para

lograr esta extensión hemos diseñado una serie de medidas distancia para calcular la distancia entre un ejemplo y sus vecinos. La medida diseñada toma en cuenta el grado de imprecisión de los datos de entrada a la hora de seleccionar a sus vecinos más cercanos. Como KNN tiene problemas al trabajar con conjuntos de datos grandes, hemos extendido una técnica muy simple y sencilla de selección de ejemplos, la técnica de condensación CNN, para que pudiera trabajar directamente con información de baja calidad y así reducir el número de ejemplos, tomando los ejemplos más característicos. Al evaluar el comportamiento de estas dos técnicas hemos obtenido resultados bastante optimistas y de nuevo llegando a la conclusión de que al trabajar con datos de baja calidad en la entrada, la precisión obtenida también es de baja calidad, siendo más imprecisa pero ajustándose más a la realidad. En conclusión, aunque los resultados obtenidos con la técnica KNN tanto en imputación como en clasificación son bastante optimistas es importante tener en cuenta que la extensión de la técnica KNN y de la técnica de condensación CNN, son trabajos iniciales y que todavía están en fase de estudio y de mejora.

En lo referente a la fase de preprocesamiento de los datos, aparte de la técnica de imputación que ya hemos comentado, hemos diseñado dos técnicas con sus respectivas mejoras; una técnica de discretización de atributos numéricos y una técnica de selección de atributos. La primera técnica es para discretizar los atributos numéricos mediante particiones fuzzy. Esta técnica está categorizada como una técnica híbrida y se compone de un árbol de decisión fuzzy y de un algoritmo genético y es capaz de trabajar con datos de baja calidad. Además en la última mejora que hemos diseñado es capaz de obtener particiones más globales cuando los conjuntos de datos tienen un número de atributos proporcionalmente mayor que de ejemplos. La técnica de discretización y las mejoras sobre la misma propuestas, han sido evaluadas utilizando varios conjuntos de datos tanto con LQD como sin LQD. Además, hemos llevado a cabo una experimentación en comparación con otros trabajos de la literatura utilizando conjuntos de datos de problemas reales. También hemos comparado las diferentes mejoras del algoritmo de discretización, para analizar el comportamiento de la técnica de forma global. Los resultados que hemos obtenido son resultados bastante buenos y optimistas comparados con otros métodos de la literatura tanto trabajando con datos crisp como con LQD. Como conclusión podemos indicar que la técnica propuesta y las mejoras de dicha técnica son robustas y tienen un buen comportamiento trabajando con datos crisp y además tienen la capacidad de poder trabajar de forma explícita con LQD, obteniendo unos resultados bastante prometedores.

La otra técnica, dentro de la fase de preprocesamiento de datos, que hemos propuesto es de selección de atributos capaz de trabajar explícitamente con LQD. La técnica de selección de atributos es una técnica híbrida ya que utiliza una técnica de filtrado y una técnica wrapper. Esta técnica primeramente hace uso del ensamble FRF extendido FRF_{LQD} para recopilar

información relevante acerca de los atributos. Después la información obtenida es combinada para crear un ranking de importancia de los atributos, para posteriormente mediante una técnica de clasificación, obtener un subconjunto de los atributos más relevantes. Para evaluar y analizar el comportamiento de la técnica hemos llevado a cabo una serie de experimentos utilizado diferentes conjuntos de datos. Hemos realizado una comparación de los resultados con otras técnicas propuestas en la literatura para seleccionar atributos, concretamente, hemos utilizado conjuntos de datos de microarrays para evaluar la robustez de la técnica propuesta cuando trabaja con datos crisp. Los resultados obtenidos son muy satisfactorios igualando a una de las técnicas de la literatura en selección de atributos y mejorando el comportamiento de otras. Además, debemos añadir que los experimentos llevados a cabo con datos de baja calidad también han resultados satisfactorios y optimistas, pudiendo concluir que la técnica es robusta y que su comportamiento es estable trabajando con LQD y sin LQD.

La última propuesta que hemos presentado ha sido el diseño e implementación de una herramienta software para preprocesar datos de baja calidad. Esta herramienta se compone de dos entornos, un primer entorno más interactivo y un segundo entorno más orientado a la experimentación. Entre las características principales de la herramienta debemos de destacar su flexibilidad en los formatos de entrada y salida de los datos, la capacidad de añadir y/o eliminar LQD de diferentes tipos, bien transformándolos en datos crisp o en otro tipo de dato de baja calidad menos impreciso, la capacidad de discretizar los atributos numéricos incluso cuando el conjunto de datos contiene LQD, etc.. En esta herramienta hemos agrupado algunas de las técnicas que hemos presentado a lo largo del trabajo, aunque como trabajo futuro todavía se pueden incorporar nuevas funcionalidades y técnicas.

Como comentario final, debemos de añadir que todas las técnicas presentadas son mejorables y ampliables de forma fácil desde el punto de vista computacional, puesto que están diseñadas e implementadas modularmente. Además, las técnicas utilizan funciones de paralelización cuando tienen que realizar una gran cantidad de cálculo aunque no nos hayamos centramos en ningún momento en los tiempos de ejecución.

Vías futuras

A pesar de que los resultados experimentales han indicado que el comportamiento de las técnicas presentadas a lo largo de este trabajo es estable y robusto, todavía son muchas las tareas que se pueden mejorar y/o ampliar, no solo referente a las técnicas presentadas, sino a la exploración de nuevas técnicas con el fin de adaptarlas para que puedan tratar con LQD. Tal y como hemos llevado a cabo a lo largo de este trabajo y teniendo en cuenta que nos hemos centrado del Análisis Inteligente de Datos, en las fases de preprocesamiento de datos y minería de datos, vamos a dividir las vías futuras de acuerdo a los posibles elementos a mejorar en estas dos fases, incluyendo también la ampliación y mejora de la herramienta software propuesta.

Comenzando por la fase de preprocesamiento de datos, a continuación indicamos la lista de posibles trabajos y/o mejoras que se pueden llevar a cabo en el futuro:

- Si nos centramos en la discretización de los datos, las posibles mejoras a llevar a cabo son:
  - Estudiar nuevas medidas de similitud para el árbol de decisión fuzzy presentado en la primera parte del algoritmo de discretización, tratando siempre de mantener la estabilidad y la robustez del árbol de decisión fuzzy.
  - Estudiar nuevas medidas de entropía fuzzy o posibles alternativas para el cálculo del valor fitness del algoritmo genético utilizado en la segunda etapa del algoritmo de discretización.
  - Para tratar con conjuntos de datos con pocos ejemplos, podríamos realizar un análisis sobre la utilización de boosting u otras técnicas, en ambas etapas del algoritmo de discretización en lugar del bagging actualmente utilizado. Además habría que realizar un análisis comparativo para ver que técnica trabaja mejor con conjuntos de datos con pocos ejemplos y obtiene mejores resultados.
- Respecto a la selección de atributos, posibles incorporaciones y mejoras que se pueden llevar a cabo son:

  - Obtener resultados utilizando como clasificador en la etapa wrapper del método de selección otro algoritmo diferente a Fuzzy Random Forest, siempre teniendo en cuenta que el nuevo algoritmo debe de ser capaz de tratar con datos de baja calidad.

- Desde el punto de vista de la imputación de valores missing, los trabajos futuros pendientes de desarrollar son:

  - Estudiar y analizar nuevas medidas para calcular la distancia entre los ejemplos con valores missing teniendo en cuenta los diferentes tipos de datos de baja calidad.
  - Analizar nuevos métodos de selección de ejemplos para tratar los problemas del método KNN al trabajar con conjuntos de datos de gran tamaño.
  - Respecto a la técnica CNN de condensación de ejemplos, estudiar nuevas medidas para ordenar los ejemplos según la imperfección de los mismos y realizar un estudio comparativo para comprobar cómo afecta el uso de diferentes medidas de la imperfección en los ejemplos a los resultados.

Si nos centramos en la etapa de minería de datos, más concretamente en la extensión que hemos realizado de la técnica Fuzzy Random Forest, posibles mejoras y modificaciones a llevar a cabo en el futuro:

- Extender el árbol de decisión fuzzy, FDT, para regresión. Al extender el FDT, por consecuencia extendemos también el ensamble FRF. Como la técnica de selección de ejemplos que proponemos utiliza el FRF para obtener la información acerca de la importancia de los atributos, por extensión esta técnica podría trabajar en la tarea de regresión, aunque habría que estudiar y analizar como le afectan los cambios de la extensión a regresión del ensamble FRF.

- Propuesta de nuevas medidas de similitud para trabajar con datos fuzzy e intervalares que aparezcan directamente en los datos.

Por último respecto a la herramienta NIP imperfection processor, los trabajos futuros pendientes de desarrollar son:

- Añadir nuevas técnicas de discretización tanto fuzzy como crisp.

- Incorporar nuevas técnicas de imputación predictivas, tanto de valores missing como de otros tipos de LQD, por ejemplo valores fuzzy e intervalos.

- Añadir nuevos módulos con técnicas de selección de ejemplos y de selección de atributos que puedan tratar con datos de baja calidad explícitamente.

- Incorporar nuevos tipos de validación cruzada en el modulo de salida de los dos entornos de NIPip, por ejemplo la validación estratificada.
- En el entorno experimenter de la herramienta añadir nuevas formas de paralelización de las ejecuciones para añadir una mayor capacidad de cómputo, actualmente llevamos a cabo una paralelización basada en memoria compartida.

Referencias

[1] M. H. Aghdam, N. Ghasem-Aghaee, and M. E. Basiri. Text feature selection using ant colony optimization. *Expert systems with applications*, 36(3):6843–6853, 2009. 42

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th International Conference Very Large Data Bases, VLDB*, volume 1215, pages 487–499, 1994. 27, 59

[3] H. Ahn, H. Moon, M. J. Fazzari, N. Lim, J. J. Chen, and R. L. Kodell. Classification by ensembles from random partitions of high-dimensional data. *Computational Statistics & Data Analysis*, 51(12):6166–6179, 2007. 85

[4] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17(2–3):255–287, 2011. 205, 220, 237

[5] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesús, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009. 205

[6] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc Natl Acad Sci U S A.*, 96:6745–6750, 1999. 166

[7] F. Angiulli. Fast nearest neighbor condensation for large data sets classification. *Knowledge and Data Engineering, IEEE Transactions on*, 19(11):1450–1464, 2007. 187

[8] S.C. Angoss Software Corporation. Knowledge seeker: Data mining software with predictive analytics made intuitive for business and technical users. http://www.angoss.com/predictive-analytics-software/software/knowledgeseeker/, 2012. 204

[9] W. H. Au, K. C. Chan, and A. K. Wong. A fuzzy approach to partitioning continuous attributes for classification. *IEEE Tran, Knowledge and Data Engineering*, 18(5):715–719, 2006. 27, 124, 136

[10] W. H. Au, K. C. Chan, and A. K. Wong. A fuzzy approach to partitioning continuous attributes for classification. *Knowledge and Data Engineering, IEEE Transactions on*, 18(5):715–719, 2006. 30

[11] J. Barnard and X. L. Meng. Applications of multiple imputation in medical studies: from aids to nhanes. *Statistical Methods in Medical Research*, 8(1):17–36, 1999. 47

[12] G. E. Batista and M. C. Monard. A study of k-nearest neighbour as an imputation method. *HIS*, 87:251–260, 2002. 48

[13] G. E. Batista and M. C. Monard. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6):519–533, 2003. 21, 51, 184, 218

[14] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *Neural Networks, IEEE Transactions on*, 5(4):537–550, 1994. 42, 46

[15] S. D. Bay. Multivariate discretization for set mining. *Knowledge and Information Systems*, 3(4):491–512, 2001. 30

[16] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel. *KNIME: The Konstanz information miner*. Springer, 2008. 206

[17] F. Berzal, J. C. Cubero, N. Marın, and D. Sánchez. Building multi-way decision trees with numerical attributes. *Information Sciences*, 165(1):73–90, 2004. 30, 69

[18] J. C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers, 1981. 34, 231

[19] J. C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers, 1981. 35

[20] P. P. Bonissone. Soft computing: the convergence of emerging reasoning technologies. *Soft computing*, 1(1):6–18, 1997. 10, 12

[21] P. P. Bonissone, J. M. Cadenas, M. C. Garrido, and R. A. Díaz-Valladares. A fuzzy random forest. *International Journal of Approximate Reasoning*, 51(7):729–747, 2010. 85, 86, 89

[22] P.P. Bonissone. *Uncertainty Management in Information Systems: From Needs to Solutions*, chapter Approximate reasoning systems: handling uncertainty and imprecision in information systems, pages 369–395. A. Motro and Ph. Smets, Eds. Kluwer Academic Publishers, 1997. 19

[23] M. Boulle. Khiops: A statistical discretization method of continuous attributes. *Machine Learning*, 55(1):53–69, 2004. 31

[24] L. Breiman. Bagging predictors. *Maching Learning*, 24(2):123–140, 1996. 132

[25] L. Breiman. Heuristics of instability and stabilization in model selection. *Annals of Statistics*, 24(6):2350–2383, 1996. 132, 133

[26] L. Breiman. Random forests. *Machine Learning*, 43:5–32, 2001. 43, 85, 167

[27] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984. 50

[28] C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *arXiv preprint arXiv:1106.0219*, 2011. 22

[29] J. M. Cadenas, M. C. Garrido, , R. Martinez, and A. Martínez. Consensus operators for decision making in fuzzy random forest ensemble. In *11th International Conference on Intelligent Systems Design and Applications (ISDA 2011)*, pages 1377–1382. IEEE, 2011. 89, 112

[30] J. M. Cadenas, M. C. Garrido, and R. Martínez. Constructing fuzzy partitions from imprecise data. In *International Conference on Fuzzy Computation Theory and Applications, in proceedings (FCTA 2011)*, pages 379–388, 2011. 128, 196

[31] J. M. Cadenas, M. C. Garrido, and R. Martinez. Learning in a fuzzy random forest ensemble from imperfect data. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC2011)*, pages 277–282. IEEE, 2011. 94, 112

[32] J. M. Cadenas, M. C. Garrido, and R. Martínez. Metodologías basadas en softcomputing para el diseño de técnicas de clasificación automática. In *CAEPIA 2011. Doctoral Consortium*, pages 1–4, 2011. 112

[33] J. M. Cadenas, M. C. Garrido, and R. Martínez. Generating optimized fuzzy partitions to classification and considerations to management imprecise data. In *Studies in Computational Intelligence*, pages 151–165. Springer, 2012. 97, 120, 196

[34] J. M. Cadenas, M. C. Garrido, and R. Martínez. A tool to manage low quality datasets. In *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, pages 1–8. IEEE, 2012. 204, 245

[35] J. M. Cadenas, M. C. Garrido, and R. Martínez. Towards an approach to select features from low quality datasets. In *International Conference on Fuzzy Computation Theory and Applications, in proceedings (FCTA 2012), Best Paper awards 2012*, pages 357–366, 2012. 159, 196

[36] J. M. Cadenas, M. C. Garrido, and R. Martínez. Feature subset selection filter-wrapper based on low quality data. *Expert Systems with Applications*, 40:1–10, 2013. doi:10.1016/j.eswa.2013.05.051. 159, 196

[37] J. M. Cadenas, M. C. Garrido, and R. Martínez. Generating fuzzy partitions from nominal and numerical attributes with imprecise values. In *Studies in Computational Intelligence*, pages 167–182. Springer, 2013. 128, 196

[38] J. M. Cadenas, M. C. Garrido, and R. Martínez. Improving a fuzzy discretization process by bagging. In *International Conference on Fuzzy Computation Theory and Application FCTA 2013*, pages 201–213, 2013. 131, 177, 196, 231

[39] J. M. Cadenas, M. C. Garrido, and R. Martínez. Nip - an imperfection processor to data mining datasets. *International Journal of Computational Intelligence Systems*, 6(sup1):3–17, 2013. 98, 167, 190, 204, 206, 245

[40] J. M. Cadenas, M. C. Garrido, and R. Martínez. Fuzzy discretization process from small datasets. In *Studies in Computational Intelligence*, page Pendiente de Publicación. Springer, 2014. 131, 196

[41] J. M. Cadenas, M. C. Garrido, R. Martínez, and P. P. Bonissone. Towards the learning from low quality data in a fuzzy random forest ensemble. In *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, pages 2897–2904. IEEE, 2011. 94, 112

[42] J. M. Cadenas, M. C. Garrido, R. Martínez, and P. P. Bonissone. Extending information processing in a fuzzy random forest ensemble. *Soft Computing*, 16(5):845–861, 2012. 94, 112, 150, 237

[43] J. M. Cadenas, M. C. Garrido, R. Martínez, and P. P. Bonissone. Ofp_class: a hybrid method to generate optimized fuzzy partitions for classification. *Soft Computing*, 16:667–682, 2012. 33, 97, 120, 152, 161, 196, 214, 231

[44] J. M. Cadenas, M. C. Garrido, R. Martínez, and R. A. Díaz-Valladares. *FRF Software. Registro de la propiedad intelectual*, Nº de asiento registral: 08 / 2014 / 0198, 3 julio 2014. 94, 113

[45] J. M. Cadenas, M. C. Garrido, R. Martínez, and A. Martínez. Regla k_m-vecinos más cercanos en bases de datos de baja calidad. In *Actas del VI Simposio de Teoría y Aplicaciones de Minería de Datos, (TAMIDA2013)*, pages 1303–1312, 2013. 113, 184, 197

[46] J. M. Cadenas, M. C. Garrido, R. Martínez, and A. Muñoz Ledesma. Towards an approach to select features from low quality datasets. In *VII Congreso Español sobre Tecnologías y Lógica Fuzzy Modelos de optimización y Soft Computing, (ESTYLF2014)*, pages 235–240, 2014. 131, 196

[47] J. M. Cadenas, M. C. Garrido, R. Martínez, D. Pelta, and P. P. Bonissone. Fuzzy discretization process from small datasets. In *Studies in Computational Intelligence*, page Pendiente de Publicación. Springer, 2014. 170, 197

[48] J. M. Cadenas, M. C. Garrido, R. Martínez, D. A. Pelta, and P. P. Bonissone. Using a fuzzy decision tree ensemble for tumor classification from gene expression data. In *International Conference on Fuzzy Computation Theory and Applications, in proceedings FCTA 2013*, pages 320–331, Vilamoura, Portugal, 2013. 170, 196

[49] J. M. Cadenas, M. C. Garrido, R. Martínez, and E. Serrano. *NIP imperfection procesor. Registro de la propiedad intelectual*, Nº de asiento registral: 08 / 2012 / 700, 18 julio 2012. 204, 245

[50] E. Cantu-Paz and C. Kamath. *Data Mining: A heuristic approach*, chapter On the use of evolutionary algorithms in data mining, pages 48–71. Ideal Group Publishing, 2001. 122

[51] J. Casillas, O. Cordón, M. J. Del Jesus, and F. Herrera. Genetic feature selection in a fuzzy rule-based classification system learning process for high-dimensional problems. *Information Sciences*, 136(1):135–157, 2001. 45

[52] J. Catlett. On changing continuous attributes into ordered discrete attributes. In *Machine learning—EWSL-91*, pages 164–178. Springer, 1991. 36

[53] C. C. Chan, C. Batur, and A. Srinivasan. Determination of quantization intervals in rule based model for dynamic systems. In *Systems, Man, and Cybernetics, 1991.'Decision Aiding for Complex Systems, Conference Proceedings., 1991 IEEE International Conference on*, pages 1719–1723. IEEE, 1991. 32

[54] C. Y. Chen, Z. G. Li, S. Y. Qiao, and S. P. Wen. Study on discretization in rough set based on genetic algorithm. In *Machine Learning and Cybernetics, 2003 International Conference on*, volume 3, pages 1430–1434. IEEE, 2003. 32

[55] J. Y. Ching, A. K. Wong, and K. C. Chan. Class-dependent discretization for inductive learning from continuous and mixed-mode data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(7):641–651, 1995. 31, 32

[56] B. S. Chlebus and S. H. Nguyen. On finding optimal discretizations for two attributes. In *Rough Sets and Current Trends in Computing*, pages 537–544. Springer, 1998. 28

[57] M. R. Chmielewski and J. W. Grzymala-Busse. Global discretization of continuous attributes as preprocessing for machine learning. *International journal of approximate reasoning*, 15(4):319–331, 1996. 32, 33

[58] Y. S. Choi and B. R. Moon. Feature selection in genetic fuzzy discretization for the pattern classification problems. *IEICE transactions on information and systems*, 90(7):1047–1054, 2007. 35, 137, 214, 231

[59] E. Cox. *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*. Morgan Kaufmann Publishers, 2005. 122

[60] A. De Luca and T. Termini. A definition of non-probabilistic entropy in setting of fuzzy set theory. *Information Control*, 20:301–312, 1971. 188, 189

[61] K. Deb, A. Pratap, S. Agarwal, and T. A. M. T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002. 46, 62

[62] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977. 48

[63] J. Demšar, B. Zupan, G. Leban, and T. Curk. *Orange: From experimental machine learning to interactive data mining*. Springer, 2004. 206

[64] J. Derrac, S. García, and F. Herrera. Ifs-coco: Instance and feature selection based on cooperative coevolution with nearest neighbor rule. *Pattern Recognition*, 43(6):2082–2105, 2010. 45

[65] R. A. Díaz-Uriarte and S. A. De Andres. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3, 2006. 43, 149, 166, 167, 168, 170

[66] E. Diday, M. Noirhomme-Fraiture, et al. *Symbolic data analysis and the SODAS software*. Wiley Online Library, 2008. 204

[67] Y. Ding and J. S. Simonoff. An investigation of missing data methods for classification trees applied to binary response data. *The Journal of Machine Learning Research*, 11:131–170, 2010. 50

[68] D. Dubois and D. Guyonnet. Risk-informed decision-making in the presence of epistemic uncertainty. *International Journal of General Systems*, 40(02):145–167, 2011. 20

[69] Didier Dubois and Henri Prade. *Possibility theory:an approach to computerized processing of uncertainty*. Springer, 1988. 19

[70] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, 2012. 45, 60, 107, 109, 184

[71] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, 2012. 56, 59

[72] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*, volume 57. CRC press, 1994. 34

[73] U. M. Fayyad. *Advances in Knowledge Discovery and Data Mining*. AAAI Press Series. AAAI Press, 1996. 9

[74] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the the Thirteenth International Joint Conference on Artificial Intelligence (Chambery, France, 1993)*, pages 1022–1027, 1993. 30, 31, 32, 33

[75] S. Ferrandiz and M. Boullé. Multivariate discretization by recursive supervised bipartition of graph. In *Machine learning and data mining in pattern recognition*, pages 253–264. Springer, 2005. 30

[76] E. W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965. 231

[77] A. Frank and A. Asuncion. Uci machine learning repository, university of california, irvine, school of information and computer sciences, 2007. *URL:¡http://www. ics. uci. edu/mlearn/MLRepository. html*, 2010. 97, 111, 117, 137, 139, 149, 166, 189, 194, 220, 237

[78] S. Garcia, J. Derrac, J. R. Cano, and F. Herrera. Prototype selection for nearest neighbor classification: taxonomy and empirical study. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):417–435, 2012. 108, 187

[79] S. García, A. Fernández, J. Luengo, and F. Herrera. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Computing*, 13(10):959–977, 2009. 96, 99, 140, 150, 169

[80] S. Garcia, J. Luengo, J. A. Sáez, V. López, and F. Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *Knowledge and Data Engineering, IEEE Transactions on*, 25(4):734–750, 2013. 27, 28, 29, 31, 32, 36

[81] M. C. Garrido, J. M. Cadenas, and P. P. Bonissone. A classification and regression technique to handle heterogeneous and imperfect information. *Soft Computing*, 14(11):1165–1185, 2010. 52, 61

[82] R. Genuer, J. M. Poggi, and C. Tuleau-Malot. Variable selection using random forests. *Pattern Recognition Letters*, 31(14):2225–2236, 2010. 43, 168, 170, 177, 179

[83] B. Ghattas and A. B. Ishak. Sélection de variables pour la classification binaire en grande dimension: comparaisons et application aux données de biopuces. *Journal de la société française de statistique*, 149(3):43–66, 2008. 168

[84] I. A. Gheyas and L. S. Smith. A neural network-based framework for the reconstruction of incomplete data sets. *Neurocomputing*, 73(16):3039–3065, 2010. 47, 50

[85] S. Ghorai, A. Mukherjee, and P. K. Dutta. Gene expression data classification by vvrkfa. *Procedia Technology*, 4:330–335, 2012. 166

[86] S. Ghosh. Statistical analysis with missing data. *Technometrics*, 30(4):455–455, 1988. 232

[87] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*, volume 412. Addison-wesley Reading Menlo Park, 1989. 35

[88] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999. 166

[89] D. E. Gustafson and W. C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Decision and Control including the 17th Symposium on Adaptive Processes, 1978 IEEE Conference on*, volume 17, pages 761–766. IEEE, 1978. 36

[90] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002. 42

[91] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009. 139, 205

[92] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006. 14, 25, 38, 53, 54, 58

[93] D. J. Hand, H. Mannila, and P. Smyth. *Principles of data mining*. MIT press, 2001. 53

[94] P. E. Hart. The condensed nearest neighbor rule. *Information Theory, IEEE Transactions on*, 14(3):515–516, 1968. 187

[95] Q. He, Z. Xie, Q. Hu, and C. Wu. Neighborhood based sample and feature selection for svm classification learning. *Neurocomputing*, 74(10):1585–1594, 2011. 45

[96] R. Hecht-Nielsen. *Neurocomputing*. New Horizons in Technology Series. Addison-Wesley Publishing Company, 1990. 59

[97] J. Hernández-Orallo, M. J. Ramírez-Quintana, , and C. Ferri-Ramírez. *Introducción a la Minería de Datos*. Pearson Prentice Hall, 2004. 14, 25, 26, 38, 46, 53, 56, 60

[98] R. Hickey. Noise modeling and evaluating learning from examples. *Artificial Intelligence*, 82(1–2):157–179, 1996. 22

[99] K. M. Ho and P. D. Scott. Zeta: A global method for discretization of cotitinuous variables. In *Proceedings of 3rd International Conference of Knowledge Discovery and Data Mining (KDD97). Newport Beach, CA*, pages 191–194, 1997. 31, 32

[100] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11(1):63–90, 1993. 31

[101] R. Ihaka and R. Gentleman. R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314, 1996. 97, 140, 151, 169

[102] A. K. Jain. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:4–37, 2000. 131

[103] A. K. Jain, R. P. Duin, and J. Mao. Statistical pattern recognition: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):4–37, 2000. 39

[104] C. Z. Janikov. Fuzzy partitioning with fid 3.1. In *Proceedings of 18th International Conference of the North American Fuzzy Information Processing Society*, pages 67–471, New York, USA, 1999. 33, 37, 138, 142

[105] C. Z. Janikow. Exemplar learning in fuzzy decision trees. In *Fuzzy Systems, 1996., Proceedings of the Fifth IEEE International Conference on*, volume 2, pages 1500–1505. IEEE, 1996. 58, 61

[106] C. Z. Janikow. Fuzzy decision trees: issues and methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 28(1):1–14, 1998. 52, 58, 61, 69, 70, 78

[107] R. Jin, Y. Breitbart, and C. Muoh. Data discretization unification. *Knowledge and Information Systems*, 19(1):1–29, 2009. 32

[108] M. M. Kabir, M. Shahjahan, and K. Murase. A new hybrid ant colony optimization algorithm for feature selection. *Expert Systems with Applications*, 39(3):3747–3763, 2012. 44

[109] H. R. Kanan, K. Faez, and S. M. Taheri. Feature selection using ant colony optimization (aco): a new method and comparative study in the application of face recognition system. In *Advances in Data Mining. Theoretical Aspects and Applications*, pages 63–76. Springer, 2007. 44

[110] M. A. Kbir, H. Benkirane, K. Maalmi, and R. Benslimane. Hierarchical fuzzy partition for pattern classification with fuzzy if-then rules. *Pattern Recognition Letters*, 21(6):503–509, 2000. 34

[111] L. Ke, Z. Feng, and Z. Ren. An efficient ant colony optimization approach to attribute reduction in rough set theory. *Pattern Recognition Letters*, 29(9):1351–1357, 2008. 44

[112] R. Kerber. Chimerge: Discretization of numeric attributes. In *Proceedings of the tenth national conference on Artificial intelligence*, pages 123–128. Aaai Press, 1992. 31, 32, 33

[113] S. S. Khan and A. Ahmad. Cluster center initialization algorithm for¡i¿k¡/i¿-means clustering. *Pattern recognition letters*, 25(11):1293–1302, 2004. 36

[114] K. Kianmehr, M. Alshalalfa, and R. Alhajj. Fuzzy clustering-based discretization for gene expression classification. *Knowledge and Information Systems*, 24:441–465, 2010. 35

[115] K. Kira and L. A. Rendell. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256. Morgan Kaufmann Publishers Inc., 1992. 45

[116] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997. 40

[117] S. Kotsiantis and D. Kanellopoulos. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006. 27, 28

[118] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006. 25, 26, 38

[119] T. Krishnan and G. J. McLachlan. The em algorithm and extensions. *Wiley*, 1:997, 1997. 58, 60

[120] Jussi Kujala and Tapio Elomaa. Improved algorithms for univariate discretization of continuous features. In *Knowledge Discovery in Databases: PKDD 2007*, pages 188–199. Springer, 2007. 30

[121] L. A Kurgan and K. J. Cios. Caim discretization algorithm. *Knowledge and Data Engineering, IEEE Transactions on*, 16(2):145–153, 2004. 32

[122] H. Langseth, T. D. Nielsen, R. Rumí, and A. Salmerón. Inference in hybrid bayesian networks. *Reliability Engineering & System Safety*, 94(10):1499–1509, 2009. 21, 60

[123] D. T. Larose. *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014. 53

[124] C. P. Lee and Y. Leu. A novel hybrid feature selection method for microarray data analysis. *Applied Soft Computing*, 11(1):208–213, 2011. 42

[125] C. Li. A combination scheme for fuzzy partitions based on fuzzy majority voting rule. In *Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC'09. International Conference on*, volume 2, pages 675–678. IEEE, 2009. 36, 137, 138

[126] C. Li, Y. Wang, and H. Dai. A combination scheme for fuzzy partitions based on fuzzy weighted majority voting rule. In *Digital Image Processing, 2009 International Conference on*, pages 3–7. IEEE, 2009. 33, 36, 137, 138

[127] X. Li, D. Ruan, and A. J. Van der Wal. Discussion on soft computing at flins'96. *International Journal of Intelligent Systems*, 13(2-3):287–300, 1998. 11

[128] S. H. Liao, P. H. Chu, and P. Y. Hsiao. Data mining techniques and applications–a decade review from 2000 to 2011. *Expert Systems with Applications*, 39(12):11303–11311, 2012. 53

[129] R. J. Little and D. B. Rubin. Statistical analysis with missing data. *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics Section (EUA).*, 1987. 21, 46, 48

[130] H. Liu, F. Hussain, C. L. Tan, and M. Dash. Discretization: An enabling technique. *Data mining and knowledge discovery*, 6(4):393–423, 2002. 27, 28, 29, 31

[131] H. Liu and H. Motoda. *Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*. Chapman & Hall/CRC, 2007. 40

[132] H. Liu and R. Setiono. Feature selection via discretization. *Knowledge and Data Engineering, IEEE Transactions on*, 9(4):642–645, 1997. 31

[133] L. Liu, A. K. Wong, and Y. Wang. A global optimal algorithm for class-dependent discretization of continuous data. *Intelligent Data Analysis*, 8(2):151–170, 2004. 31

[134] Y. Liu and Y. F. Zheng. Fs_sfs: A novel feature selection method for support vector machines. *Pattern recognition*, 39(7):1333–1345, 2006. 38, 41

[135] X. Llorá. E2k: evolution to knowledge. *ACM SIGEVOlution*, 1(3):10–17, 2006. 206

[136] C. P. López. *Minería de datos: técnicas y herramientas*. Editorial Paraninfo, 2007. 14, 25, 53

[137] J. Luengo, S. García, and F. Herrera. A study on the use of imputation methods for experimentation with radial basis function network classifiers handling missing attribute values: The good synergy between rbfns and eventcovering method. *Neural Networks*, 23(3):406–418, 2010. 47, 219

[138] J. Luengo, S. García, and F. Herrera. On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowledge and information systems*, 32(1):77–108, 2012. 47, 48, 50, 51

[139] J. Luengo, J. A. Sáez, and F. Herrera. Missing data imputation for fuzzy rule-based classification systems. *Soft Computing*, 16(5):863–881, 2012. 47

[140] P. Luukka. Feature selection using fuzzy entropy measures with similarity classifier. *Expert Systems with Applications*, 38(4):4600–4607, 2011. 43

[141] F. Marcelloni. Feature selection based on a modified fuzzy c-means algorithm with supervision. *Information Sciences*, 151:201–226, 2003. 42

[142] R. Martinez, J. M. Cadenas, and M. C. Garrido. The experimenter environment of the nip imperfection processor. In *Fuzzy Systems (FUZZ), 2014 IEEE International Conference on*, pages 2141–2148. IEEE, 2014. 204, 225, 233, 245

[143] R. Martínez, J. M. Cadenas, M. C. Garrido, and A. Martínez. Imputing missing values from low quality data by nip tool. In *IEEE International Conference on Fuzzy Systems (FUZZIEEE2013)*, pages 1–8. IEEE, 2013. 184, 187, 197, 207, 219, 235, 245

[144] I. MathWorks. Simulink design optimization getting started guide, 2011. 204

[145] S. Mehta, S. Parthasarathy, and H. Yang. Toward unsupervised correlation preserving discretization. *Knowledge and Data Engineering, IEEE Transactions on*, 17(9):1174–1185, 2005. 30

[146] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940. ACM, 2006. 205

[147] B. G. Mirkin and G. A. Satarov. Method of fuzzy additive types for analysis of multidimensional data: I, ii. *Automation and Remote Control*, 51(5–6):683–688,817–821, 1990. 35

[148] S. Mitra and S. K. Pal. Fuzzy multi-layer perceptron, inferencing and rule generation. *Neural Networks, IEEE Transactions on*, 6(1):51–63, 1995. 59, 60

[149] S. Mitra and S. K. Pal. Fuzzy sets in pattern recognition and machine intelligence. *Fuzzy Sets and systems*, 156(3):381–386, 2005. 12, 13, 14

[150] S. Mitra, S. K. Pal, and P. Mitra. Data mining in soft computing framework: A survey. *IEEE transactions on neural networks*, 13(1):3–14, 2002. 9, 53, 57

[151] Su. Mitra and T. Acharya. *Data mining: multimedia, soft computing, and bioinformatics*. John Wiley & Sons, 2005. 53

[152] D. Mladenić. *Feature selection for dimensionality reduction*. Springer, 2006. 41

[153] K. Morik and M. Scholz. The miningmart approach to knowledge discovery in databases. In *Intelligent technologies for information analysis*, pages 47–65. Springer, 2004. 205

[154] A. J. Myles and S. D. Brown. Induction of decision trees using fuzzy partitions. *Journal of chemometrics*, 17(10):531–536, 2003. 34, 69, 137, 138

[155] S. Nascimento, B. Mirkin, and F. Moura-Pires. A fuzzy clustering model of data and fuzzy c-means. In *Fuzzy Systems, 2000. FUZZ IEEE 2000. The Ninth IEEE International Conference on*, volume 1, pages 302–307. IEEE, 2000. 35

[156] D. Nitsch, J. P. Gonzalves, F. Ojeda, B. de Moor, and Y. Moreau. Candidate gene prioritization by network analysis of differential expression using machine learning approaches. *BMC Bioinformatics*, 11:460, 2010. 166

[157] C. Olaru and L. Wehenkel. A complete fuzzy decision tree technique. *Fuzzy sets and systems*, 138(2):221–254, 2003. 69

[158] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999. 85

[159] A. M. Palacios, M. J. Gacto, and J. Alcalá-Fdez. Mining fuzzy association rules from low-quality data. *Soft computing*, 16(5):883–901, 2012. 62, 220

[160] A. M. Palacios, L. Sánchez, and I. Couso. Extending a simple genetic cooperative-competitive learning fuzzy classifier to low quality datasets. *Evolutionary Intelligence*, 2:73–84, 2009. xxiii, xxv, xxvi, 61, 62, 96, 97, 100, 101, 102, 103, 104, 145, 146, 147, 155

[161] A. M. Palacios, L. Sánchez, and I. Couso. Diagnosis of dyslexia with low quality data with genetic fuzzy systems. *Int. J. Approx. Reasoning*, 51:993–1009, 2010. xxiii, xxvi, 61, 62, 96, 97, 100, 103, 104, 105, 145, 146, 147, 148, 155

[162] A. M. Palacios, L. Sánchez, and I. Couso. Linguistic cost-sensitive learning of genetic fuzzy classifiers for imprecise data. *International Journal of Approximate Reasoning*, 52(6):841–862, 2011. 62, 101, 102

[163] A. M. Palacios, L. Sanchez, and I. Couso. Boosting of fuzzy rules with low quality data. *Multiple-Valued Logic and Soft Computing*, 19(5-6):591–619, 2012. 61, 63

[164] W. Pedrycz and G. Vukovich. Feature analysis through information granulation and fuzzy sets. *Pattern Recognition*, 35(4):825–834, 2002. 42

[165] Y. Peng and P. A. Flach. Soft discretization to enhance the continuous decision tree induction. *Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, 1:109–118, 2001. 34

[166] P. Y. Piñero, L. Arco, M. M. García, and L. Acevedo. Algorítmos genéticos en la construcción de funciones de pertenencia borrosas. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 7(18):25–36, 2003. 35

[167] D. Pyle. *Data preparation for data mining*, volume 1. Morgan Kaufmann, 1999. 46

[168] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986. 30, 58, 69

[169] J. R. Quinlan. *C4.5: programs for machine learning*, volume 1. Morgan kaufmann, 1993. 27, 30, 31, 32, 50, 60, 61, 69

[170] T. Qureshi and D. A. Zighed. A soft discretization technique for fuzzy decision trees using resampling. *Intelligent Data Engineering and Automated Learning - IDEAL 2009, Lecture Notes in Computer Science*, "5788:586–593, 2009. 34

[171] R. Rakotomalala. Tanagra: a free software for research and academic purposes. In *Proceedings of EGC*, pages 697–702, 2005. 206

[172] S. J. Redmond and C. Heneghan. A method for initialising the k-means clustering algorithm using kd-trees. *Pattern recognition letters*, 28(8):965–973, 2007. 36

[173] J. C. Riquelme, R. Ruiz, and K. Gilbert. Mineria de datos: Conceptos y tendencias. *Revista Iberoamericana de Inteligencia Artificial*, 10(29):11–18, 2006. 14, 25, 53

[174] A. Ruiz, P. E. López-de Teruel, and M. C. Garrido. Probabilistic inference from arbitrary uncertainty using mixtures of factorized generalized gaussians. *J. Artificial Intell. Res.(JAIR)*, 9:167–217, 1998. 60, 61

[175] J. Rushing, R. Ramachandran, U. Nair, S. Graves, R. Welch, and H. Lin. Adam: a data mining toolkit for scientists and engineers. *Computers & Geosciences*, 31(5):607–618, 2005. 206

[176] S. Saha and S. Bandyopadhyay. A fuzzy genetic clustering technique using a new symmetry based distance for automatic evolution of clusters. In *Computing: Theory and Applications, 2007. ICCTA'07. International Conference on*, pages 309–314. IEEE, 2007. 36

[177] L. Sánchez and I. Couso. Obtaining fuzzy rules from interval-censored data with genetic algorithms and a random sets-based semantic of the linguistic labels. *Soft Computing*, 15(10):1945–1957, 2011. 61, 62

[178] L. Sánchez, I. Couso, and J. Casillas. Modeling vague data with genetic fuzzy systems under a combination of crisp and imprecise criteria. In *Computational Intelligence in Multicriteria Decision Making, IEEE Symposium on*, pages 30–37. IEEE, 2007. 38

[179] L. Sánchez, I. Couso, and J. Casillas. Genetic learning of fuzzy rules based on low quality data. *Fuzzy Sets and Systems*, 160(17):2524–2552, 2009. 61

[180] L. Sánchez, M. R. Suárez, J. R. Villar, and I. Couso. Mutual information-based feature selection and partition design in fuzzy rule-based classifiers from vague data. *International Journal of Approximate Reasoning*, 49(3):607–622, 2008. 38, 46, 52

[181] T. Schneider. Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values. *Journal of Climate*, 14(5):853–871, 2001. 50

[182] K. Shehzad. Edisc: a class-tailored discretization technique for rule-based classification. *Knowledge and Data Engineering, IEEE Transactions on*, 24(8):1435–1447, 2012. 30, 32

[183] D. Song, C. H. Ek, K. Huebner, and D. Kragic. Multivariate discretization for bayesian network structure learning in robot grasping. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1944–1950. IEEE, 2011. 30

[184] D. F. Specht. A general regression neural network. *Neural Networks, IEEE Transactions on*, 2(6):568–576, 1991. 50

[185] D. J. Stekhoven and P. Bühlmann. Missforest-non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012. 51

[186] M. R. Suarez, J. R. Villar, and J. Grande. A feature selection method using a fuzzy mutual information measure. *International Journal of Reasoning-based Intelligent Systems*, 2(2):133–141, 2010. 46

[187] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001. 51

[188] C. J. Tsai, C. I. Lee, and W. P. Yang. A discretization algorithm based on class-attribute contingency coefficient. *Information Sciences*, 178(3):714–731, 2008. 32

[189] M. Umano, H. Okamoto, I. Hatono, H. Tamura, F. Kawachi, S. Umedzu, and J. Kinoshita. Fuzzy decision trees by fuzzy id3 algorithm and its application to diagnosis systems. In *Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on*, pages 2113–2118 vol.3, 1994. 69, 70

[190] D. Ventura and T. R. Martinez. Brace: A paradigm for the discretization of continuously valued data. In *Proceedings of the Seventh Florida Artificial Intelligence Research Symposium*, pages 117–21, 1994. 32

[191] J. L. Verdegay. Una revisión de las metodologías que integran la soft computing. In *Actas del Simposio sobre Lógica Fuzzy y Soft Computing*, pages 151–156, 2005. 13

[192] J. L. Verdegay, R. R. Yager, and P. P. Bonissone. On heuristics as a fundamental constituent of soft computing. *Fuzzy Sets and Systems*, 159(7):846–855, 2008. 11, 12, 16

[193] S. M. Vieira, J. Sousa, and U. Kaymak. Fuzzy criteria for feature selection. *Fuzzy Sets and Systems*, 189(1):1–18, 2012. 39, 40, 44

[194] S. M. Vieira, J. Sousa, and T. A. Runkler. Ant colony optimization applied to feature selection in fuzzy classifiers. In *Foundations of Fuzzy Logic and Soft Computing*, pages 778–788. Springer, 2007. 44

[195] J. R. Villar, A. Berzosa, E. de la Cal, J. Sedano, and M. García-Tamargo. Multi-objective learning of white box models with low quality data. *Neurocomputing*, 75(1):219–225, 2012. 61, 62

[196] H. Wang and S. Wang. Mining incomplete survey data through classification. *Knowledge and information systems*, 24(2):221–233, 2010. 47

[197] X. Wang and E. E. Kerre. Reasonable properties for the ordering of fuzzy quantities (i-ii). *Fuzzy Sets and Systems*, 118(3):375–405, 2001. 130

[198] X. Wang, J. Yang, X. Teng, W. Xia, and R. Jensen. Feature selection based on rough sets and particle swarm optimization. *Pattern Recognition Letters*, 28(4):459–471, 2007. 45

[199] Y. Wang, C. Li, and Y. Zuo. A selection model for optimal fuzzy clustering algorithm and number of clusters based on competitive comprehensive fuzzy evaluation. *Fuzzy Systems, IEEE Transactions on*, 17(3):568–577, 2009. 36

[200] D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3):257–286, 2000. 187

[201] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005. 14, 25, 38, 58, 59, 60

[202] Ian H Witten, Eibe Frank, and A Mark. Hall. 2011. data mining: Practical machine learning tools and techniques, 2011. 53, 54, 58

[203] A. K. Wong and D. K. Chiu. Synthesizing statistical knowledge from incomplete mixed-mode data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 9(6):796–805, 1987. 30, 32

[204] K. L. Wu and M. S. Yang. Alternative c-means clustering algorithms. *Pattern recognition*, 35(10):2267–2278, 2002. 36

[205] X. Wu. A bayesian discretizer for real-valued attributes. *The Computer Journal*, 39(8):688–691, 1996. 32

[206] X. Wu and V. Kumar. *The top ten algorithms in data mining*. Champman & Hall/CRC Data Mining and Knowledge Discovery, 2010. 27

[207] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE transactions on Systems, Man and Cybernetics*, 18:183–190, 1988. 163

[208] R. R. Yager and L. A. Zadeh. *An introduction to fuzzy logic applications in intelligent systems*. Springer, 1992. 27

[209] Y. Yang, Z. Jia, C. Chang, X. Qin, T. Li, H. Wang, and J. Zhao. An efficient fuzzy kohonen clustering network algorithm. In *Fuzzy Systems and Knowledge Discovery, 2008. FSKD'08. Fifth International Conference on*, volume 1, pages 510–513. IEEE, 2008. 35

[210] Y. Yang and G. I. Webb. Discretization for naive-bayes learning: managing discretization bias and variance. *Machine learning*, 74(1):39–74, 2009. 27, 30

[211] Y. Yang, G. I. Webb, and X. Wu. Discretization methods. In *Data Mining and Knowledge Discovery Handbook*, pages 101–116. Springer, 2010. 27

[212] Y. Q. Yao, J. S. Mi, and Z. J. Li. Attribute reduction based on generalized fuzzy evidence theory in fuzzy decision systems. *Fuzzy Sets and Systems*, 170(1):64–75, 2011. 46

[213] D. Yu, Q. Hu, and C. Wu. Uncertainty measures for fuzzy relations and their applications. *Applied soft computing*, 7(3):1135–1143, 2007. 42

[214] Y. C. Yuan. Multiple imputation for missing data: Concepts and new development (version 9.0). *SAS Institute Inc, Rockville, MD*, 2010. 49

[215] L. A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965. 12

[216] L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning—i. *Information sciences*, 8(3):199–249, 1975. 27

[217] L. A. Zadeh. Fuzzy logic, neural networks, and soft computing. *Communications of the ACM*, 37(3):77–84, 1994. 10, 13, 14

[218] L. A. Zadeh. Soft computing and fuzzy logic. *Software, IEEE*, 11(6):48–56, 1994. 10, 11, 12, 13

[219] X. Zhu, X. Wu, and Y. Yang. Error detection and impact-sensitive instance ranking in noisy datasets. In *AAAI*, pages 378–384, 2004. 216