Máster en Estadística Aplicada
Trabajo de Fin de Máster

Técnicas de minería de datos y software disponible. Una evaluación comparativa. Aplicación a datos reales.

Diandra María Correa Merlo

UNIVERSIDAD DE GRANADA

Departamento de Estadística e Investigación Operativa

Universidad de Granada

Tutor

Prof. Dr. D. Juan Antonio Maldonado Jurado

Granada, junio de 2022

Máster en Estadística Aplicada
Trabajo de Fin de Máste

Técnicas de minería de datos y software disponible. Una evaluación comparativa. Aplicación a datos reales.

UNIVERSIDAD DE GRANADA

Declaro explícitamente que el trabajo presentado es original, entendido en el sentido de que no he utilizado fuentes sin citarlas debidamente.

Diandra María Correa Merlo

AUTOEVALUACIÓN

En el Trabajo de Fin de Máster he desarrollado el concepto de *minería de datos*, con el objetivo de estudiar su evolución y aplicar dicho concepto a partir del software *Weka*. Indicar la existencia de diversos softwares existentes que utilizan este concepto. En la comprensión de estos conocimientos, he tenido algún problema con algunas partes de teoría. Para la resolución de estos problemas, he contado con la ayuda de mi tutor y con apuntes de otras asignaturas, para la comprensión de ideas. Además, gracias a las diferentes fuentes de información, puede llegar a estructurar la teoría. Con respecto a lo explicado en el trabajo, me parecen muy útiles a nivel profesional, puesto que actualmente se trabaja con grandes volúmenes de datos.

En la parte práctica, he aplicado los algoritmos de clasificación más usados en un archivo de datos denominado *winequalitywhite.arff*. Gracias a este apartado realizamos una introducción al software *Weka* e analizamos los resultados que se obtienen en cada caso.

En conclusión, estoy satisfecha con el desarrollo del trabajo y con el tema seleccionado, al haber aprendido y adquirido conocimientos, sobre algunos métodos de clasificación. Indicar que el tema es muy interesante, útil, y me parece un buen inicio para profundizar más sobre la *minería de datos*.

RESUMEN

La *minería de datos* es una disciplina que estudia el análisis de grandes volúmenes de datos para obtener conocimiento a partir de ellos, surgiendo de la mezcla de conceptos y disciplinas como la estadística, las bases de datos, la visualización, el aprendizaje automático y otros sistemas de apoyo. Se trata de la etapa más importante del proceso **KDD** *(Knowledge Discovery in Databases)*, en la cual se pueden aplicar diferentes técnicas para describir el conjunto de datos o predecir resultados mediante el uso de algoritmos de aprendizaje automático. Estas técnicas o métodos se dividen en tareas predictivas, conocidas también como métodos de aprendizaje supervisado, y tareas descriptivas, conocidas también como métodos de aprendizaje no supervisado, que se suelen clasificar dependiendo del tipo de modelo que sean capaces de generar.

Este concepto se puede aplicar de forma práctica en diversos softwares, ayudando a las corporaciones y otros usuarios a extraer datos útiles de una gran cantidad de datos sin procesar, para descubrir correlaciones, tendencias y anomalías. Hay una variedad de sistemas disponibles tanto de licencia gratuita como comercial.

En este Trabajo Fin de Máster se estudia uno de los software más sencillos de utilizar con licencia gratuita, aplicando algunos algoritmos de clasificación sobre un archivo de datos. Además se realiza una explicación exhaustiva del software.

La estructura de la memoria es la siguiente:

- En el Capítulo 1 encontramos toda la parte de teoría, donde se desarrolla el origen, la definición, las fases, las técnicas y los softwares que aplican el concepto de la minería de datos.

- En el Capítulo 2 explicamos uno de los sistemas con licencia gratuita que aplica el concepto de la minería de datos, denominado *Weka*. Se divide en varios subapartados donde se explica el sistema, el tipo de archivo de datos a utilizar, su instalación paso por paso y su interfaz. Indicar que la interfaz de Weka se divide en cinco posibles aplicaciones

para acceder a las diferentes funcionalidades del programa, éstas son *Simple CLI*, *Explorer*, *Experimenter*, *KnowledgeFlow* y *Workblench*. Este sistema destaca por su sencillez, siendo una buena introducción para el manejo de grandes volúmenes de datos.

- En el Capítulo 3 se aplica un campo del aprendizaje automático muy usado para desarrollar diferentes técnicas o algoritmos que nos ayudarán a extraer información relevante de la que podemos dar uso dependiendo del objetivo buscado, siendo el de la clasificación. Existen ocho familias de clasificadores, pero los más utilizados son los bayesianos, los metaclasificadores, las reglas y los árboles de decisión. Dentro de cada familia observamos que los algoritmos más utilizados son *Naïve Bayes*, *Stacking*, *OneR* y *J48*, respectivamente. Seguidamente, estos algoritmos son los aplicados a un archivo de datos para cada modo de evaluación *(Use training set, Supplied test set, Cross-validation y Percentage split)*, con el objetivo de realizar una comparación entre modelos y poder estudiar la eficiencia de clasificación del modelo sobre una clase sin etiquetar.

SUMMARY

*Data mining*is a discipline that studies the analysis of large volumes of data in order to obtain knowledge from them, emerging from the mixture of concepts and disciplines such as statistics, databases, visualization, automatical learning and other support systems. This is the most important phase of the **KDD** *(Knowledge Discovery in Databases)* process, in which different techniques can be applied to describe the data set or predict results through the use of automatical learning algorithms. These techniques or methods are divided into predictive tasks, also known as supervised learning methods, and descriptive tasks, also known as unsupervised learning methods, which are usually classified depending on the type of model they are able to generate.

This concept can be practically applied in various software, helping corporations and other users to extract useful data from a large amount of raw data, to discover correlations, trends and anomalies. There are a variety of systems available whose license is free as well as commercial.

In this Master's Thesis, one of the simplest software to use with a free license is studied, applying some classification algorithms on a data file. In addition, an exhaustive explanation of the software is made. The memory structure is as follows:

- In Chapter 1 we find all the theoric part, where the origin, definition, phases, techniques and software that apply the concept of data mining are developed.
- In Chapter 2 we explained one of the free licensed systems that implements the concept of data mining, called *Weka*. It is divided into several subsections where the system is explained, the type of data file to use, its installation step by step and its interface. Must be indicated that the Weka interface is divided into five possible applications to access the different functionalities of the program, these are *Simple CLI*, *Explorer*, *Experimenter*, *KnowledgeFlow* and *Workblench*. This system stands out for its simplicity, being a good introduction to manage large volumes of data.

- In Chapter 3, a widely used field of automatical learning is applied to develop different techniques or algorithms that will help us extract relevant information that we can use depending on the objective searched, being this the classification one. There are eight families of classifiers, but the most widely used are the bayesian ones, metaclassifiers, the rules and the decision trees. Within each family we observe that the most used algorithms are *Naïve Bayes*, *Stacking*, *OneR* and *J48*, respectively. Next, these algorithms are applied to a data file for each evaluation mode *(Use training set, Supplied test set, Cross-validation and Percentage split)*, with the aim of making a comparison between models and being able to study the classification efficiency of the model about an unlabeled class.

Índice general

Capítulo 1

Minería de Datos

1.1. Evolución de la Minería de Datos

El origen de la **minería de datos** surge por otras técnicas y herramientas utilizadas en la década de 1960, cuando los estadísticos manejaban términos como *data fishing* (buscar o "pescar" en los datos), *data dredging* (dragado de datos) o *data archaeology* (recuperar datos informáticos codificados y/o encriptados), con la idea de encontrar correlaciones sin partir de una hipótesis previa en bases de datos con ruido.

A principios de los años 1980, Rakesh Agrawal, Gio Wiederhold, Robert Blum y Gregory Piatetsk y Shapiro, entre otros, empezaron a consolidar los términos de **data mining** *(minería de datos)* y **KDD** *(Knowledge Discovery in Databases)*, siendo la **minería de datos** la etapa más importante de dicho proceso. Apareciendo en la década de 1990 el término **data mining** con el que se conoce actualmente al proceso de obtención de conocimiento a partir de los datos por medio de su análisis.

Este término ha tenido mucho que ver con el cambio de concepción de los datos, unido a la gran cantidad de estos que se generan y almacenan continuamente en cualquier ámbito. Otro factor que también ha favorecido la consolidación de la *minería de datos* como disciplina, es el gran avance en los últimos tiempos en las prestaciones y la capacidad computacional.

Tradicionalmente, las técnicas de **minería de datos** se aplicaban sobre información contenida en almacenes de datos. No obstante, actualmente está cobrando una mayor

importancia la minería de datos desestructurados como es la información contenida en ficheros de texto (text mining), en Internet (web mining), etc. Además, han surgido otras necesidades de tipo operativo, como la integración de los resultados obtenidos en los sistemas de información en línea, con la exigencia, por tanto, de que los procesos funcionen prácticamente en tiempo real; por ejemplo, la alerta temprana frente a alarmas en una cadena de montaje, la detección instantánea del fraude en operaciones bancarias, un sistema de recomendación de productos en una tienda en línea, etc.

1.2. ¿Qué es la Minería de Datos?

La **minería de datos**, también conocida como descubrimiento de conocimienos en bases de datos o *knowledge discovery in databases* ***(KDD)***, es una disciplina que estudia el análisis de grandes volúmenes de datos *(datasets)* para obtener **conocimiento** a partir de ellos, surgiendo de la mezcla de conceptos y disciplinas como:

- **Estadística**: muchas de las técnicas que se aplican en la minería de datos son o tienen su raíz en la Estadística. Se podría decir que la Estadística es la madre de la minería de datos.

- **Bases de datos**: el proceso de KDD parte de datos que, habitualmente, se encuentran almacenados en bases de datos.

- **Visualización**: el objetivo final de la minería de datos es obtener conocimiento que sea útil. Para lograrlo, es un requisito fundamental que ese conocimiento pueda ser visualizado por los expertos de cada dominio. De ahí la importancia de las técnicas de visualización (diagramas, gráficos, resúmenes, etc).

- **Aprendizaje automático**: se encuentra profundamente ligado con la minería de datos, ya que ambos, de alguna manera, persiguen la obtención de modelos por medio de mecanismos automáticos.

- **Otras**: sistemas de apoyo a la decisión, recuperación de información, procesamiento de señales, etc.

En consecuencia, se trata de una nueva tecnología de manejo y análisis de información, que aprovecha la capacidad existente hoy día de procesamiento, almacenamiento y transmisión de datos a gran velocidad y bajo costo, con el objetivo de extraer información de un conjunto de datos y transformarla en una estructura comprensible para su uso posterior. Pudiendo decir que es una combinación de procesos como extracción de datos, limpieza de datos, selección de características, algoritmos y análisis de resultados.

La **minería de datos** combina distintas técnicas que otorgan efectos inesperados que se transforman en un valor añadido a la empresa, con unos resultados fáciles de entender, contribuyendo a la toma de decisiones tácticas y estratégicas para detectar la información clave. Además, abre nuevas oportunidades de negocios y ahorra costes a la empresa. Sin embargo, también existen pequeños incovenientes al usar la **minería de datos** como: la dificultad de recopilación de los datos, dependiendo de su tipo; y aunque cada vez menos, el requerimiento de una gran inversión al utilizar tecnologías para llevar a cabo la recopilación de datos, consumiendo muchos recursos que podrían suponer un coste elevado.

Con relación a otras técnicas podemos observar que la **minería de datos**, al basarse en analizar grandes volúmenes de datos y en descubrir patrones desconocidos, guarda cierta similitud con **Big Data**, con la diferencia de que este último analiza volúmenes de datos que superan la capacidad de los procesamientos informáticos habituales; y con **marchine learning** *(Inteligencia artificial)*, con la diferencia de que este se usa para reproducir patrones conocidos y hacer predicciones basadas en los patrones. De igual forma, existe una diferencia muy clara entre el **análisis de datos** y la **minería de datos**, la cual es que uno se utiliza para probar modelos e hipótesis en el conjunto de datos, mientras que el otro utiliza modelos estadísticos y de aprendizaje automático para descubrir patrones ocultos en un gran volumen de datos.

1.3. Fases de la Minería de Datos

El proceso de **minería de datos** implica una serie de pasos desde la recopilación de datos hasta la visualización para extraer información valiosa de grandes conjuntos de datos. Las técnicas de minería de datos se utilizan para generar descripciones y predicciones sobre un conjunto de datos. Los científicos de datos y estadísticos describen los datos a través de la observación de patrones, asociaciones y correlaciones. También clasifican y agrupan datos a través de métodos de clasificación y regresión e identifican valores atípicos.

El proceso de **data mining** *(minería de datos)* se puede descomponer en varias fases: (1) Preparación de datos, (2) Costrucción de modelos y data mining, (3) Evaluación de resultados e implementación del conocimiento y (4) Establecer los objetivos comerciales. El gráfico siguiente ilustra estas fases.

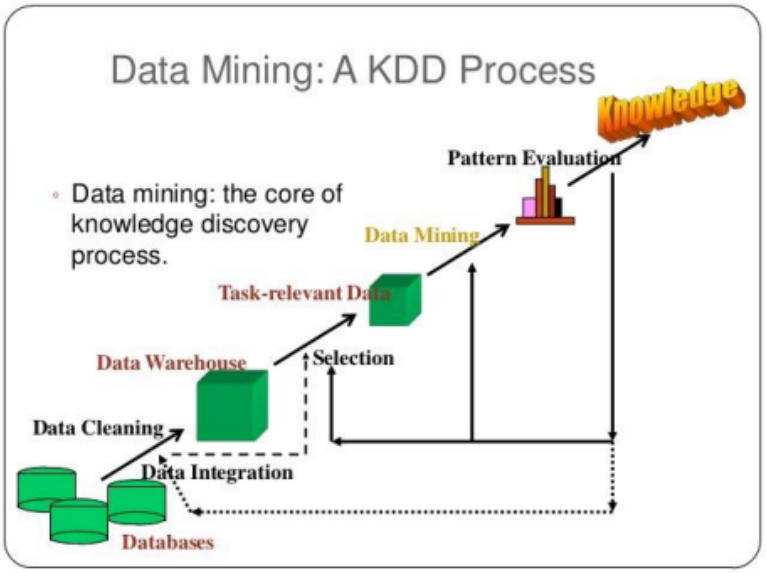


Figura 1.1: Fases del **data mining**

- **Preparación de datos:** una vez que se define el alcance del problema, es más fácil para los científicos de datos identificar qué conjunto de datos ayudará a responder

las preguntas pertinentes para el negocio. Una vez que recopilen los datos relevantes, los datos se limpiarán, eliminando cualquier ruido, como duplicados, valores perdidos y valores atípicos. Dependiendo del conjunto de datos, se puede tomar un paso adicional para reducir el número de dimensiones, ya que demasiadas características pueden ralentizar cualquier cálculo posterior. Los científicos de datos buscarán retener los predictores más importantes para garantizar una precisión óptima dentro de cualquier modelo.

- **Construcción de modelos y data mining:** dependiendo del tipo de análisis, los científicos de datos pueden investigar cualquier relación de datos interesante, como patrones secuenciales, reglas de asociación o correlaciones. Si bien los patrones de alta frecuencia tienen aplicaciones más amplias, a veces las desviaciones en los datos pueden ser más interesantes, destacando áreas de posible fraude.

- **Evaluación de resultados e implementación del conocimiento:** una vez agregados los datos, los resultados deben ser evaluados e interpretados. Al finalizar los resultados, deben ser válidos, novedosos, útiles y comprensibles. Cuando se cumple este criterio, las organizaciones pueden utilizar este conocimiento para implementar nuevas estrategias, logrando sus objetivos previstos.

- **Establecer los objetivos comerciales:** esta puede ser la parte más difícil del proceso de data mining y muchas organizaciones dedican muy poco tiempo a este importante paso. Los científicos de datos y las partes interesadas comerciales deben trabajar juntos para definir el problema comercial, lo que ayuda a informar las preguntas y los parámetros de datos para un proyecto determinado. Es posible que los analistas también necesiten realizar una investigación adicional para comprender el contexto empresarial de manera adecuada.

1.4. Técnicas de la Minería de Datos

En la etapa del **data mining** *(minería de datos)* se pueden aplicar diferentes técnicas para describir el conjunto de datos o para predecir resultados mediante el uso de algoritmos de aprendizaje automático. Estas técnicas o métodos se dividen en **tareas predictivas**, conocidas también como métodos de *aprendizaje supervisado*, y **tareas descriptivas**, conocidas también como métodos de *aprendizaje no supervisado* , que se suelen clasificar dependiendo del tipo de modelo que sean capaces de generar:

- **Tareas predictivas** *(aprendizaje supervisado)*. Son aquellas en las que se aprenden funciones, relaciones que asocian entradas con salidas, por lo que se ajustan a un conjunto de ejemplos de los que conocemos la relación entre la entrada y la salida deseada. Este hecho incluso llega a proporcionar una de las clasificaciones más habituales en el tipo de algoritmos que se desarrollan, así, dependiendo del tipo de salida, suele darse una subcategoría que se diferencian en:

  - *Clasificación*. Consiste en encontrar un modelo que, aplicado a un nuevo ejemplo sin clasificar, lo clasifique dentro de un conjunto predefinido de clases. Normalmente, el atributo a predecir es de tipo cualitativo, y recibe el nombre de atributo de clase.
  - *Análisis de Regresión*. Es similar al de clasificación, con la diferencia de que, en este caso, el atributo es de tipo cuantitativo.

- **Tareas descriptivas** *(aprendizaje no supervisado)*. Son aquellas en las que no estamos interesados en ajustar pares (entrada, salida), sino en aumentar el conocimiento estructural de los datos disponibles (y posibles datos futuros que provengan del mismo fenómeno). Entre las más importantes se encuentran:

  - *Agrupación o Clustering*. Pretende dividir una población heterogénea de objetos en grupos homogéneos, denominados clústeres, de forma que los objetos de

cada grupo sean muy similares entre sí. También se denomina segmentación o agrupamiento.

- *Reglas de asociación.* Pretende encontrar reglas que muestran la relación que existe entre los distintos atributos de los datos analizados, denominadas reglas de asociación.
- *Detección de valores atípicos.* Consiste en encontrar objetos que, dentro de un conjunto, manifiesten características significativamente diferentes a las del resto de los objetos del conjunto.

1.5. Software disponible para la Minería de Datos

El **software de minería de datos** ayuda a las corporaciones y otros usuarios a extraer datos útiles de una gran cantidad de datos sin procesar, para descubrir correlaciones, tendencias y anomalías. Recordemos que todos los sistemas de minería de datos procesan la información de manera diferente entre sí, y es precisamente eso lo que lo hace que sea tan complicado elegir uno en específico. Además, algunos de los principales enfoques utilizados para extraer datos son: el análisis de datos estadísticos, los algoritmos especializados, el aprendizaje automático, las estadísticas de bases de datos y la inteligencia artificial.

Hay una variedad de sistemas disponibles, y algunos de ellos tienen características más complejas. Sabiendo esto, mostraremos una lista con las mejores herramientas comerciales y gratuitas existentes.

1.5.1. Software de licencia comercial

- **IBM SPSS (Statistical Package for the Social Sciences):** es uno de los programas estadísticos más conocidos, teniendo en cuenta su capacidad para trabajar con grandes bases de datos y una sencilla interfaz para la mayoría de los análisis, compuesto por un conjunto de programas. Este software te permite generar una

gran variedad de algoritmos de minería de datos sin programación. También te ayuda en la detección de anomalías, redes bayesianas, CARMA, Cox y redes neuronales básicas que utilizan perceptores multicapa con aprendizaje retráctil.

- **Alteryx Analytics:** es una herramienta multidimensional que nos permite desde preparar nuestro conjunto de datos hasta obtener modelos predictivos pasando por la limpieza de éstos, la aplicación de análisis de diferentes orígenes y la capacidad de unir o mapearlos con una enorme facilidad. Se trata de una herramienta diseñada para analistas de datos y líderes empresariales, cuya principal ventaja está en la posibilidad de combinar sus diferentes productos para obtener una solución global que sea capaz de abarcar el proceso completo del procesamiento de los datos.

- **Xplenty:** es un conjunto de herramientas completo para crear canalizaciones de datos con capacidades de código bajo y sin código, que proporciona una plataforma que tiene funcionalidades para integrar, procesar y preparar datos para análisis. Tiene una interfaz intuitiva para implementar ETL (extraer, transformar, cargar) o una solución de replicación.

- **Sisence Analytics:** es el único software de creación de informes de inteligencia comercial que permite a cualquier usuario transformar fácilmente los datos en informes interactivos deslumbrantes. Además, simplifica cada paso del proceso de análisis, desde la preparación de datos hasta el descubrimiento de ideas. Su principal ventaja es la facilidad de visualización instantánea de información empresarial en entornos de datos complejos.

- **XLminer de Solver:** es un tipo de solucionador analítico que integra una gran cantidad de análisis productivos o prescriptivos. Sus servicios también incluyen pronósticos, optimización, minería de datos y simulación de varios problemas de datos en Excel. El software de minería de datos que incluye es muy adecuado para proteger el desarrollo de modelos a través de múltiples sistemas de bases de datos y

análisis, debido a que proporciona gráficos para mejorar la comprensión y análisis de los datos; clasifica los datos en varios grupos para representarlos como una variable categórica; admite la generación de reglas de asociación para todos los atributos de datos; y proporciona una oferta de series temporales con las técnicas exploratorias, incluyendo técnicas de suavizado y función de autocorrelación parcial.

- **Board:** es una plataforma de toma de decisiones que proporciona una solución continua para el soporte, control y gestión de los procesos clave de la empresa que garantiza un acceso unificado y coherente a la información, cubriendo todas las áreas de la empresa (comercial, logística, producción, control de gestión, finanzas, etc.) y cualquier industria, (retail, alimentación, manufacturing, legal, banca y seguros, química y muchas otras). Además, permite una visión completa y precisa de la información de toda la organización, completamente integrada en los procesos de negocio desde la planificación más estratégica hasta el dato operacional más detallado.

- **SAS Enterprise Miner:** es un software que agiliza el proceso de minería de datos para crear modelos predictivos y descriptivos basados en el análisis de grandes cantidades de datos. Se puede acceder a los datos desde archivos locales o desde conexiones de bases de datos remotas. Además, puede transformar y manipular datos mediante filtros y análisis estadísticos para extraer los datos deseados de grandes conjuntos de datos.

- **ODM (Oracle Data Mining):** es una de las componentes que proporciona una potente capacidad de minería de datos de última generación dentro de *Oracle Database*. Se puede utilizar para crear e implementar aplicaciones de minería de datos predictivas y descriptivas, agregar capacidades inteligentes a las aplicaciones existentes y generar consultas predictivas para la exploración de datos. Además, ofrece un conjunto integral de algoritmos en la base de datos para realizar una variedad de tareas de minería, como clasificación, regresión, detección de anomalías, extrac-

ción de características, agrupación y análisis de canasta de mercado. Los algoritmos pueden funcionar en datos de casos estándar, datos transaccionales, esquemas en estrella y texto y otras formas de datos no estructurados.

1.5.2. Software de licencia gratuita

- **RapidMiner (YALE, Yet Another Learning Environment):** ofrece un paquete de productos que permiten a los analistas de datos crear nuevos procesos de extracción de datos, configurar análisis predictivos y más. Permite el desarrollo de procesos de análisis de datos mediante el encadenamiento de operadores, a través de un entorno gráfico. Las aplicaciones móviles y los chatbots tienden a depender de esta plataforma de software para el aprendizaje automático, creación rápida de prototipos, desarrollo de aplicaciones, minería de texto y análisis predictivo, para mejorar la experiencia del cliente. La lista de productos que incluye son: *RapidMiner Studio*, *RapidMiner Server*, *RapidMiner Radoop* y *RapidMiner Streams*.

- **R Commander:** es un software de código abierto y es fácil de usar, sin importar que no tengas ninguna experiencia en programación. Se ejecuta en casi todos los sistemas operativos y puedes descargarle algoritmos súper avanzados para trabajar con grandes paquetes de información. Además, te permite manipular datos fácilmente, visualizarlos a través de gráficos interactivos y animados y realizar grandes análisis estadísticos de ellos.

- **SAS (Statistical Analysis System):** utilizando *SAS Rapid Predictive Modeler*, los usuarios son guiados automáticamente a través de un flujo de trabajo entre bastidores de preparación de datos y tareas de extracción de datos, lo que les permite generar sus propios modelos y obtener información. Se usa principalmente a nivel empresarial, y sin duda es una muy buena opción para las técnicas de reducción de dimensión de modelos de mercado predictivos así como también para crear visuali-

zaciones interactivas.

- **Python (Statistical Analysis System):** es un lenguaje de programación que a menudo se usa para crear sitios web y software, automatizar tareas y realizar análisis de datos. Además es un lenguaje de propósito general, lo que significa que puede usarse para crear una variedad de programas diferentes y no está especializado para ningún problema específico.

- **Orange Canvas:** es un software de aprendizaje automático y de procesos de manipulación de datos. Este es sin duda un gran ejemplo de lo que **Python** puede crear, siendo una de las mejores herramientas para hacer data mining. Además, consta de una serie de componentes desarrollados en **C++** que implementan algoritmos de minería de datos, así como operaciones de preprocesamiento y representación gráfica de datos.

- **KNIME (Konstanz Information Miner):** es una plataforma de data mining que ayuda a las pequeñas y grandes empresas a crear y administrar aplicaciones o servicios de ciencia de datos. Está desarrollado sobre la plataforma Eclipse, y programado en **Java**. Está concebido como una herramienta gráfica y dispone de una serie de nodos (que encapsulan distintos tipos de algoritmos) y flechas (que representan flujos de datos) que se despliegan y combinan de manera gráfica e interactiva.

- **Spark:** es un *framework (marco de referencia)* que ofrece una serie de plataformas interconectadas, estándares y sistemas con los que se llevan a cabo proyectos de **Big Data**. Al tratarse de código abierto, cualquier desarrollador puede usar el código de forma libre para implementar nuevas versiones destinadas a resolver problemas que vayan surgiendo. Además puede ser considerado como un sistema de computación en clúster de propósito general, distinguiéndose de otras herramientas por su velocidad y simplicidad, proporcionando *APIs* en lenguajes de programación como **Python**, **Java** y **R**.

- **Apache Mahout:** es un proyecto de *Apache Software Foundation*, destinado a producir algoritmos de aprendizaje automático escalables, extraer recomendaciones y relaciones de conjuntos de datos de una manera simplificada. Este software crece continuamente y cuenta con una extensa biblioteca en **JAVA**, cuyo rendimiento, al igual que su velocidad, es impresionante.

- **Weka (Waikato Environment for Knowledge Analysis):** es un entorno para experimentación de análisis de datos que permite aplicar, analizar y evaluar las técnicas más relevantes de análisis de datos, principalmente las provenientes del aprendizaje automático, sobre cualquier conjunto de datos del usuario (se requiere que los datos a analizar se almacenen en el formato *ARFF (Attribute-Relation File Format)*). Está constituido por una serie de paquetes de código abierto con diferentes técnicas de preprocesado, clasificación, agrupamiento, asociación, y visualización, así como facilidades para su aplicación y análisis de prestaciones cuando son aplicadas a los datos de entrada seleccionados. Además, se distribuye como software de libre distribución desarrollado en **JAVA**.

- **SNNS (Stuttgart Neural Network Simulator):** es un simulador de software para redes neuronales en estaciones de trabajo Unix desarrollado en el *Instituto de Sistemas de Alto Rendimiento Paralelos y Distribuidos (IPVR)* de la Universidad de Stuttgart, cuyo objetivo es crear un entorno de simulación eficiente y flexible para la investigación y aplicación de redes neuronales. Este simulador consta de dos componentes principales: núcleo del simulador escrito en C, que opera en las estructuras de datos de la red interna de las redes neuronales y realiza todas las operaciones de aprendizaje y recuperación; y interfaz gráfica de usuario bajo X11R4 o X11R5.

Capítulo 2

Software WEKA

2.1. Introducción a WEKA

Weka (Waikato Environment for Knowledge Analysis), es un entorno para la experimentación de análisis de datos que permite aplicar, analizar y evaluar las técnicas más relevantes de análisis de datos principalmente, procedentes del aprendizaje automático sobre cualquier conjunto de datos del usuario, originado en la Universidad de Waikato de Nueva Zelanda en 1993. Se trata de un software de libre distribución desarrollado en **Java** con licencia **GNU** *(General Public License)*, significando que es de libre distribución y difusión.

Para su utilización, es imprescindible que los datos a analizar se almacenen con un cierto formato, conocido como **ARFF** *(Attribute-Relation File Format)*. Está constituido por una serie de paquetes de código abierto con diferentes técnicas de preprocesado, clasificación, agrupamiento, asociación, y visualización, así como facilidades para su aplicación y análisis de prestaciones cuando son aplicadas a los datos de entrada seleccionados. Estos paquetes pueden ser integrados en cualquier proyecto de análisis de datos, e incluso pueden extenderse con contribuciones de los usuarios que desarrollen nuevos algoritmos. Con objeto de facilitar su uso por un mayor número de usuarios, **WEKA** incluye además una *interfaz gráfica de usuario* para acceder y configurar las diferentes herramientas integradas.

2.2. Preparación de los datos

Los datos de entrada a la herramienta, sobre los que operarán las técnicas implementadas, deben estar codificados en un formato específico, denominado *ARFF (Attribute Relation File Format)*. La herramienta permite cargar los datos desde: un fichero de texto, una base de datos y mediante acceso a través de internet a la dirección URL de un servidor web.

El formato está compuesto por una estructura claramente diferenciada en tres partes:

1. **Cabecera.** Se define el nombre de la relación, con el siguiente formato:

$$@relation < nombre - relacion >$$

   donde $< nombre - relacion >$ es de tipo *String*. Si dicho nombre contiene algún espacio será necesario expresarlo entre comillas.

2. **Declaración de atributos.** Se declaran los atributos que compondrán el archivo junto a su tipo, con la siguiente sintaxis:

$$@attribute < nombre - atributo >< tipo >$$

   donde $< nombre - atributo >$ es de tipo *String*, con las mismas restricciones que en el caso anterior. El tipo de atributos que acepta **WEKA** se muestra en la siguiente lista,

   a) **NUMERIC.** Expresa números reales, cuya separación de la parte decimal y entera se realiza mediante un punto en vez de una coma.

   b) **INTEGER.** Expresa números enteros.

   c) **DATA.** Expresa fechas, que deben ir precedidas de una etiqueta de formato situada entrecomillas. Esta etiqueta se compone por caracteres separadores

y unidades de tiempo *(dd (Día), MM (Mes), yyyy (Año), HH (Horas), mm (Minutos) y ss (Segundos))*.

d) **STRING.** Expresa cadenas de texto, con las mismas restricciones comentadas anteriormente.

e) **ENUMERADO.** Este tipo de identificador consiste en expresar, entre llaves y separados por comas, los posibles valores que puede tomar el atributo. Por ejemplo,

$$@attributetiempo(soleado, lluvioso, nublado)$$

3. **Sección de datos.** Declaramos los datos que componen la relación, separados entre comas los atributos y con saltos de línea las relaciones. También se puede definir de una forma abreviada cuando tenemos una muestra con muchos valores nulos, pudiendo prescindir ellos. Para ello, sería necesario situar cada una de las filas entre llaves y colocar delante de cada uno de los datos el número del atributo (la numeración de los atributos comienza en 0).

En el caso de que algún dato sea desconocido se expresará con un símbolo de cerrar interrogación *("?")*. Es posible añadir comentarios con el símbolo *"%"*, que indicará que desde ese símbolo hasta el final de la línea es todo un comentario.

```
% Archivo de prueba para Weka.
@relation prueba

@attribute nombre STRING
@attribute ojo_izquierdo {Bien,Mal}
@attribute dimension NUMERIC
@attribute fecha_analisis DATE "dd-MM-yyyy HH:mm"

@data
Antonio,Bien,38.43,"12-04-2003 12:23"
'Maria Jose',?,34.53,"14-05-2003 13:45"
Juan,Bien,43,"01-01-2004 08:04"
Maria,?,?,"03-04-2003 11:03"
```

Figura 2.1: Ejemplo de un archivo de prueba *(prueba.arff)*

2.3. Instalación del software WEKA

WEKA se puede descargar a través de la web de la Universidad de Waikato por el siguiente enlace *https://www.cs.waikato.ac.nz/ml/weka/downloading.html*, observando que la última versión es **WEKA 3.8**. Además se pueden consultar, desde el manual de referencia para la aplicación y otras publicaciones relacionadas, como descargar ejemplos para realizar ensayos con esta herramienta.

Una vez descargado exitosamente el software, el usuario podrá instalar **WEKA** siguiendo las directrices siguientes:

1. Comenzamos abriendo la ubicación del archivo, apareciendo el asistente. Para continuar cliqueamos en *Next*.

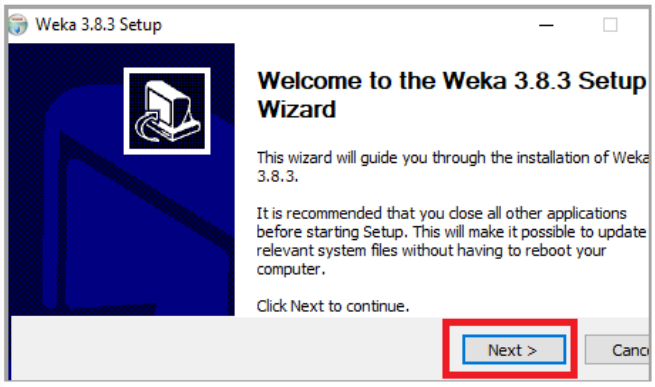


Figura 2.2: Asistente

2. Seguidamente nos aparecerán los términos del Acuerdo de licencia, los cuales se leerán y aceptarán cliqueando en *I Agree*.

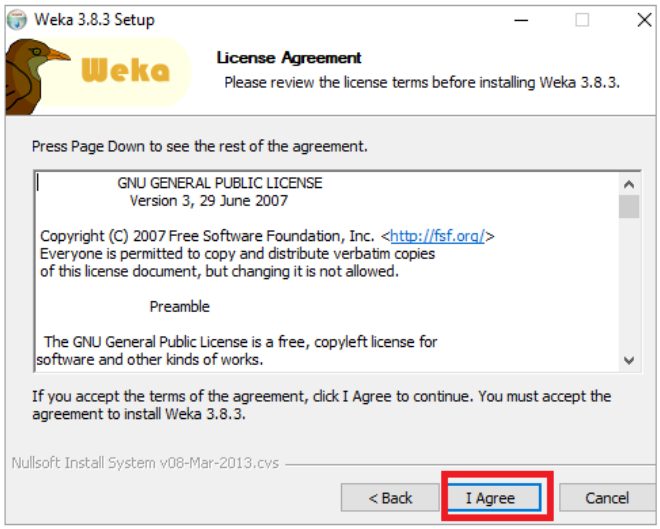


Figura 2.3: Términos del Acuerdo de licencia

3. Realizaremos una instalación completa de todas las componentes y cliquearemos en *Next*.

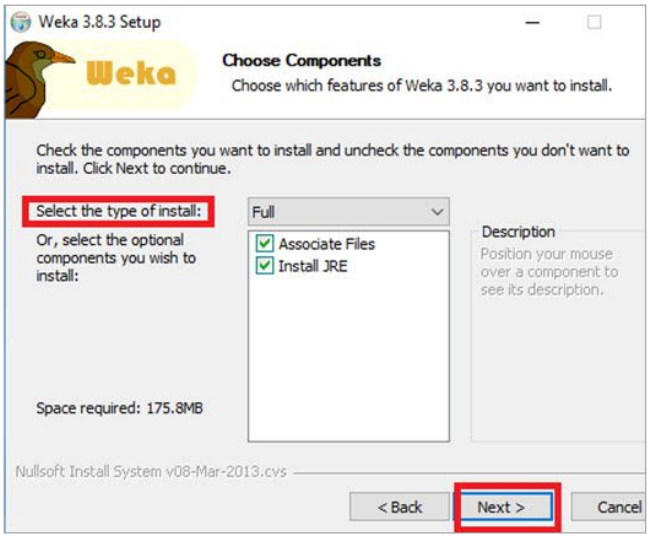


Figura 2.4: Instalación de componentes

4. Seleccionaremos la carpeta de destino y cliquearemos en *Next*. De esta forma se comenzará con la instación.

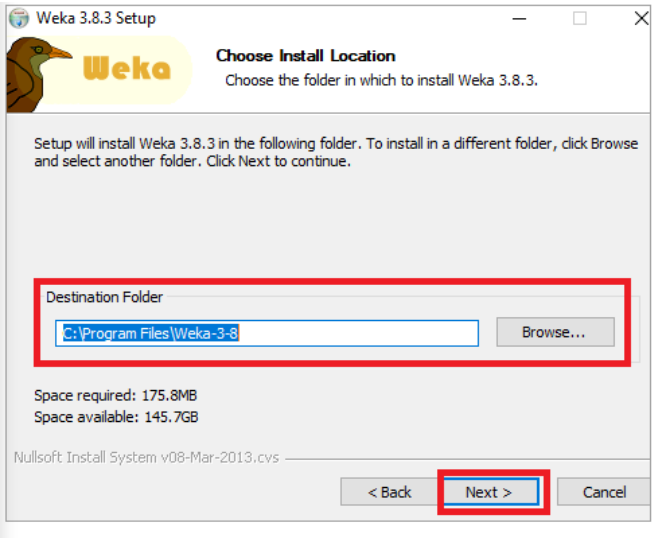


Figura 2.5: Instalación del software

5. Finalizada la instalación aparecerá la siguiente ventana,teniendo que cliquear en *Next*.

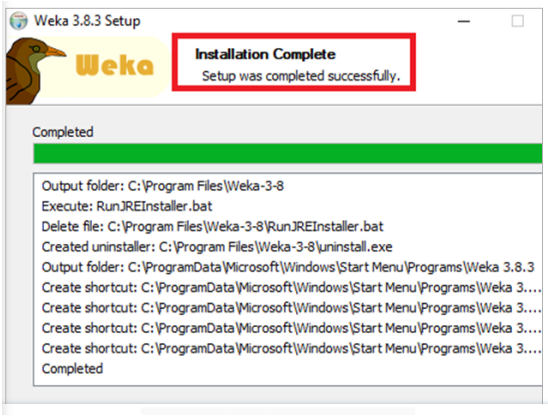


Figura 2.6: Instalación completada

6. Por último lugar, seleccionamos en la ventana que aparece, la casilla de *Star Weka* e cliqueamos en *Finish*.

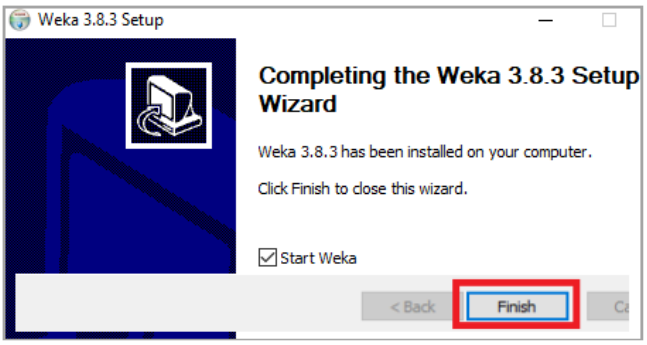


Figura 2.7: Instalación finalizada

2.4. Interfaz de usuario

Al ejecutar el programa nos aparecerá el selector de **interfaz de Weka** que da la opción de seleccionar entre cinco posibles aplicaciones *(interfaces)* para acceder a las funcionalidades del programa de **WEKA**, como se muestra en la siguiente imagen

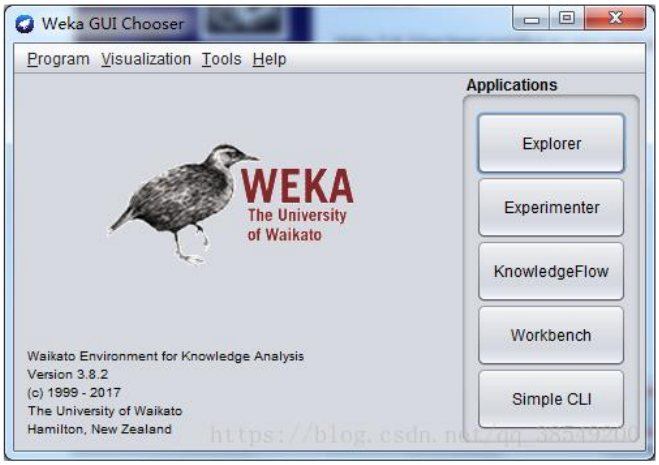


Figura 2.8: Interfaz principal de WEKA

Éstas son:

- **Simple CLI (Simple Command-Line Interface):** es una consola que permite acceder a todas las opciones de **WEKA** desde línea de comandos. Esta sencilla

interfaz de línea de comandos se utiliza para interactuar con los usuarios y puede ejecutar directamente comandos de Weka.

- **Explorer (Explorador):** es la opción que permite llevar a cabo la ejecución de los algoritmos de análisis implementados sobre los ficheros de entrada, una ejecución independiente por cada prueba.

- **Experimenter (Experimentador):** esta opción permite definir experimentos más complejos, con objetivo de ejecutar uno o varios algoritmos sobre uno o varios conjuntos de datos de entrada, y comparar estadísticamente los resultados.

- **KnowledgeFlow (flujo de conocimiento):** es el interface gráfico utilizado para desarrollar proyectos a través de flujos de información. La interfaz de flujo de conocimiento permite a los usuarios arrastrar y soltar componentes gráficos que representan algoritmos de aprendizaje y fuentes de datos en la pantalla, y combinarlos de una determinada manera y secuencia.

- **Workbench (Mesa de trabajo):** reune todas las interfaces en un solo lugar.

Para una explicación más detallada de cada opción, se representarán los apartados siguientes.

2.5. Simple CLI

Simple CLI *(Simple Command-Line Interfaz)* es una interfaz que proporciona una consola para poder introducir órdenes en **JAVA**, de la que se puede acceder a partir de la **interfaz de WEKA**. A pesar de ser en apariencia muy simple es extremadamente potente porque permite realizar cualquier operación soportada por **Weka** de forma directa; no obstante, es muy complicada de manejar ya que es necesario un conocimiento completo de la aplicación. Nace con la primera versión de **WEKA**, aunque actualmente sólo es

útil como una herramienta de ayuda a la fase de pruebas, debido a la aparición de otras aplicaciones.

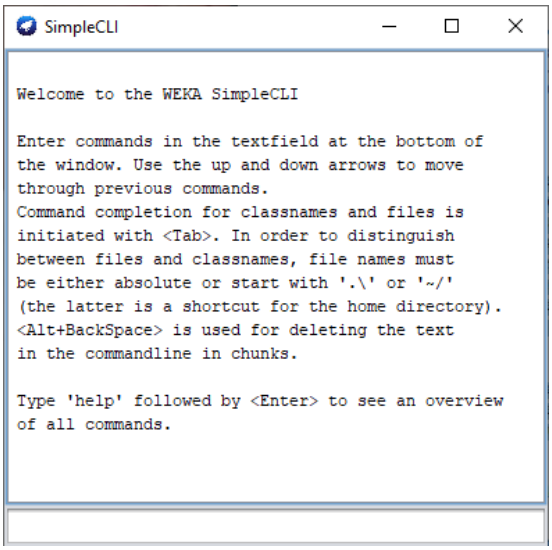


Figura 2.9: Interfaz de Simple CLI

En la interfaz se pueden ejecutar los siguientes comandos:

- $java < nombre - clase >< args >$. Ejecuta el método *"main"* de la clase dada con los argumentos especificados al ejecutar este mandato (en el caso de que realmente se haya proporcionado alguno). Cada mandato es atendido en un hilo independiente por lo que es posible ejecutar varias tareas concurrentemente.
- *break*. Detiene la tarea actual.
- *kill*. Este comando es desaconsejable, sólo debe aplicarse en caso de que la tarea no pueda ser detenida por el comando *break*.
- *cls*. Su extensión es *Clear Screen*, indicando que su función es limpiar el contenido de la consola.
- *exit*. Sale de la aplicación.
- $help < mandato >$. Proporciona una breve descripción de cada mandato.

2.6. Explorer

La interfaz **Explorer** permite llevar a cabo la ejecución de los algoritmos de análisis implementados sobre los ficheros de entrada, una ejecución independiente por cada prueba, a la que se accede por la *interfaz principal de Weka*. Para acceder a los diferentes tipos de operaciones que se pueden realizar sobre los datos, tenemos que acceder a las siguientes pestañas:

- **Preprocess:** Selección de la fuente de datos (introducción de datos) y su preparación (filtrado).
- **Classify:** útil para la aplicación de algoritmos de clasificación y de regresión (entrenar modelos y evaluar su precisión).
- **Cluster:** Herramienta que aplica algoritmos de agrupamiento.
- **Associate:** Aplica algoritmos de búsqueda de reglas de asociación.
- **Select Attributes:** Búsqueda supervisada de subconjuntos de atributos representativos (selección de atributos).
- **Visualize:** Herramienta que permite visualizar los datos por parejas de atributos de manera interactiva.

Una vez seleccionada, aparece la siguiente ventana con todas las pestañas en la parte superior,

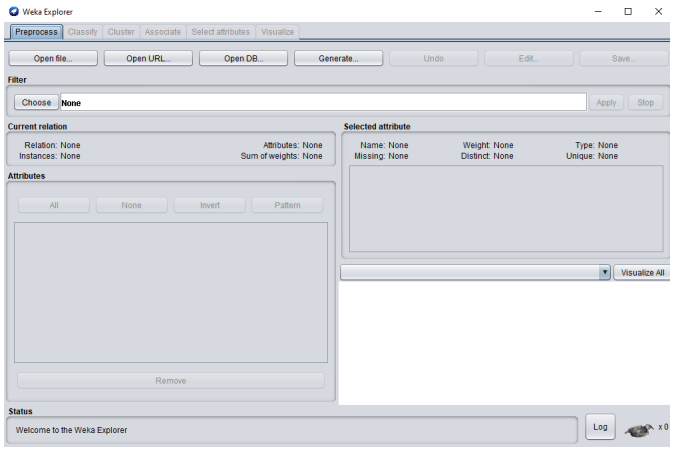


Figura 2.10: Interfaz de Explorer

Además de estas pestañas de selección, en la parte inferior de la ventana aparecen dos elementos comunes: el botón *"Log"*, que al activarlo presenta una ventana textual donde se indica la secuencia de todas las operaciones que se han llevado a cabo dentro del **Explorer**, junto a sus tiempos de ejecución y mensajes de error; y el indicador *status*, que muestra qué tarea se está realizando en este momento dentro del **Explorer**.

2.6.1. Pestaña Preprocces

La opción **Preprocess** corresponde a la primera pestaña de la ventana principal de la interfaz **Explorer**, que incluye filtros y diferentes herramientas para la manipulación de datos. Antes de seleccionar ninguna otra opción es preciso para comenzar, introducir los datos con los que vamos a trabajar, existiendo cuatro posibilidades según el origen de los datos: fichero de texto *Open file*, dirección URL *Open URL*, base de datos *Open DB* o introducción de la información directamente en la aplicación *Generate*. Estas posibilidades se encuentran en la parte superior de la ventana.

Por ejemplo, si a partir de la web de **WEKA** descargamos el fichero de datos *weather.arff* para cargarlo en el programa, necesitaremos selecionar en este caso el botón *Open file* y buscar el archivo. El resultado se muestra a contnuación,

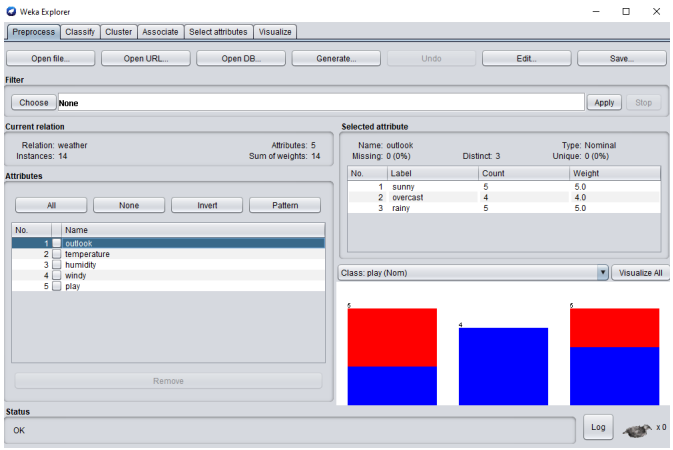


Figura 2.11: Introducción del fichero de datos *weather.arff*

Una vez cargados los datos, aparece un cuadro resumen, *Current relation*, con el nombre de la relación que se indica en el fichero, el número de instancias y el número de atributos. Más abajo, aparece una lista con todos los atributos disponibles, con los nombres especificados en el fichero, de modo que se pueden seleccionar para ver sus detalles y propiedades, apareciendo en la parte derecha las propiedades del atributo seleccionado *(Selected attribute)*. Según el tipo de atributo, se presentará en *Selected attribute*, el número de instancias (atributo nominal) o los valores máximo, mínimo, valor medio y desviación estándar (atributo numérico). Otras características que se destacan del atributo seleccionado son el tipo *(Type)*, el número de valores distintos *(Distinct)*, el número y porcentaje de instancias con valor desconocido para el atributo *(Missing)*, y los valores de atributo que solamente se dan en una instancia *(Unique)*.

Además, en la parte inferior se representan gráficamente los valores que toma el atributo seleccionado, mostrándose una distribución de frecuencias, si es de tipo nominal, o un histograma con intervalos uniformes, si es numérico. Por último, hay un botón *"Visualize All"* que permite representar los histogramas de todos los atributos simultáneamente.

Por otro lado, la opción de preprocesamiento *(Filter)* permite realizar filtros sobre los datos con el objetivo de realizar manipulaciones sobre los mismos en dos niveles:

- *Atributos:* Actúan en vertical en la base de datos, modificando un atributo completo. Se utilizan por ejemplo para eliminar atributos, para discretizar atributos numéricos, y para añadir nuevos atributos con expresiones, entre otros.

- *Instancias:* Actúan en horizontal, seleccionando un grupo de registros. Se utilizan por ejemplo para la selección de instancias (selección de rangos, muestreos, etc.).

Para aplicar cualquier filtro integrado en Weka, en función de nuestro objetivo, seleccionaremos el botón *Choose* en *Filter* desplegándose un árbol donde aparecen los filtros *(Figura 2.12)*, divididos en:

- *Supervisados:* son los que necesitan conocer información acerca de la salida, es decir, conocer la clase asociada a los datos. Esto les hace ser poco aplicables en problemas reales, donde rara vez se conocen de antemano la clasificación correcta.

- *No supervisados:* son aquellos de los cuales no podemos conocer en el momento de la clasificación su clase, actuando en unión con clasificadores para analizar su efecto y guiar su actuación. Esto les hace ser los más aplicables en problemas reales.

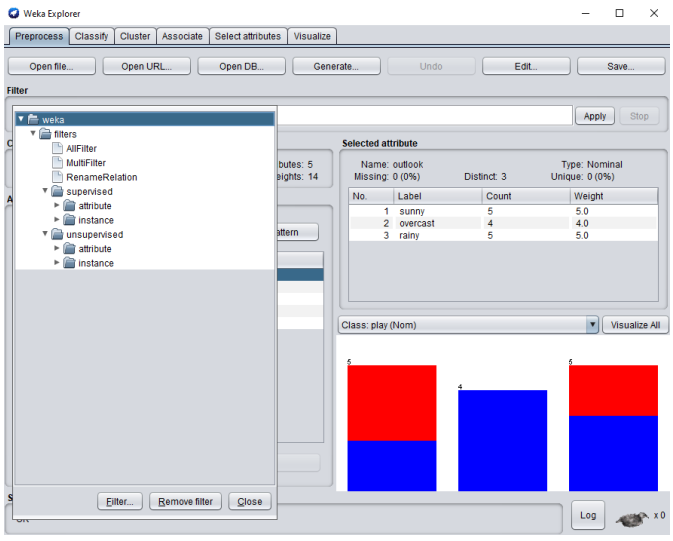


Figura 2.12: Representación de los filtros

2.6.2. Pestaña Classify

La opción **Classify** corresponde a la segunda pestaña de la ventana principal de la interfaz **Explorer**, pudiendo definir y resolver un problema de clasificación. En ocasiones, el problema de clasificación se formula como un refinamiento en el análisis, una vez que se han aplicado algoritmos no supervisados de agrupamiento y asociación para describir relaciones de interés en los datos.

Además, se pretende construir un modelo que permita predecir la categoría de las instancias en función de una serie de atributos de entrada. En Weka, la clase es uno de los atributos nominales disponibles, que se convierte en la variable objetivo a predecir. Por defecto, es el último atributo (última columna) a no ser que se indique otro explícitamente. La configuración de la clasificación se efectúa en la ventana siguiente:

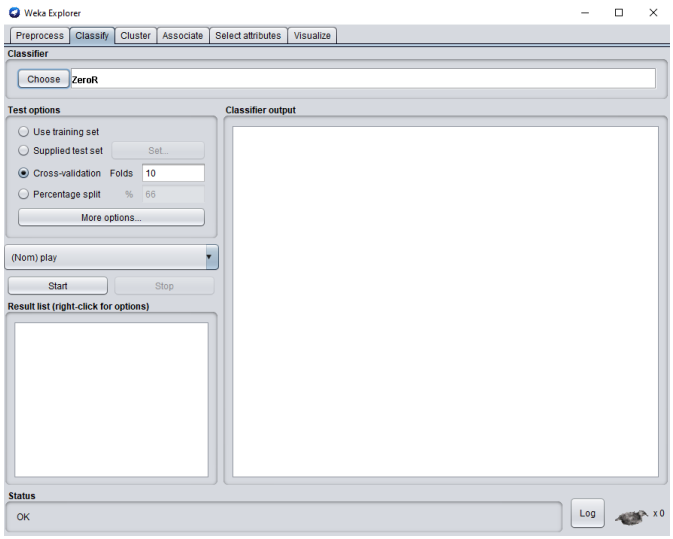


Figura 2.13: Pestaña Classify

Para realizar una clasificación, primero se tendrá que elegir un clasificador y configurarlo a nuestro gusto, para ello pulsaremos sobre el botón *Choose* en *Classifier* desplegando un árbol que nos permitirá seleccionar el clasificador deseado, teniendo en cuenta que por defecto aparece en la etiqueta contigua al botón *Choose* el calsificador *ZeroR*. Una vez

seleccionado, aparecerá en la etiqueta contigua el filtro seleccionado y los argumentos con los que se ejecutará, los cuales se pueden modificar haciendo doble click sobre el filtro.

Elegido el clasificador y sus características, el próximo paso será la configuración del modo de entrenamiento *(Test Options)*, debido a que el resultado de aplicar el algoritmo de clasificación se efectúa comparando la clase predicha con la clase real de las instancias, cuya validación puede realizarse a través de las siguientes opciones:

- *Use training set*. Esta opción evalúa el clasificador sobre el mismo conjunto sobre el que se construye el modelo predictivo para determinar el error, que se denomina error de resustiución. Por tanto, esta opción puede proporcionar una estimación demasiado optimista del comportamiento del clasificador al evaluarlo sobre el mismo conjunto sobre el que se hizo el modelo.

- *Supplied test set*. Se realiza la evaluación sobre el conjunto independiente. Esta opción permite seleccionar un fichero de datos *(seleccionando Set)*, con el que se probará el clasificador obtenido con el método de clasificación usado y los datos iniciales.

- *Cross-validation*. Evaluación con validación cruzada estratificada. En esta opción se realizan tantas evaluaciones como se indica en el parámetro *Folds*. Se dividen las instancias en tantas carpetas como indica este parámetro *(Folds)* y en cada evaluación se toman las instancias de cada carpeta como datos de test, y el resto como datos de entrenamiento para construir el modelo.

- *Percentage split*. Esta opción divide los datos en dos grupos, de acuerdo con el porcentaje indicado (%). El valor indicado es el porcentaje de instancias para construir el modelo, que a continuación es evaluado sobre las que se han dejado aparte. Cuando el número de instancias es suficientemente elevado, esta opción es suficiente para estimar con precisión las prestaciones del clasificador en el dominio.

Además de estas opciones para seleccionar el modo de evaluación, podemos acceder por el botón *More Options* a un cuadro con otras opciones adicionales: *Output model*, *Output per-class stats*, *Output entropy evaluation measures*, *Store predictions for visualization* y *Cost-sensitive evaluation*.

Finalmente se podrá visualizar los resultados en la sección *Classifier output* pulsando el botón *Start*, en función del algoritmo de clasificación seleccionado. Además en la sección *Result list* aparecerá una lista de todos los experimentos realizados, de la que podemos acceder a una lista de opciones haciendo click derecho sobre algún experimento para visualizar de manera diferente los resultados obtenidos incluyendo la ejecución de gráficas.

2.6.3. Pestaña Cluster

La opción **Cluster** corresponde a la tercera pestaña de la ventana principal de la interfaz **Explorer**, que permite realizar agrupamientos de las instancias de la base de datos basándose en las semejanzas y diferencias que existen entre los datos que componen la muestra. Estas técnicas aplican algoritmos no supervisados en bases de datos en las que la variable clase no se ha definido, con el objetivo de descubrir dichas clases o estructuras diferenciadas en los datos.

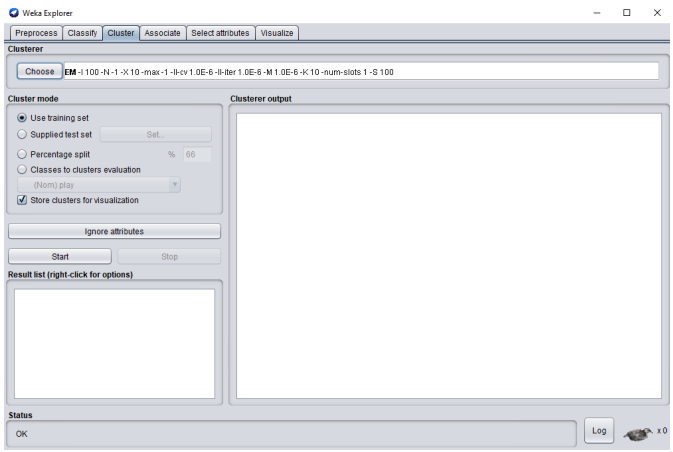


Figura 2.14: Pestaña Cluster

El funcionamiento y la distribución de los apartados son similares al de la pestaña de *clasificación*. Por tanto, se elegirá un método de clustering selecionando *Choose*, se indicará el modo de entrenamiento *(Training set)* y con el botón *Start* se ejecutará el experimento. También, antes de ejecutar el experimento, se puede activar la opción *Store cluster for evaluation* para tener la posibilidad de ver de una forma gráfica la asignación de las muestras en cluster.

Finalmente se podrán visualizar los resultados en la sección *Classifier output*, en función del método cluster seleccionado. Además en la sección *Result list* aparecerá una lista de todos los experimentos realizados de la que podemos acceder a una lista de opciones haciendo click derecho sobre algún experimento, llegando a otra forma de visualizar la gráfica de asignación de las muestras en clusters, a partir de la opción *Visualize cluster assignments*.

2.6.4. Pestaña Associate

La opción **Associate** corresponde a la cuarta pestaña de la ventana principal de la interfaz **Explorer**, que permite aplicar métodos orientados a buscar asociaciones entre datos. Es importante reseñar que estos métodos sólo funcionan con datos de tipo nominal, además de ser algoritmos no supervisados por no disponer de una variable clase definida con anterioridad. Por tanto, tiene como objetivo descubrir dichas clases o estructuras diferenciadas en los datos.

Éste es sin duda el apartado más sencillo y más simple de manejar, carente de apenas opciones, su funcionamiento se basa en elegir un método por *Choose*, configurarlo y ejecutar el experimento con el botón *Start*.

Finalmente se podrán visualizar los resultados en la sección *Associator output*, en función del método seleccionado. Además en la sección *Result list* aparecerá una lista de todos los experimentos realizados de la que podemos acceder a una lista de opciones haciendo click derecho sobre algún experimento.

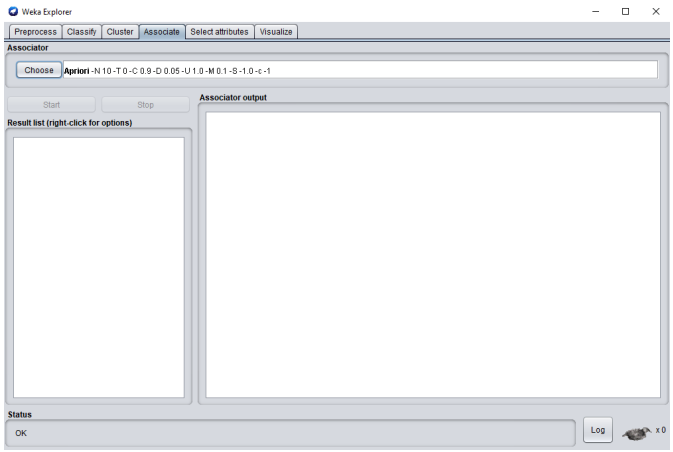


Figura 2.15: Pestaña Associate

2.6.5. Pestaña Select attributes

La opción **Select attributes** corresponde a la quinta pestaña de la ventana principal de la interfaz **Explorer**, que permite acceder al área de selección de atributos. El objetivo de estos métodos es identificar, mediante un conjunto de datos que poseen unos ciertos atributos, aquellos atributos que tienen más peso a la hora de determinar si los datos son de una clase u otra. En caso de tener un número excesivo de atributos, se podría obtener un modelo demasido complejo produciendo un sobreajuste.

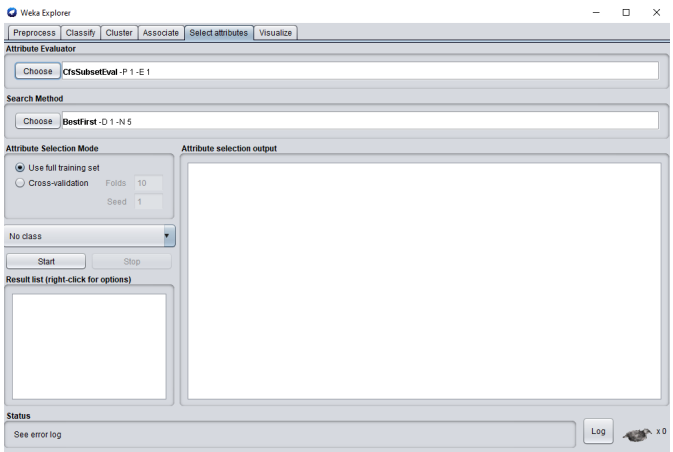


Figura 2.16: Pestaña Select attributes

En Weka, la selección de atributos se puede hacer de varias maneras, aunque la forma más directa es usando la pestaña **attribute selection**. En ella aparecen dos métodos denominados:

- *Método de evaluación (Attribute Evaluator):* es la función que determina la calidad del conjunto de atributos para discriminar la clase. Se pueden distinguir entre dos métodos: los que directamente utilizan un clasificador específico para medir la calidad del subconjunto de atributos a través de la tasa de error del clasificador y los que no.

- *Método de búsqueda (Search Method):* es la forma de realizar la búsqueda de conjuntos de forma eficiente.

El funcionamiento para seleccionar este método es el mismo que con otros métodos en Weka, siendo necesario elegir un método de busqueda *(Attribute Evaluator)* y un método de evaluación *(Search Method)*. Para ello, selecionamos el botón *Choose* situado dentro de cada cuadro, desplegándose un árbol de algoritmos. Una vez seleccionados, podemos acceder a sus propiedades pulsando sobre el nombre de la etiqueta que muestra el nombre del método seleccionado.

Una vez seleccionado el método de búsqueda y de evaluación, se indicará si la selección de los atributos se realizará usando un conjunto completo de entrenamiento o mediante validación cruzada *(Attribute Selection Mode)*, y con el botón *Start* se ejecutará el experimento.

Finalmente se podrán visualizar los resultados en la sección *Attribute selection output*, en función de los métodos seleccionados. Además en la sección *Result list* aparecerá una lista de todos los experimentos realizados de la que podemos acceder a una lista de opciones haciendo click derecho sobre algún experimento. Una de ellas es la opción *Visualize Reduced Data*, que nos mostrará los datos habiendo tomado los mejores atributos en una ventana.

2.6.6. Pestaña Visualize

Una de las primeras etapas recomendables para el análisis de datos puede ser el análisis visual de estos, siendo de gran utilidad para desvelar relaciones de interés a partir de nuestra capacidad para comprender imágenes.

La opción **Visualize** corresponde a la sexta pestaña de la ventana principal de la interfaz **Explorer**, que muestra gráficamente la distribución de todos los atributos mostrando gráficas en dos dimensiones, en las que va representando en ambos ejes todos los posibles pares de combinaciones de los atributos. Este modo nos permite ver correlaciones y asociaciones entre los atributos de una forma gráfica, que pueden ser almacenadas posteriormente en formato *ARFF*.

Introducido el archivo de datos *weather.arff*, en la pestaña **Preprocces**, vamos a visualizar las gráficas en la pestaña **Visualize** como se muestra seguidamente,

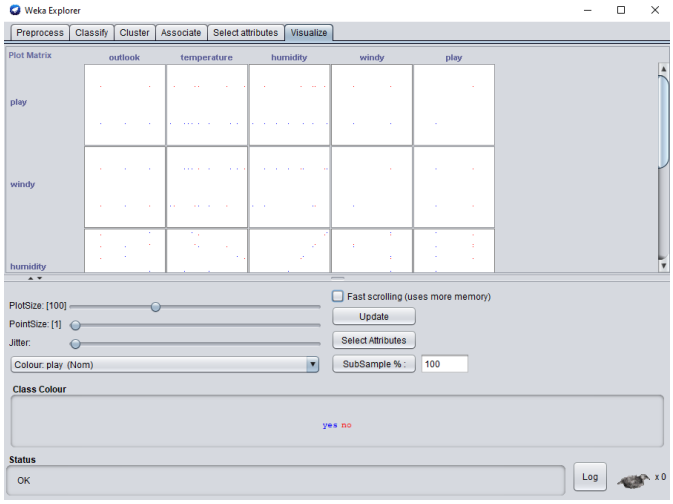


Figura 2.17: Pestaña Visualize utilizando los datos *weather.arff*

Si realizamos un doble click sobre cualquier gráfica se nos mostrará en una ventana nueva la gráfica seleccionada más ampliada. Las opciones de edición que aparecen debajo de las gráficas se activan mediante las barras deslizantes. Las posibles opciones son:

- *Plotsize:* define el tamaño del lado de cada una de las gráficas en píxeles.
- *Pointsize:* define el tamaño de los puntos de las gráficas en píxeles.
- *Jitter:* Añade un ruido aleatorio a las muestras, de manera que espacia las muestras que están físicamente muy próximas, esto tiene útilidad cuando se concentran tanto los puntos que no es posible diferenciar la cantidad de éstos en un área.
- *Colour:* define el color que se utilizará para las clases de los atributos.

Una vez seleccionado las modificaciones pertinentes, es imprescindible pulsar el botón *Update* para que se representen de nuevo las gráficas. Otro botón útil, es el *Select Atributes*, que nos permite elegir los atributos que se representarán en las gráficas. El último botón que se encuentra en esta ventana es el *Subsample*, que permite definir el tanto por ciento de muestras (que escogerá aleatoriamente) que queremos representar.

2.7. Experimenter

La interfaz **Experimenter** *(Experimentador)* es muy útil para aplicar uno o varios algoritmos de clasificación sobre un gran conjunto de datos, para luego poder realizar contrastes de hipótesis entre ellos y obtener otros índices estadísticos. Por tanto, podemos comparar cómo se comportan distintos algoritmos de aprendizaje sobre nuestros conjuntos de datos y decidir, según las estadísticas que nos da, cuáles son los que mejores resultados dan.

La pantalla principal de la interfaz se compone de varias ventanas dentro de un orden lógico correspondiente a las diferentes etapas de extracción:

- *Setup.* Se configura el experimento.
- *Run.* Se ejecuta el experimento.
- *Analyse.* Analiza los resultados.

2.7.1. Pestaña Setup

Abierta la interfaz, observamos una ventana que corresponde a la pestaña *Setup*. Por defecto la interfaz **Experimenter** está configurada en modo simple, no obstante, esto puede variarse a modo avanzado en la sección *Experiment Configuration Mode*. Para empezar se explicará el modo simple y posteriormente se pasará a explicar el modo avanzado.

La **configuración simple** es la que aparece por defecto *(Figura 2.18)*. En ella habrá que definir un fichero de configuración que contenga todos los ajustes pertenecientes a un experimento. Para ello, podemos abrir uno ya creado a partir del botón *Open* o crear uno nuevo con el botón *New*, el cual debe ser guardarlo con el botón *Save*. Además, podremos guardar los resultados del experimento en un fichero de tipo *ARFF*, *CSV* o en una base de datos, seleccionando la opción dentro del cuadro *Results Destination*.

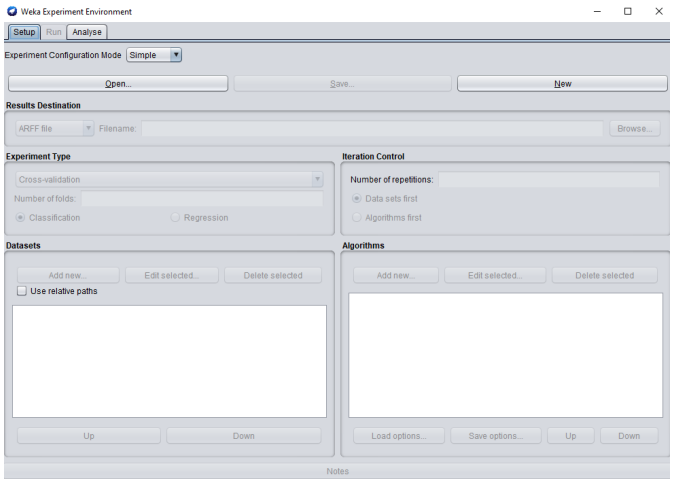


Figura 2.18: Ventana con una configuración simple

El siguiente paso será definir el tipo de validación que tendrá el experimento en la sección *Experiment Type*, distinguiéndose entre: validación-cruzada estratificada, entrenamiento con un porcentaje de la población tomando ese porcentaje de forma aleatoria y entrenamiento con un porcentaje de la población tomando el porcentaje de forma ordenada.

Realizados los pasos anteriores, deberemos indicar qué archivos de datos queremos que formen parte de nuestro experimento, a partir del cuadro *Datasets*. Seleccionando la pestaña *Use relative paths* facilitaremos poder llevar a cabo experimentos sobre los mismos conjuntos de datos en otros ordenadores que no comparten las mismas rutas de los archivos. Una vez seleccionada o no esta pestaña, pasamos a añadir nuestros archivos con el botón *Add new*, que nos permite seleccionar archivos o carpetas.

En el cuadro denominado *Iteration control* definiremos el número de repeticiones de nuestro experimento, especificando si queremos que se realicen primero los archivos de datos o los algoritmos. Debajo de este cuadro se encuentra la sección *Algorithms*, en la que podremos añadir los algoritmos que queramos con el botón *Add new*. En caso de querer eliminar alguno de los seleccionados pulsamos el botón *Delete selected*.

La **configuración advanced** se indica en la sección *Experiment Configuration Mode (Figura 2.19)*. El funcionamiento de este modo está orientado a realizar tareas específicas más concretas que un experimento simple. A través de esta opción es posible repetir el experimento variando el tamaño del conjunto de datos, distribuir la ejecución del experimento entre varios ordenadores o realizar experimentos de modo incremental.

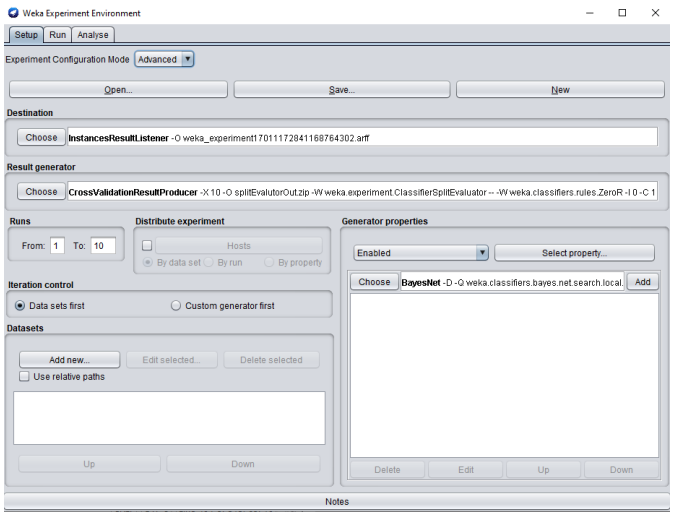


Figura 2.19: Ventana con una configuración advanced

En el cuadro *Destination* se guardará el resultado del experimento, siendo su funcionalidad la misma que en la configuración simple, pero no su funcionamiento. En este caso, debemos elegir el método a utilizar para la exportación de los resultados (Base de datos, fichero *ARFF* o fichero *CSV*) por el botón *Choose*, y configurarlo pulsando sobre la etiqueta del método.

Ahora el cambio fundamental entre una interfaz y otra se basa en el *Result Generator*. Para ello, es necesario seleccionar con el botón *Choose* qué método generador de resultados utilizaremos. Seguidamente lo configuraremos haciendo doble click sobre la etiqueta del método. En Weka se permite seleccionar los métodos siguientes:

- *CrossValidationResultProducer.* Genera resultados fruto de una validación cruzada.
- *AveragingResultProducer.* Toma los resultados de un método generador de resultados y calcula los promedios de los resultados.
- *LearningRateResultProducer.* Llama a un método generador de resultados para ir repitiendo el experimento variando el tamaño del conjunto de datos.
- *RandomSplitResultProducer.* Genera un conjunto de entrenamiento y de prueba aleatorio para un método de clasificación dado.
- *DatabaseResultProducer.* Toma los resultados de una base de datos, coincidiendo con los resultados obtenidos con un método generador. Si existen datos que no aparezcan en la base de datos se calculan.

Escogido el generador de resultados, podremos pasar al *Generator properties* configurando de una forma más sencilla algunas de las propiedades de la sección *Result Generator*. Para activarlo, es necesario pulsar el botón donde pone *Disabled (desactivado)* y elegir *Enabled (activado)*. Además, en la sección *Runs* se puede seleccionar las iteraciones con la que se realizará el experimento.

Indicar que una de las características más interesantes de la interfaz **Experiment**, es que permite distribuir la ejecución de un experimento entre varios ordenadores mediante **Java RMI**, a partir de la sección *Distribute experiment*. Otras secciones que aparecen son *Iteration control*, donde se establece el orden de la iteración, y *Datasets*, donde se definen los conjuntos de datos sobre los que actuarán los algoritmos de aprendizaje.

2.7.2. Pestaña Run

La ventana para comenzar la ejecución del experimento se obtiene pulsando la segunda pestaña superior, etiquetada como **Run**. En esta sencilla pestaña, las únicas opciones que tenemos son ejecutar los experimentos descritos en *Setup* con *Start* o detener la ejecución en transcurso con *Stop*.

Finalmente, en *Log* nos aparecerán los tiempos y las incidencias que ocurran a lo largo del proceso, mientras que en *Status* veremos si se está ejecutando algún algoritmo, qué algoritmo es, sobre qué archivo se está ejecutando y cuántas iteraciones ha realizado ya.

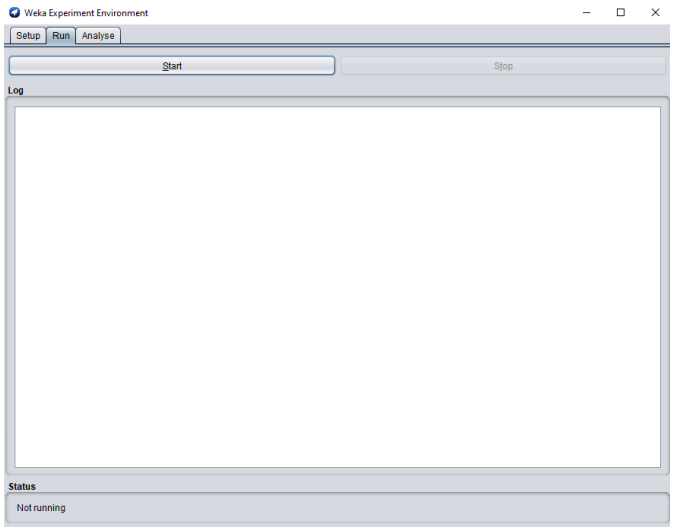


Figura 2.20: Pestaña Run

2.7.3. Pestaña Analyse

La ventana para comenzar el análisis del experimento se obtiene pulsando la tercera pestaña superior, etiquetada como **Analyse**. Ésta nos permite ver los resultados de los experimentos, realizar contrastes estadísticos, etc.

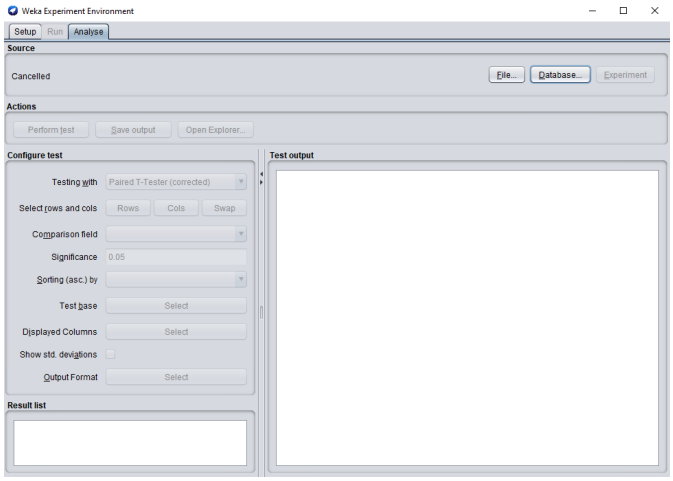


Figura 2.21: Pestaña Analyse

El primer paso es definir, en la sección *Source*, el origen de los datos de los resultados seleccionando los botones *File*, *Database* o *Experiment*. A continuación definimos el test a realizar utilizando las opciones que aparecen en la sección *Configure test*; explicadas brevemente en la siguiente lista:

- *Testing with.* Utiliza la comparación de parejas de esquemas bajo el estándar T-Test (corregido o no).

- *Select rows and cols.* Podemos definir la información de las filas y columnas de la matriz de salida, donde los conjuntos de datos serán las filas y los algoritmos con sus opciones las columnas.

- *Comparison field.* Selecionamos el atributo que va a ser comparado en el contraste.

- *Significance.* Se indica el nivel de significación para realizar el contraste (0,05 por defecto). A menor significación, mayor confianza en las conclusiones.
- *Sorting (asc.) by.* Permite ordenar los resultados, según un atributo, de manera ascendente.
- *Test base.* Podemos realizar un test que realice una matriz comparativa de todos los algoritmos.
- *Displayed Columns.* Nos deja seleccionar qué columnas visualizar.
- *Show std. deviations.* Marcando esta casilla, nos aparecerán las desviaciones estadísticas (útil cuando se ha usado validación cruzada).
- *Output Format.* Nos abre una ventana donde poder elegir la precisión para la media y para las desviaciones estadísticas, así como el formato de la salida. También podemos mostrar la media de cada columna o quitar el nombre del filtro y las opciones de procesado.

Finalmente, clicando el botón *Perform test* de la sección *Actions*, se llevará a cabo el análisis de los resultados cuyos efectos se visualizarán en la sección *Test output*. Además podemos observar un listado de todos los experimentos realizados en la sección *Result list*, cuyas salidas se pueden mostrar en una nueva ventana haciendo clic derecho sobre cualquiera de ellos.

2.8. Knowledge Flow

La interfaz **Knowledge Flow** *(Flujo de conocimiento)* permite llevar a cabo las mismas funciones del **Explorer**, con una configuración totalmente gráfica, inspirada en herramientas de tipo *"data-flow"* para seleccionar componentes y conectarlos en un proyecto de *minería de datos*, desde que se cargan los datos, y se aplican algoritmos de tratamiento

y análisis, hasta el tipo de evaluación deseada. Aunque esta interfaz trabaje de la misma forma que **Explorer**, existe alguna diferencia entre ellos, observando que **Explorer** maneja los datos por lotes mientras que **Knowledge Flow** maneja los datos de manera incremental o por lotes.

A continuación se muestra la ventana principal de esta interfaz:

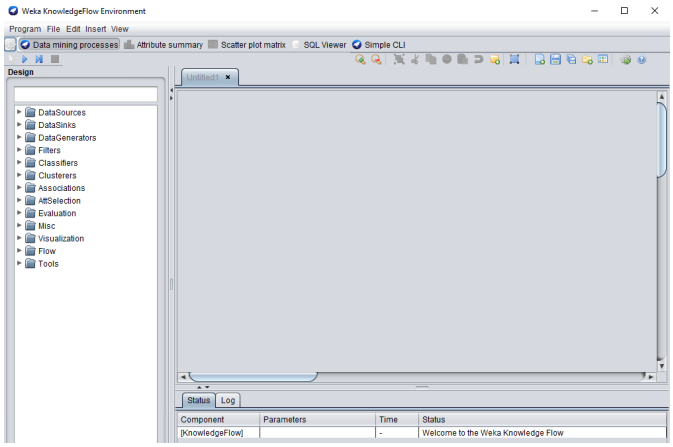


Figura 2.22: Interfaz de Knowledge Flow

Como se puede observar aparece una lista con un conjunto de componentes:

- **DataSources:** Contiene los distintos cargadores de datos, elegiendo el fichero de entrada de datos.
- **DataSinks:** Contiene los almacenadores de datos, es decir, donde se elige el lugar para almacenar los resultados del experimento.
- **Filters:** Se encuentran todos los tipos de filtros estudiados que se aplicarán a los datos.
- **Classifiers:** Contiene los clasificadores de datos, para establecer algoritmos de clasificación y regresión.
- **Clusterers:** Contiene los agrupadores de datos usados en el software.

- **Associations:** Se aplicarán métodos de asociación a los datos.
- **Evaluation:** Contiene distintos métodos de evaluación de datos. A continuación se mencionarán las componentes más utilizadas:
    1. *TrainingSetMaker:* Crea un conjunto de datos en el conjunto de entrenamiento.
    2. *TestSetMaker:* Crea un conjunto de datos en el conjunto de prueba.
    3. *TrainTestSplitMaker:* Divide cualquier conjunto de datos en un conjunto de entrenamiento y un conjunto de prueba.
    4. *ClassAssigner:* Asigna una columna para ser la clase de cualquier conjunto de datos, sea de entrenamiento o de prueba.
    5. *ClassValuePiker:* Elige el valor de una clase considerado como la clase positiva, éste es útil cuando se generan datos para estilos de *curva ROC (Receiver Operating Characteristic)*.
    6. *ClassifierPerformanceEvaluator:* Evalúa el desempeño de los lotes formados y los clasificadores formados.
    7. *ClustererPerformanceEvaluator:* Evalúa el desempeño de los lotes formados y los agrupadores formados.
    8. *CrossValidationFoldMaker:* Divide cualquier conjunto de datos en pliegues.
    9. *PredictionAppender:* Anexa predicciones del clasificador a un conjunto de pruebas. Para problemas de clase discreta, se pueden adjuntar etiquetas de clase o distribuciones de probabilidad.
    10. *IncrementalClassifierEvaluator:* Evalúa el desempeño de los clasificadores formados incrementalmente.
- **Visualization:** Permite representar los gráficos asociados al experimento en diferentes formas. Las componentes más utilizadas en esta área se muestran a continuación:

1. *TextViewer:* Componente para mostrar datos textuales. Puede mostrar conjuntos de datos, estadísticas de rendimiento de clasificación, etc.
2. *AttributeSummarizer:* Muestra en un panel una matriz de histogramas, cada uno de los atributos en los datos cargados.
3. *StripChart:* Componente que puede mostrar un panel donde se visualiza un gráfico con el desplazamiento de los datos, observando el rendimiento de clasificadores incrementales.
4. *ModelPerformanceChart:* Componente que puede mostrar un panel para la visualización de *curvas ROC*.
5. *DataVisualizer:* Permite la aparición de un panel para visualizar los datos en un sólo gráfico de dispersión 3D.
6. *ScatterPlotMatrix:* Muestra en un panel una matriz con pequeños gráficos de dispersión.
7. *GraphViewer:* Muestra un panel con árboles basados en los modelos de decisión.
8. *CostBenefitAnalysis:* Componente que puede mostrar una gráfica emergente, para explorar las compensaciones de costo/beneficio al seleccionar de forma interactiva diferentes tamaños de población de una lista clasificada de prospectos o al variar el umbral en el probabilidad predicha de la clase positiva. Exponiendo un gráfico acumulado de ganancias y un diagrama de costo/beneficio.

Su funcionamiento se basa en situar en el panel de trabajo *(Untitled1)* elementos base (situados en una lista a la izquierda), de manera que creemos un *"circuito"* o flujo que defina nuestro experimento. Por tanto habrá que hacer doble click sobre el elemento que queramos introducir, para poder arrastrarlo y situarlo donde creamos conveniente.

Situados dos elementos base en el panel de trabajo, procederemos a unirlos. Para ello, abrimos el menú de opciones del cargador y mediante la opción *dataset*, dibujamos una

flecha que una ambos objetos. En caso de querer eliminarla, pulsaremos el botón derecho del ratón sobre la flecha y seleccionaremos la opción *Delete*. Otra opción que se puede utilizar es *New note*, la cual permite incluir anotaciones.

Para explicar el funcionamiento de manera más práctica, realizaremos un proceso de *Validación cruzada* sobre un conjunto de datos almacenados en un fichero *ARFF*. Para ello, lo primero será introducir el conjunto de datos, teniendo que seleccionar la opción *ArffLoader* ubicada en **Datasources**. Hay que tener en cuenta que en *Datasources* hay cargadores de datos para *ARFF*, *csv*, *C45* e instancias serializadas; no obstante, no los hay para bases de datos.

El siguiente paso será definir cuál es el atributo que determina la clasificación correcta (habitualmente suele ser el último), teniendo que hacer uso del objeto *Class asigner* incluido en **Evaluation**. Situado en la zona de trabajo y configurado, debemos conectarlo con *ArffLoader*.

En la *Figura 2.23*, se representa un ejemplo sobre lo explicado anteriormente suponiendo un conjunto de datos almacenados en un fichero *ARFF*.

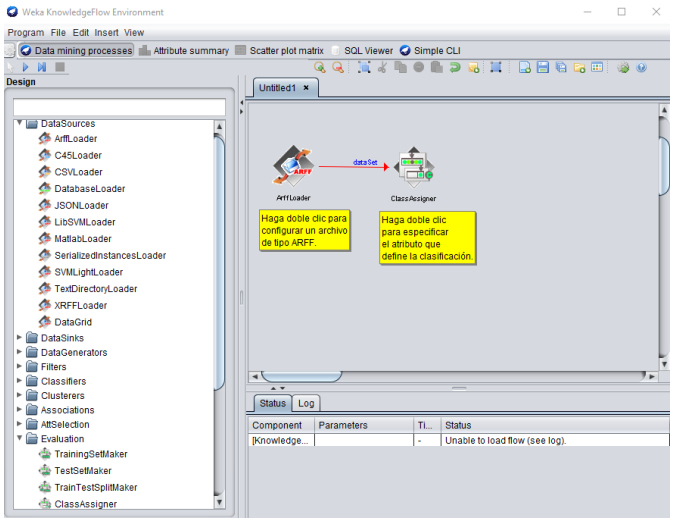


Figura 2.23: Origen de datos definido

Definido por completo el origen de los datos, podemos realizar la *validación cruzada*

que enfrenta al clasificador. Para ello existe el objeto *Crossvalidationfoldmaker* ubicado en **Evaluation**, que será el encargado de realizarnos cada una de las particiones. Situado en la zona de trabajo y configurado, debemos conectarlo con *ClassAsigner*. Este proceso se visualiza en la siguiente imagen,

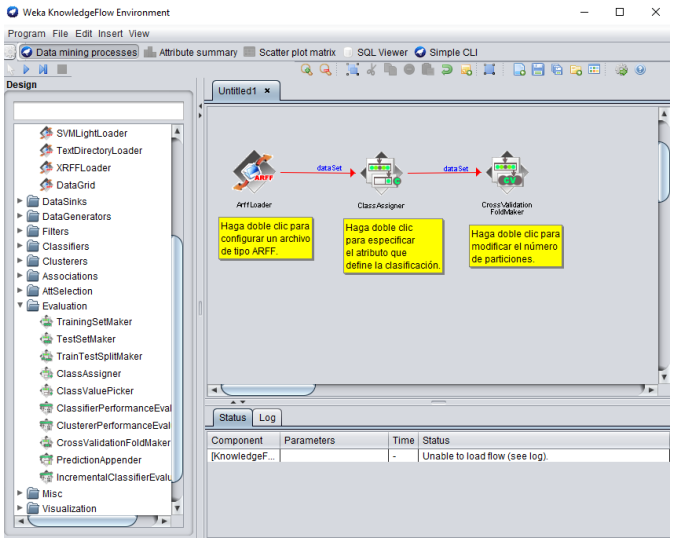


Figura 2.24: Interfaz de KnowledgeFlow

Siguiendo este proceso, podremos construir un experimento completo incluyendo el clasificador y algún objeto para facilitar los resultados, bien sea en modo texto o gráfico. Los objetos que corresponden a los clasificadores se encuentran en el apartado **Classifiers** y los correspondientes a los resultados en **Visualization**.

Finalmente, para comenzar con la ejecución del experimento selecionaremos el objeto ▷, ubicado en la parte superior izquierda. Además para poder observar los resultados eligiremos *Show results* en el menú emergente para el componente *TextViewer*, obteniendo una información mucho más completa.

Capítulo 3

Aplicación a datos reales

3.1. Técnicas de Clasificación aplicadas a datos obtenidos en UCI Machine Learning Repository

El repositorio de aprendizaje automático de **UCI** es una colección de bases de datos, teorías de dominio y generadores de datos que utiliza la comunidad de aprendizaje automático para el análisis empírico de los algoritmos de aprendizaje automático. Desde su creación, ha sido ampliamente utilizado por estudiantes, educadores e investigadores de todo el mundo como fuente principal de conjuntos de datos de aprendizaje automático. La versión actual del sitio web fue diseñada en 2007 por *Arthur Asunción* y *David Newman*, siendo este proyecto una colaboración con la Universidad de Massachusetts Amherst. El apoyo financiero se recibe de *National Science Foundation*.

Actualmente se pueden observar 622 conjuntos de datos de aprendizaje automático de forma gratuita, llegando a obtener información detallada de cada conjunto haciendo click sobre él. Además, se puede seleccionar un conjunto de datos según: la tarea predeterminada *(Default Task)*, el tipo de atributo *(Attribute Type)*, el tipo de datos *(Data Type)*, área *(Area)*, atributos *(Attributes)*, instancias *(Instances)* y tipo de formato *(Format Type)*.

Finalmente, para obtener más información sobre los grupos de trabajo y las líneas de investigación seguidas, se puede acceder a su web https://archive.ics.uci.edu/ml/datasets.php.

3.2. Clasificadores

Como se ha comentado previamente, el aprendizaje automático puede ser usado en varios campos. Uno de ellos muy usado es el de la **clasificación**, donde podemos usar diferentes técnicas o algoritmos que nos ayudarán a extraer información relevante de la que podemos dar uso dependiendo del objetivo buscado.

En Weka, la clase es uno de los atributos nominales disponibles, que se convierte en la variable objetivo a predecir. En caso de tener atributos numéricos, es preciso discretizarlo antes en intervalos que representarán los valores de clase. La función aprendida será capaz de determinar la clase para cada nuevo ejemplo sin etiquetar. El éxito de un algoritmo de aprendizaje para clasificación depende en gran medida de la calidad de los datos que se le proporcionan.

Existen distintos algoritmos de aprendizaje de clasificación. En función de la técnica usada por cada algoritmo la **precisión de acierto** con cada dato será mejor o peor ya que la distribución de los datos y la naturaleza de los mismos podrá afectar de una manera u otra los cálculos internos que él mismo realice.

Como conclusión, la aplicación de un algoritmo de aprendizaje clasificador tiene como objetivo extraer conocimiento de un conjunto de datos y modelar dicho conocimiento para su posterior aplicación en la toma de decisiones. Entre las estructuras más utilizadas están la representación proposicional, los árboles de decisión, las reglas y listas de decisión, reglas con excepciones, reglas jerárquicas de decisión, reglas difusas y probabilidades, y redes bayesianas.

3.3. Evaluación del rendimiento de un clasificador

En la evaluación del comportamiento de los algoritmos de aprendizaje es fundamental estudiar, no sólo la comparación entre algoritmos, sino la eficiencia de clasificación del

modelo sobre una clase sin etiquetar. La forma más habitual de medirla es por la **precisión predictiva** *(accuracy)*, la cual muestra el porcentaje de instancias bien clasificadas de entre todas las seleccionadas como positivas, teniendo en cuenta que si se calcula la precisión sobre el propio conjunto de datos utilizado para generar el modelo, se obtiene con frecuencia una precisión mayor a la real. Por tanto, la idea básica es estimar el modelo con una porción de los datos y luego comprobar su validez con el resto de los datos. Esta separación es necesaria para garantizar la independencia de la medida de precisión resultante; de no ser así, la precisión del modelo será sobreestimada.

Hemos indicado el uso de la **precisión** de un algoritmo como medida de rendimiento del mismo. Sin embargo existen otras métricas que pueden ser consideradas para evaluar el rendimiento de un clasificador, métricas que nos servirán para nuestro trabajo en la medida de poder evaluar y comparar distintos algoritmos de clasificación. Estas métricas nos ayudarán a elegir cuál es el mejor de todos los algoritmos evaluando los distintos valores obtenidos de las métricas.

Para entender las distintas métricas que vamos a plantear a continuación es necesario entender previamente de donde se extraen las mismas. Para ello vamos a proceder a explicar lo que es la **matriz de confusión**, una matriz que nos ayuda a saber cuál ha sido el rendimiento de un algoritmo. Esta matriz simplemente nos muestra en un cuadro los siguientes valores, una vez realizado el entrenamiento y posterior validación con aquellos datos que tenemos para testear usando la validación cruzada: falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos. Para entender mejor este concepto procedamos a mostrar cómo sería esta matriz para una clasificación binaria:

		Predicción	
		Positivos	Negativos
Observación	Positivos	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Tabla 3.1: Matriz de confusión para clasificación binaria

Como podemos observar, las etiquetas conocidas corresponden a las filas, y las que el algoritmo predice a las columnas. Para definir correctamente estos valores podemos indicar que:

- **Verdaderos Positivos (VP).** Tenemos que es la cantidad de aquellas observaciones que fueron clasificados como pertenecientes a una clase y acertó.
- **Falsos Positivos (FP).** Esas observaciones que no pertenecían a una clase pero se llegaron a considerar perteneciente a ellas, son considerados positivos pero como sabemos se ha equivocado.
- **Falsos Negativos (FN).** Aquellas observaciones que pertenecían a una clase pero no se consideraron en ella, considerados como negativos.
- **Verdaderos Negativos (VN).** Es la cantidad de aquellas observaciones que no pertenecían a una clase y las clasificó correctamente.

De todo algoritmo podemos sacar esta **matriz de confusión**, pudiendo extraer los valores indicados previamente:

- **Tasa de verdaderos positivos.** Probabilidad de que una obervación positiva dé como resultado en la predicción positivo. Su fórmula es como sigue:

$$TVP = Pr(rP|oP) = \frac{VP}{VP + FN}$$

- **Tasa de falsos positivos.** Probabilidad de que una obervación negativa dé como resultado en la predicción positivo. Su fórmula es como sigue:

$$TFP = 1 - \frac{VN}{FP + VN}$$

- **Precisión.** Mide la habilidad de un clasificador de no etiquetar como positivo una observación que se debe considerar como negativa. Cuanto mayor sea este valor

mejor será. Su fórmula es:

$$P = \frac{VP}{VP + FP}$$

- **Recall.** Proporción de instancias clasificadas como una clase determinada dividida por el total real en esa clase (equivalente a la tasa de TVP). Su fórmula es como sigue:

$$Recall = TVP$$

- **Medida-F.** Medida de precisión que tiene un test. Su fórmula es:

$$F - Measure = \frac{2TP}{2TP + FP + FN}$$

3.4. Selección y configuración de clasificadores con WEKA

La realización de un problema de clasificación se basa en aplicar un algoritmo de clasificación y en seleccionar el modo de entrenamiento, debido a que el resultado se efectúa comparando la clase predicha con la clase real de las instancias. Esta evaluación puede realizarse de diferentes modos, según la selección en el cuadro *Test options (Use training set, Supplied test set, Cross-validation y Percentage split).*

Siempre se realiza sobre un atributo simbólico que puede ser de tipo nominal o numérico, para el caso de utilizar un atributo numérico se precisa, por tanto, discretizarlo antes en intervalos que representarán los valores de clase. Existen ocho familias de clasificadores, pero los más utilizados son: los bayesianos, los metaclasificadores, las reglas y los árboles de decisión. A continuación se explicará cada uno de estos clasificadores, añadiendo sus principales algoritmos, para facilitar su comprensión antes de ponerlos en práctica.

- **Bayesianos (bayes):** es una técnica que busca determinar relaciones causales que

expliquen un fenómeno y es aplicado en aquellos casos que son de carácter predictivo, otorgando una medida probabilística de la importancia de esas variables en el problema. Hay que tener en cuenta, que entre los atributos del conjunto de entrenamiento no pueden existir correlaciones, para no anular el resultado.

  - *Naïve Bayes:* es un algoritmo de clasificación basado en los teoremas de Bayes, que se puede usar para el modelado exploratorio y predictivo. Supone la hipótesis de que todos los atributos son independientes entre sí, conocido el valor de la variable clase, es decir, asume que la presencia de una característica particular en una clase no está relacionada con la presencia de ninguna otra característica. Por tanto da lugar a un modelo de un único nodo raíz, correspodiente a la clase, y en el que todos los atributos son nodos hoja que tienen como único origen a la variable clase.

- **Metaclasificadores (meta):** incluye todos aquellos clasificadores complejos, que se obtienen mediante composición de clasificadores simples o que incluyen algún preprocesamiento de los datos.

  - *Stacking:* es un meta clasificador, que se basa en la combinación de modelos, construyendo un conjunto con los generados por diferentes algoritmos de aprendizaje. Como cada uno de los modelos se aprende con un mecanismo de aprendizaje diferente, se logra que los modelos del conjunto sean distintos.

- **Reglas (rules):** generación de reglas que podrían aplicarse en un cierto momento, sin barrer exhaustivamente todas las posibilidades. En estos sistemas se busca una condición cuyo cumplimiento sea cubierto por el mayor número posible de instancias de una de las clases, hasta construir una clasificación adecuada.

  - *OneR:* es uno de los clasificadores más sencillos y rápidos, aunque en ocasiones sus resultados son sorprendentemente buenos en comparación con algoritmos

mucho más complejos. Simplemente selecciona el atributo que mejor "explica" la clase de salida. Si hay atributos numéricos, busca los umbrales para hacer reglas con mejor tasa de aciertos.

- **Árboles de decisión (trees):** son una manera práctica de visualizar la clasificación de un conjunto de datos.

  - *J48:* es una implementación del algoritmo **C4.5**, uno de los algoritmos de minería de datos que se ha utilizado en multitud de aplicaciones. Resaltar que uno de los parámetros más importantes de configuración, denominado *confidence level*, influye en el tamaño y capacidad de predicción del árbol construido.

    Una explicación simplificada de este parámetro es la siguiente: para cada operación de poda, define la probabilidad de error que se permite a la hipótesis de que el empeoramiento debido a esta operación es significativo. Cuanto más baja se haga esa probabilidad, se exigirá que la diferencia en los errores de predicción antes y después de podar sea más significativa para no podar. El valor por defecto de este factor es del 25 %, y conforme va bajando se permiten más operaciones de poda y por tanto llegar a árboles cada vez más pequeños.

3.5. Análisis de datos

3.5.1. Descripción de los datos

Los datos utilizados para la realización del trabajo proceden de *UCI Machine Learning Repository*. Este archivo incluye dos conjuntos de datos relacionados con muestras de *"Vinho Verde"* tinto y blanco, del norte de Portugal. En este caso utilizaremos el conjunto de datos denominado *winequality-white.csv*, correspondiente a la calidad del vino portugués *"Vinho Verde"* en su variante blanco, realizado en 2009 por *Paulo Cortez*

(Universidad de Minho), A. Cerdeira, F. Almeida, T. Matos and J. Reis, Comisión de Vitivinicultura de la Región de Vino Verde (CVRVV), Porto, Portugal.

Antes de comenzar con la aplicación de las técnicas de Weka a los datos es muy conveniente revisar los objetivos perseguidos en el análisis, siendo en este caso, encontrar patrones y modelos entre la calidad de los vinos de una comarca, y distintos atributos (sulfatos, grado de alcohol, ph, densidad, etc) que, creemos, condicionan las características del vino.

La base de datos original consta de 4864 instancias y 12 atributos numéricos, relacionadas a características del vino. La descripción de cada una de ellas se recoge en la siguiente tabla:

Atributos	**Descripción**
Acidez fija	es el conjunto de ácidos naturales del vino.
Acidez volátil	es una parte de la acidez total de un vino.
Ácido cítrico	contribuye al equilibrio gustativo del vino.
Azúcar residual	es la cantidad total de azúcar que queda en el vino que no ha sido fermentada por las levaduras.
Cloruros	mide el grado de oxidación del vino.
Anhídrido sulfuroso libre	tiene distintas formas, una de ellas es la forma gaseosa, como dióxido de azufre que es soluble en el agua del vino.
Anhídrido sulfuroso total	está constituido por la suma del anhídrido sulfuroso libre y el anhídrido sulfuroso combinado.
Densidad	se relaciona con la cantidad de alcohol.
PH	este valor nos indica el grado de acidez o basicidad del vino.
Sulfatos	es la cantidad de conservante que posee el vino.
Alcohol	se le relaciona con la densidad, cuanto mayor sea la densidad menor será su grado de alcohol.
Calidad	es el conjunto de propiedades del vino que hacen que se pueda caracterizar y valorar con respecto a otros vinos. Los catadores otorgan una calificación numérica de 0 a 10 puntos.

Tabla 3.2: Descripción de cada atributo

3.5.2. Construcción del archivo *ARFF*

Antes de comenzar nuestro estudio tenemos que realizar algunas simplificaciones a la base de datos original. La primera de ellas, y la más importante es clasificar la calidad del vino tinto en tres tipos cualitativos (*malos (calificados de 0 a 4)*, *regulares (calificados de 5 a 7)* y *buenos (los calificados de 8 en adelante)*), debido a que el atributo original con 10 valores diferentes daría lugar a modelos complicados de entender. Para ello, abriremos el archivo en *csv* y modificaremos la columna denominada *quality (calidad)*.

La segunda simplificación tiene que ver con el número de instancias, ya que la base de datos inicial *"no es balanceada"* es decir, el número de instancias de cada uno de los valores correspondientes al atributo calidad es muy diferente al darse pocos casos de algunos tipos. Si queremos realizar de una forma correcta la clasificación de esa clase minoritaria debemos tener unos datos con un número de observaciones similar para cada clase. De no ser así los algoritmos tienden a favorecer la clase con mayor proporción de observaciones pudiendo obtener un modelo que no predice de forma correcta la clase minoritaria.

Seguidamente, pasamos a crear el archivo de datos de tipo *ARFF*. Por tanto, necesitamos abrir el archivo en un bloc de notas y codificarlo en el formato como se explicó en el apartado *Preparación de los datos*. Como resultado tenemos:

3.5.3. Carga del fichero de datos

Codificado el archivo, procedemos a guardarlo, añadiendo *.arff*, y a cargarlo en Weka utilizando el botón *Open file* de la pestaña *Preprocess* disponible en la interfaz **Explorer**. Una vez seleccionado el fichero *winequalitywhite.arff* se mostrará la siguiente pantalla,

Una vez cargados los datos, aparece un cuadro resumen, *Current relation*, con el nombre de la relación que se indica en el fichero, el número de instancias y el número de atributos. Más abajo, aparece una lista con todos los atributos disponibles, con los nom-

```
% Archivo sobre la calidad del vino tinto
@relation winequalitywhite

@attribute fixed_acidity numeric
@attribute volatile_acidity numeric
@attribute citric_acid numeric
@attribute residual_sugar numeric
@attribute chlorides numeric
@attribute free_sulfur numeric
@attribute total_sulfur numeric
@attribute density numeric
@attribute pH numeric
@attribute sulphates numeric
@attribute alcohol numeric
@attribute quality {Malo,Regular,Bueno}

@data
6.2,0.66,0.48,1.2,0.029,29,75,0.9892,3.33,0.39,12.8,Bueno
7.4,0.34,0.42,1.1,0.033,17,171,0.9917,3.12,0.53,11.3,Regular
6.5,0.31,0.14,7.5,0.044,34,133,0.9955,3.22,0.5,9.5,Regular
6.2,0.66,0.48,1.2,0.029,29,75,0.9892,3.33,0.39,12.8,Bueno
6.4,0.31,0.38,2.9,0.038,19,102,0.9912,3.17,0.35,11,Regular
6.8,0.26,0.42,1.7,0.049,41,122,0.993,3.47,0.48,10.5,Bueno
6.2,0.45,0.26,4.4,0.063,63,206,0.994,3.27,0.52,9.8,Malo
6.7,0.23,0.31,2.1,0.046,30,96,0.9926,3.33,0.64,10.7,Bueno
7.4,0.24,0.29,10.1,0.05,21,105,0.9962,3.13,0.35,9.5,Regular
6.2,0.27,0.43,7.8,0.056,48,244,0.9956,3.1,0.51,9,Regular
6.8,0.3,0.23,4.6,0.061,50.5,238.5,0.9958,3.32,0.6,9.5,Regular
6,0.27,0.28,4.8,0.063,31,201,0.9964,3.69,0.71,10,Regular
8.6,0.23,0.46,1,0.054,9,72,0.9941,2.95,0.49,9.1,Regular
6.7,0.23,0.31,2.1,0.046,30,96,0.9926,3.33,0.64,10.7,Bueno
7.4,0.24,0.29,10.1,0.05,21,105,0.9962,3.13,0.35,9.5,Regular
7.1,0.18,0.36,1.4,0.043,31,87,0.9898,3.26,0.37,12.7,Regular
7,0.32,0.34,1.3,0.042,20,69,0.9912,3.31,0.65,12,Regular
7.4,0.18,0.3,8.8,0.064,26,103,0.9961,2.94,0.56,9.3,Regular
6.7,0.54,0.28,5.4,0.06,21,105,0.9949,3.27,0.37,9,Regular
```

Figura 3.1: Archivo de datos *winequalitywhite*

bres especificados en el fichero, de modo que se pueden seleccionar para ver sus detalles y propiedades, apareciendo en la parte derecha las propiedades del atributo seleccionado *(Selected attribute)*. Según el tipo de atributo, se presentará en *Selected attribute*, el número de instancias (atributo nominal) o los valores máximo, mínimo, valor medio y desviación estándar (atributo numérico).

Finalmente, el objetivo que nos planteamos con este trabajo es aplicar los métodos de clasificación descritos anteriormente para el conjunto de datos y estudiar cuales son los factores que caracterizan cada una de las categorías que hemos creado.

3.5.4. Filtros de discretización en los atributos

Estos filtros son muy útiles cuando se trabaja con atributos numéricos, puesto que muchas herramientas de análisis requieren datos simbólicos como es el caso de la aplicación de algoritmos de clasificación. Esta transformacion es necesaria aplicarla antes de ejecu-

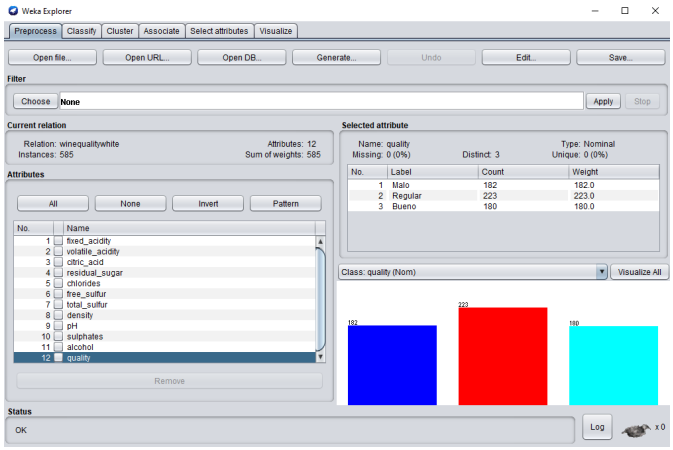


Figura 3.2: Preprocesamiento de datos *winequalitywhite.arff*

tar cualquier algoritmo de clasificación. Este filtrado transforma los atributos numéricos seleccionados en atributos simbólicos, con una serie de etiquetas resultantes de dividir la amplitud total del atributo en intervalos, con diferentes opciones para seleccionar los límites. Para aplicarlo tenemos que pulsar en la pestaña **Preprocess** el botón *Choose* de la sección *Filter*, y seleccionar *Weka/ Filters/ Unsupervised/ attribute/ Discretize*. Una vez seleccionado es necesario pulsar en la etiqueta del filtro para definir los parámetros, obteniendo la ventana siguiente

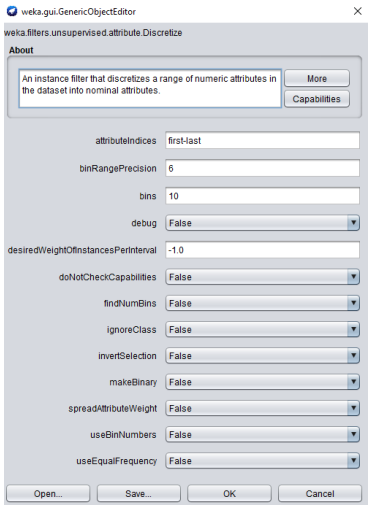


Figura 3.3: Definición del filtro *Discretize*

donde

- *attributeIndices.* Índices de los atributos sobre los que actuará el filtro. Por defecto los indica todos.
- *binRangePrecision.* Indica el rango del intervalo, por defecto 6.
- *bins.* Se divide la amplitud del intervalo en tantas *cajas* como se indique, con la misma amplitud. Por defecto 10.
- *debug.* Si se establece en verdadero, el filtro puede generar información adicional a la consola.
- *desiredWeightOfInstancesPerInterval.* Número de observaciones en cada intervalo.
- *doNotCheckCapabilities.* Si se cambia a verdadero, las capacidades del filtro no se verifican antes de crearse.
- *findNumBins.* Permite optimizar el número de cajas (de la misma amplitud), con un criterio de clasificación de mínimo error en función de las etiquetas.
- *ignoreClass.* Nos permitirá aplicar el filtro al atributo clase.
- *invertSelection.* Establece el modo de selección de atributos. Si es falso, solo los atributos (numéricos) seleccionados en el rango serán discretizados; si es verdadero, solo los atributos no seleccionados serán discretizados.
- *makeBinary.* Hace que los atributos resultantes (nominales) sean binarios (cada uno de ellos tendrá dos posibles valores).
- *spreadAttributeWeight.* Al generar atributos binarios, distribuye el peso del atributo antiguo entre los atributos nuevos.

- *useBinNumbers.* Al cambiar a *True*, permite usar números de bin en lugar de rangos para atributos discretos.

- *useEqualFrequency.* Forzar a una distribución uniforme de instancias por categoría con distinta amplitud.

Definido el filtro, selecionamos *ok* y *Apply*, para realiza el proceso de discretización en los datos; así pues, cada observación de cada una de las variables se introducirá en su correspondiente intervalo. En algunos conjuntos de datos no es necesario aplicar este filtro al no disponer de atributos de tipo numérico, siendo en este caso necesario.

Explicado el proceso de *discretización* pasaremos a ponerlo en práctica en nuestro conjunto de datos, al ser la mayoría de los atributos de tipo numérico. Para ello, definimos el filtro como se muestra en la *Figura 3.4*.

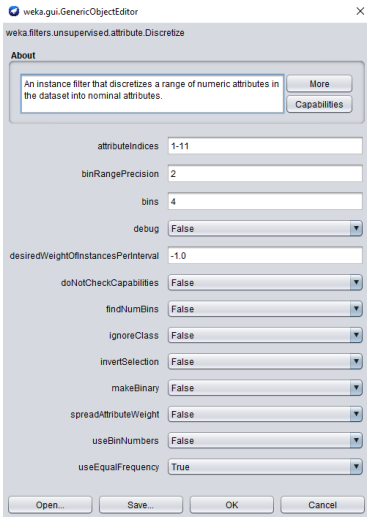


Figura 3.4: Definición del filtro *Discretize*

Podemos observar por la imagen anterior que se fija, un número de 4 cajas *(bins)*, una precisión de 2 decimales **(binRangePrecision)** y los índices de los atributos de tipo numérico **(attributeIndices)**. Además, forzamos a que todos los intervalos tengan la misma frecuencia marcando *True* en **useEqualFrequency**. Seguidamente seleccionamos *ok* y *Apply*.

Aplicado el filtro, podemos observar la subdivisión deseada en el recuadro denominado *Selected attribute*, selecionando cualquier atributo transformado contenido en la sección *Attributes*. También se pueden visualizar los datos transformados seleccionando *Edit (Figura 3.5)*.

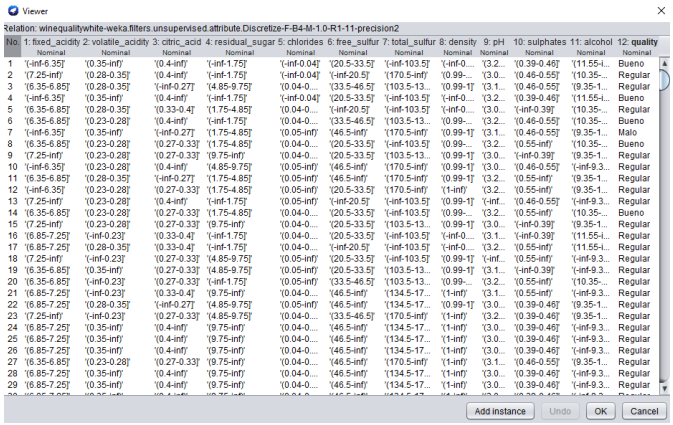


Figura 3.5: Datos transformados

3.5.5. Aplicación de los métodos de clasificación

En esta sección, aplicaremos a nuestro conjunto de datos los clasificadores más usuales comentados anteriormente, en los modos de evaluación descritos : *Use training set*, *Cross-validation* y *Percentil split*. No lo haremos para el caso *Supplied test set* puesto que no poseemos un conjunto independiente sobre el que evaluar los resultados.

1. **Método de clasificación Naïve Bayes**

   Este método, perteneciente a los clasificadores Bayesianos, se basa en una técnica de clasificación estadística llamada *"teorema de Bayes"*. En ellos se supone que las variables predictoras son independientes entre sí. En otras palabras, que la presencia de una cierta característica en un conjunto de datos no está en absoluto relacionada con la presencia de cualquier otra característica. Además involucra una hipótesis de difícil cumplimiento que funciona bien con bases de datos reales; en especial,

cuando se combina con procedimientos de selección de atributos para eliminar la redundancia.

Para su aplicación es necesario acceder a la pestaña **Classify**, situada en la parte superior de la interfaz **Explorer**, y pulsar el botón *Choose* de la sección *Classifier*. Seguidamente seleccionaremos *Weka/ Classifiers/ Bayes/ NaiveBayes*.

Seleccionado este método, podemos aplicarlo directamente en los diferentes métodos de evaluación. Además podemos modificar sus propiedades como sea necesario haciendo click en la etiqueta, en nuestro caso no se modificaran. Así pues tenemos:

- *Use training set (Conjunto de entrenamiento).* Seleccionando la opción *Use training set* en la sección *Test options*, y pulsando *Start*, podemos observar la iformación siguiente:

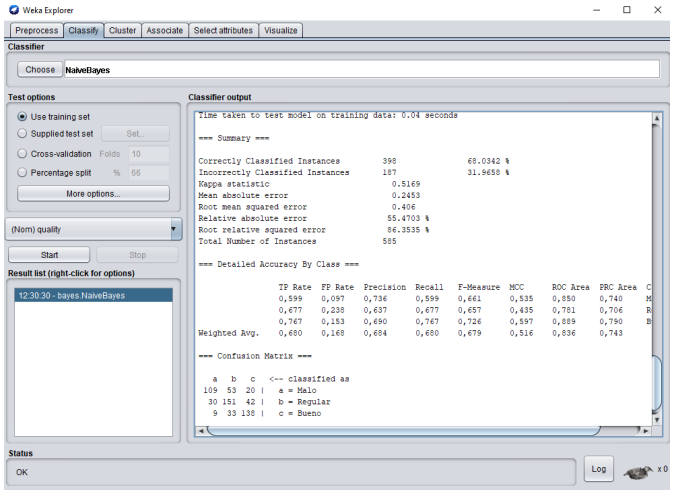


Figura 3.6: Salida de aplicar *Naïve Bayes* usando la opción *Use training set*

A continuación analizaremos la salida resultante de aplicar el clasificador, definiendo los términos que aparecen y que son útilies para la comparación entre los distintos métodos de clasificación aplicados.

En la *Figura 3.6*, vemos que se han clasificado correctamente el 68.0342 % de

las observaciones, mientras que el 31.9658 % lo han hecho de forma incorrecta. Además, se observa que la precisión media por clase es del 0.684.

Seguidamente se muestra el valor del *índice Kappa* (0.5169), indicando que en este caso tenemos una concordancia moderada. Esté índice es una medida de concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Si su valor fuese 1, la concordancia sería perfecta; si fuese 0, la concordancia es debida al azar; y si tomase un valor negativo, la concordancia es menor a la esperada por el azar.

Posteriormente aparecen ciertas medidas asociadas al *error de la clasificación*, cuyos indicadores asociados son:

- *Mean Absolute Error (error absoluto medio):* es la cantidad que se usa para medir las diferencias entre los cálculos previstos y los observados. El algoritmo ha conseguido un 0.2453.
- *Root Mean Squared Error (error cuadrático medio de la raíz):* es un número que permite medir la magnitud media del error, es decir, la diferencia existente entre lo que pronosticamos y sus valores observados. Este valor es siempre mayor o igual al obtenido en *Mean Absolute Error*. El valor aportado por el algoritmo es de 0.406.
- *Relative absolute error (%):* es una forma de medir el rendimiento de un modelo predictivo. Se expresa como una razón, comparando un *error medio (residual)* con los errores producidos por un modelo trivial o ingenuo. El algoritmo ha conseguido un 55.4703 %.
- *Root relative squared error (%):* es relativo a lo que habría sido si se hubiera utilizado un predictor simple. Más específicamente, este predictor simple es solo el promedio de los valores reales. El algoritmo ha conseguido un 86.3535 %.

A continuación, aparece el número total de casos (585) y un cuadro donde se muestran una serie de indicadores relativos a la precisión de la clasificación. Estos coeficientes se basan en los resultados de la matriz de confusión que aparece al final de la salida. En este caso, la matriz de confusión muestra la siguiente información:

```
=== Confusion Matrix ===

   a   b   c   <-- classified as
 109  53  20 |   a = Malo
  30 151  42 |   b = Regular
   9  33 138 |   c = Bueno
```

Figura 3.7: Matriz de confusión

Por esta matriz, podemos observar que de los 585 casos han sido correctamente clasificados: 109 en el grupo *Malo*, 151 en el grupo *Regular* y 138 en el grupo *Bueno*.

También por estos valores podemos extraer para cada clase los siguientes indicadores de precisión: la **Tasa de verdaderos positivos**, la **Tasa de falsos positivos**, la **Precisión**, **Recall** y la **Medida-F**. Estos valores aparecen en la salida de resultados del algoritmo *(Figura 3.8)*.

```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,599    0,097    0,736      0,599    0,661      0,535    0,850     0,740     Malo
               0,677    0,238    0,637      0,677    0,657      0,435    0,781     0,706     Regular
               0,767    0,153    0,690      0,767    0,726      0,597    0,889     0,790     Bueno
Weighted Avg.  0,680    0,168    0,684      0,680    0,679      0,516    0,836     0,743
```

Figura 3.8: Indicadores de precisión

- *Mejora de precisión del clasificador (Filtrado de atributos).*

  Destacar de este algoritmo de clasificación, que pierde precisión rápidamente cuando se añaden características con una correlación baja, es decir, se reduce la

calidad ante la presencia de atributos no relevantes. Al utilizar este algoritmo, con un modo de evaluación mediante un conjunto de entrenamiento, hemos obtenido una precisión del 68.0342 %. Por tanto pasaremos a comprobar si los atributos no relevantes están afectando a la calidad del método, efectuando un filtrado de atributos. Para ello, pasamos a la sección *Select Attributes* donde encontramos dos familias de técnicas para realizar este proceso: *Métodos de filtros*, donde se seleccionan y evalúan los atributos en forma independiente del algoritmo de clasificación; y *Wrappers*, donde se procede a la ejecución de algún clasificador para determinar lo deseable de un subconjunto.

Dadas las características del problema, en este caso podemos probar con una técnica *Wrapper* realizando una búsqueda exhaustiva. Para ello, pulsamos *Choose* en la sección *Attribute Evaluator* y seleccionamos el método *WrapperSubsetEval*. Para configurarlo pulsamos en la ventana de texto. Además vamos a utilizar el propio **Naive Bayes** para el *Wrapper*, teniendo que seleccionar este método en *Classifier*. Por otra parte en *Search Method*, indicamos que queremos una búsqueda exhaustiva eligiendo *ExhaustiveSearch*. De esta manera tendremos la herramienta configurada tal y como aparece en la *Figura 3.9*.

A continuación ejecutamos el método de filtrado pulsando *Start*, observando que este método de filtraje nos recomienda usar todos los atributos. Por tanto, en este caso no sería necesario volver a la pestaña *Preprocess* y eliminar los atributos descartados seleccionando *Weka/ Filters/ Unsupervised/ attribute/ Remove*, ni volver a ejecutar el método **NaiveBayes**, usando un conjunto de entrenamiento.

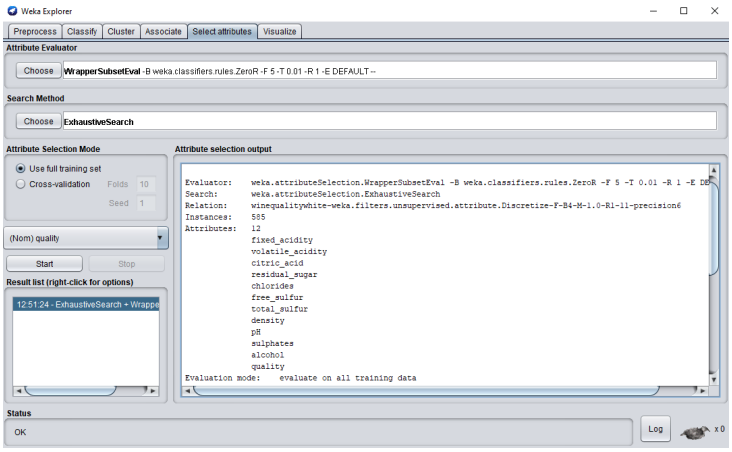


Figura 3.9: Filtrado de atributos

- *Cross-validation (Validación cruzada).* En este caso todos atributos son necesarios para la clasificación y no se puede eliminar niguno, pero si hubiese sido el caso contrario, habría que suponer que hemos eliminado aquellos atributos que disminuyen la precisión de los estimadores de igual forma que lo hemos hecho en el apartado anterior. A continuación, seleccionando la opción *Cross-validation* en la sección *Test options*, y pulsando *Start*, podemos observar la iformación siguiente:

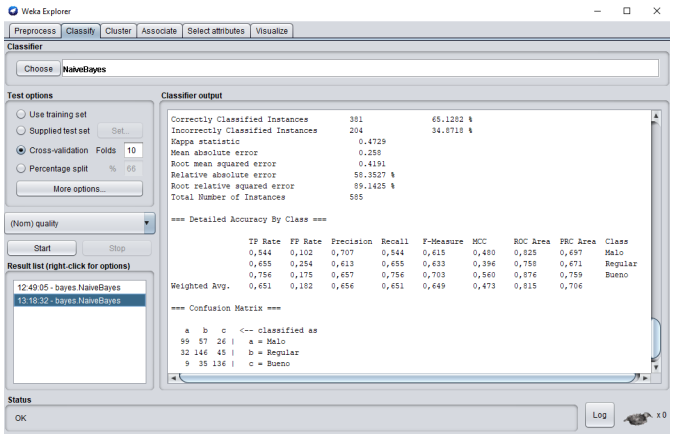


Figura 3.10: Salida de aplicar *Naïve Bayes* usando la opción *Cross-validation*

A continuación analizaremos la salida resultante de aplicar el clasificador, de-

finiendo los términos que aparecen y que son útilies para la comparación entre los distintos métodos de clasificación aplicados.

En la *Figura 3.10*, vemos que se han clasificado correctamente el 65.1282 % de las observaciones, mientras que el 34.8718 % lo han hecho de forma incorrecta. Además, se observa que la precisión media por clase es del 0.656.

Seguidamente se muestra el valor del *índice Kappa* (0.4729), indicando que en este caso tenemos una concordancia moderada. Esté índice es una medida de concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Si su valor fuese 1, la concordancia sería perfecta; si fuese 0, la concordancia es debida al azar; y si tomase un valor negativo, la concordancia es menor a la esperada por el azar.

Posteriormente aparecen ciertas medidas asociadas al *error de la clasificación*, cuyos indicadores asociados son:

- *Mean Absolute Error (error absoluto medio):* es la cantidad que se usa para medir las diferencias entre los cálculos previstos y los observados. El algoritmo ha conseguido un 0.258.
- *Root Mean Squared Error (error cuadrático medio de la raíz):* es un número que permite medir la magnitud media del error, es decir, la diferencia existente entre lo que pronosticamos y sus valores observados. Este valor es siempre mayor o igual al obtenido en *Mean Absolute Error*. El valor aportado por el algoritmo es de 0.4191.
- *Relative absolute error (%):* es una forma de medir el rendimiento de un modelo predictivo. Se expresa como una razón, comparando un *error medio (residual)* con los errores producidos por un modelo trivial o ingenuo. El algoritmo ha conseguido un 58.3527 %.
- *Root relative squared error (%):* es relativo a lo que habría sido si se hubiera

utilizado un predictor simple. Más específicamente, este predictor simple es solo el promedio de los valores reales. El algoritmo ha conseguido un error alto, en concreto 89.1425 %.

A continuación, aparecen el número total de casos (585) y un cuadro donde se muestran una serie de indicadores relativos a la precisión de la clasificación. Estos coeficientes se basan en los resultados de la matriz de confusión que aparece al final de la salida. En este caso, la matriz de confusión muestra la siguiente información:

```
=== Confusion Matrix ===

   a   b   c   <-- classified as
  99  57  26 |   a = Malo
  32 146  45 |   b = Regular
   9  35 136 |   c = Bueno
```

Figura 3.11: Matriz de confusión

Por esta matriz, podemos observar que de los 585 casos han sido correctamente clasificados: 99 en el grupo *Malo*, 146 en el grupo *Regular* y 136 en el grupo *Bueno*.

También por estos valores podemos extraer para cada clase los siguientes indicadores de precisión: la **Tasa de verdaderos positivos**, la **Tasa de falsos positivos**, la **Precisión**, **Recall** y la **Medida-F**. Estos valores aparecen en la salida de resultados del algoritmo *(Figura 3.12)*.

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,544    0,102    0,707      0,544    0,615      0,480    0,825     0,697     Malo
                 0,655    0,254    0,613      0,655    0,633      0,396    0,758     0,671     Regular
                 0,756    0,175    0,657      0,756    0,703      0,560    0,876     0,759     Bueno
Weighted Avg.    0,651    0,182    0,656      0,651    0,649      0,473    0,815     0,706
```

Figura 3.12: Indicadores de precisión

- *Percentage split (Dividiendo el fichero de datos).* En este caso todos atributos son necesarios para la clasificación y no se puede eliminar ninguno, pero si hubiese sido el caso contrario, habría que suponer que hemos eliminado aquellos atributos que disminuyen la precisión de los estimadores de igual forma que lo hemos hecho en el apartado anterior. A continuación, seleccionando la opción *Percentage split* en la sección *Test options*. A través de este método el fichero de datos se divide en dos partes una se usa para construir el clasificador y la otra para evaluar su rendimiento, por defecto se indica un 66 % para construir el modelo. Pulsando *Start*, podemos observar la iformación siguiente:

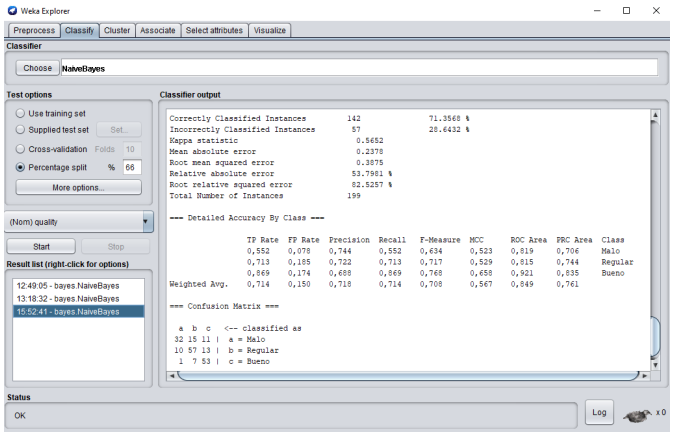


Figura 3.13: Salida de aplicar *Naïve Bayes* usando la opción *Percentage split*

A continuación analizaremos la salida resultante de aplicar el clasificador, definiendo los términos que aparecen y que son útilies para la comparación entre los distintos métodos de clasificación aplicados.

En la *Figura 3.13*, vemos que se han clasificado correctamente el 71.3568 % de las observaciones, mientras que el 28.6432 % lo han hecho de forma incorrecta. Además, se observa que la precisión media por clase es del 0.718.

Seguidamente se muestra el valor del *índice Kappa* (0.5652), indicando que en este caso tenemos una concordancia moderada. Esté índice es una medida de

concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Si su valor fuese 1, la concordancia sería perfecta; si fuese 0, la concordancia es debida al azar; y si tomase un valor negativo, la concordancia es menor a la esperada por el azar.

Posteriormente aparecen ciertas medidas asociadas al *error de la clasificación*, cuyos indicadores asociados son:

- *Mean Absolute Error (error absoluto medio):* es la cantidad que se usa para medir las diferencias entre los cálculos previstos y los observados. El algoritmo ha conseguido un 0.2378.
- *Root Mean Squared Error (error cuadrático medio de la raíz):* es un número que permite medir la magnitud media del error, es decir, la diferencia existente entre lo que pronosticamos y sus valores observados. Este valor es siempre mayor o igual al obtenido en *Mean Absolute Error*. El valor aportado por el algoritmo es de 0.3875.
- *Relative absolute error (%):* es una forma de medir el rendimiento de un modelo predictivo. Se expresa como una razón, comparando un *error medio (residual)* con los errores producidos por un modelo trivial o ingenuo. El algoritmo ha conseguido un 53.7981 %.
- *Root relative squared error (%):* es relativo a lo que habría sido si se hubiera utilizado un predictor simple. Más específicamente, este predictor simple es solo el promedio de los valores reales. El algoritmo ha conseguido un 82.5257 %.

A continuación, aparecen el número total de casos y un cuadro donde se muestran una serie de indicadores relativos a la precisión de la clasificación. Estos coeficientes se basan en los resultados de la matriz de confusión que aparece al final de la salida. En este caso, la matriz de confusión muestra la siguiente

información:

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 32 15 11 |  a = Malo
 10 57 13 |  b = Regular
  1  7 53 |  c = Bueno
```

Figura 3.14: Matriz de confusión

Por esta matriz, podemos observar que de los 199 casos han sido correctamente clasificados: 32 en el grupo *Malo*, 57 en el grupo *Regular* y 53 en el grupo *Bueno*.

También por estos valores podemos extraer para cada clase los siguientes indicadores de precisión: la **Tasa de verdaderos positivos**, la **Tasa de falsos positivos**, la **Precisión**, **Recall** y la **Medida-F**. Estos valores aparecen en la salida de resultados del algoritmo *(Figura 3.15)*.

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,552    0,078    0,744      0,552    0,634      0,523    0,819     0,706     Malo
                0,713    0,185    0,722      0,713    0,717      0,529    0,815     0,744     Regular
                0,869    0,174    0,688      0,869    0,768      0,658    0,921     0,835     Bueno
Weighted Avg.   0,714    0,150    0,718      0,714    0,708      0,567    0,849     0,761
```

Figura 3.15: Indicadores de precisión

2. **Método de clasificación Stacking**

   Esto método, perteneciente a los meta-clasificadores, es un procedimiento general para ensamblar modelos base (también conocidos como modelos de primer nivel) a través de entrenar un meta-modelo (o modelo de segundo nivel) que tendrá como objetivo encontrar una manera óptima de combinar la salida de los modelos individuales. Como cada uno de los modelos aprende a través de un mecanismo de aprendizaje diferente, se logra que los modelos del conjunto sean distintos.

Por tanto definiremos como calsificadores base *Naïve Bayes*, *OneR* y *J48*, utilizando el clasificador *J48* como meta-clasificador para el conjunto de datos discretizado.

Para su aplicación es necesario acceder a la pestaña **Classify**, situada en la parte superior de la interfaz **Explorer**, y pulsar el botón *Choose* de la sección *Classifier*. Seguidamente seleccionaremos *Weka/Classifiers/ Meta/ Stacking*, definimos los parámetros del algoritmo en función a lo descrito en el párrafo anterior, como se muestra en la *Figura 3.16*, y pulsaremos *ok*.

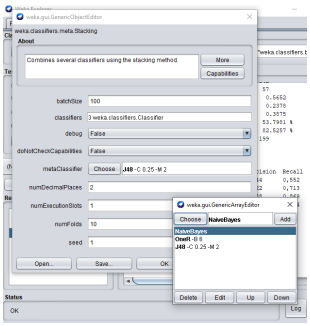


Figura 3.16: Selección de los clasificadores base y meta clasificador en el algoritmo *Stacking*

Realizados los cambios necesarios, podemos aplicarlo en los diferentes métodos de evaluación. Así pues tenemos:

- *Use training set (Conjunto de entrenamiento).* Seleccionando la opción *Use training set* en la sección *Test options*, y pulsando *Start*, se muestran los modelos inducidos para cada clasificador individual y para el modelo aprendido por el meta-clasificador. La salida se puede observar en la *Figura 3.17*.

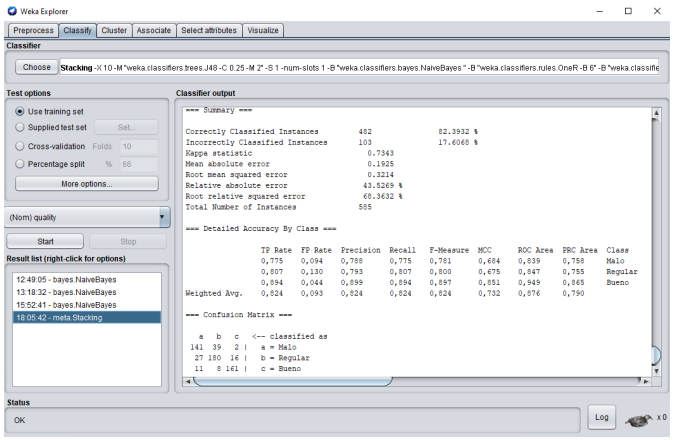


Figura 3.17: Salida de aplicar *Stacking* usando la opción *Use training set*

A continuación analizaremos la salida resultante de aplicar el clasificador, definiendo los términos que aparecen y que son útilies para la comparación entre los distintos métodos de clasificación aplicados.

En la *Figura 3.17*, vemos que se han clasificado correctamente el 82.3932 % de las observaciones, mientras que el 17.6068 % lo han hecho de forma incorrecta. Además, se observa que la precisión media por clase es del 0.824.

Seguidamente se muestra el valor del *índice Kappa* (0.7343), indicando que en este caso tenemos una concordancia cosiderada. Esté índice es una medida de concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Si su valor fuese 1, la concordancia sería perfecta; si fuese 0, la concordancia es debida al azar; y si tomase un valor negativo, la concordancia es menor a la esperada por el azar.

Posteriormente aparecen ciertas medidas asociadas al *error de la clasificación*, cuyos indicadores asociados son:

- *Mean Absolute Error (error absoluto medio):* es la cantidad que se usa para medir las diferencias entre los cálculos previstos y los observados. El

algoritmo ha conseguido un 0.1925.

- *Root Mean Squared Error (error cuadrático medio de la raíz):* es un número que permite medir la magnitud media del error, es decir, la diferencia existente entre lo que pronosticamos y sus valores observados. Este valor es siempre mayor o igual al obtenido en *Mean Absolute Error*. El valor aportado por el algoritmo es de 0.3214.
- *Relative absolute error (%):* es una forma de medir el rendimiento de un modelo predictivo. Se expresa como una razón, comparando un *error medio (residual)* con los errores producidos por un modelo trivial o ingenuo. El algoritmo ha conseguido un 43.5269 %.
- *Root relative squared error (%):* es relativo a lo que habría sido si se hubiera utilizado un predictor simple. Más específicamente, este predictor simple es solo el promedio de los valores reales. El algoritmo ha conseguido un 68.3632 %.

A continuación, aparecen el número total de casos (585) y un cuadro donde se muestran una serie de indicadores relativos a la precisión de la clasificación. Estos coeficientes se basan en los resultados de la matriz de confusión que aparece al final de la salida. En este caso, la matriz de confusión muestra la siguiente información:

```
=== Confusion Matrix ===

   a   b   c   <-- classified as
 141  39   2 |   a = Malo
  27 180  16 |   b = Regular
  11   8 161 |   c = Bueno
```

Figura 3.18: Matriz de confusión

Por esta matriz, podemos observar que de los 585 han sido correctamente clasificados: 141 en el grupo *Malo*, 180 en el grupo *Regular* y 161 en el grupo

Bueno.

También por estos valores podemos extraer para cada clase los siguientes indicadores de precisión: la **Tasa de verdaderos positivos**, la **Tasa de falsos positivos**, la **Precisión**, **Recall** y la **Medida-F**. Estos valores aparecen en la salida de resultados del algoritmo *(Figura 3.19)*.

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,775    0,094    0,788      0,775    0,781      0,684    0,839     0,758     Malo
                 0,807    0,130    0,793      0,807    0,800      0,675    0,847     0,755     Regular
                 0,894    0,044    0,899      0,894    0,897      0,851    0,949     0,865     Bueno
Weighted Avg.    0,824    0,093    0,824      0,824    0,824      0,732    0,876     0,790
```

Figura 3.19: Indicadores de precisión

- *Cross-validation (Validación cruzada).* Seleccionando la opción *Cross-validation* en la sección *Test options*, y pulsando *Start*, se muestran los modelos inducidos para cada clasificador individual y para el modelo aprendido por el meta-clasificador. La salida se puede observar en la *Figura 3.20*.

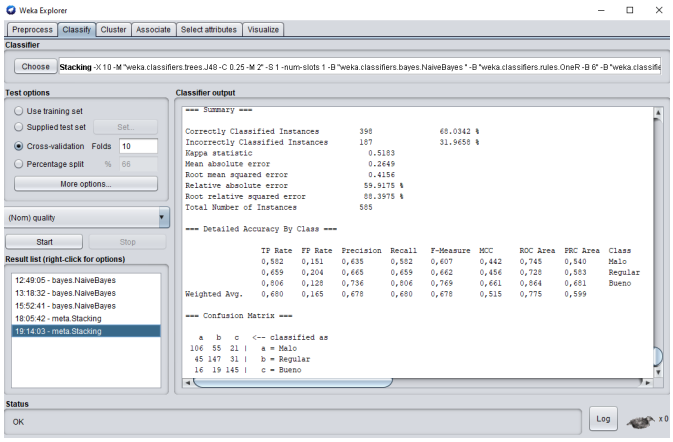


Figura 3.20: Salida de aplicar *Stacking* usando la opción *Cross-validation*

A continuación analizaremos la salida resultante de aplicar el clasificador, definiendo los términos que aparecen y que son útilies para la comparación entre los distintos métodos de clasificación aplicados.

En la *Figura 3.20*, vemos que se han clasificado correctamente el 68.0342 % de las observaciones, mientras que el 31.9658 % lo han hecho de forma incorrecta. Además, se observa que la precisión media por clase es del 0.678.

Seguidamente se muestra el valor del *índice Kappa* (0.5183), indicando que en este caso tenemos una concordancia moderada. Esté índice es una medida de concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Si su valor fuese 1, la concordancia sería perfecta; si fuese 0, la concordancia es debida al azar; y si tomase un valor negativo, la concordancia es menor a la esperada por el azar.

Posteriormente aparecen ciertas medidas asociadas al *error de la clasificación*, cuyos indicadores asociados son:

- *Mean Absolute Error (error absoluto medio):* es la cantidad que se usa para medir las diferencias entre los cálculos previstos y los observados. El algoritmo ha conseguido un 0.2649.
- *Root Mean Squared Error (error cuadrático medio de la raíz):* es un número que permite medir la magnitud media del error, es decir, la diferencia existente entre lo que pronosticamos y sus valores observados. Este valor es siempre mayor o igual al obtenido en *Mean Absolute Error*. El valor aportado por el algoritmo es de 0.4156.
- *Relative absolute error (%):* es una forma de medir el rendimiento de un modelo predictivo. Se expresa como una razón, comparando un *error medio (residual)* con los errores producidos por un modelo trivial o ingenuo. El algoritmo ha conseguido un 59.9175 %.
- *Root relative squared error (%):* es relativo a lo que habría sido si se hubiera utilizado un predictor simple. Más específicamente, este predictor simple es solo el promedio de los valores reales. El algoritmo ha conseguido un

88.3975 %.

A continuación, aparecen el número total de casos (585) y un cuadro donde se muestran una serie de indicadores relativos a la precisión de la clasificación. Estos coeficientes se basan en los resultados de la matriz de confusión que aparece al final de la salida. En este caso, la matriz de confusión muestra la siguiente información:

```
=== Confusion Matrix ===

   a   b   c   <-- classified as
 106  55  21 |   a = Malo
  45 147  31 |   b = Regular
  16  19 145 |   c = Bueno
```

Figura 3.21: Matriz de confusión

Por esta matriz, podemos observar que de los 585 casos han sido correctamente clasificados: 106 en el grupo *Malo*, 147 en el grupo *Regular* y 145 en el grupo *Bueno*.

También por estos valores podemos extraer para cada clase los siguientes indicadores de precisión: la **Tasa de verdaderos positivos**, la **Tasa de falsos positivos**, la **Precisión**, **Recall** y la **Medida-F**. Estos valores aparecen en la salida de resultados del algoritmo *(Figura 3.22)*.

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,582    0,151    0,635      0,582    0,607      0,442    0,745     0,540     Malo
                0,659    0,204    0,665      0,659    0,662      0,456    0,728     0,583     Regular
                0,806    0,128    0,736      0,806    0,769      0,661    0,864     0,681     Bueno
Weighted Avg.   0,680    0,165    0,678      0,680    0,678      0,515    0,775     0,599
```

Figura 3.22: Indicadores de precisión

- *Percentage split (Dividiendo el fichero de datos).* Seleccionamos la opción *Percentage split* en la sección *Test options*, y pulsando *Start*. A través de este

método, el fichero de datos se divide en dos partes, de acuerdo al porcentaje indicado (que dejaremos en el 66 %). La primera se utilizará para construir el clasificador y la segunda para evaluar su rendimiento. La salida se puede observar en la *Figura 3.23*.

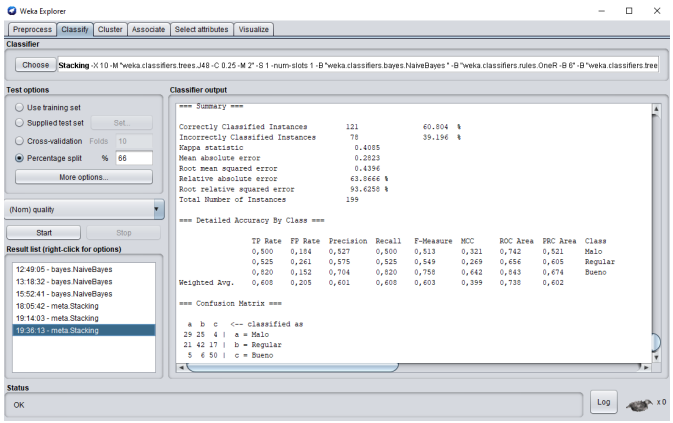


Figura 3.23: Salida de aplicar *Stacking* usando la opción *Percentage split*

A continuación analizaremos la salida resultante de aplicar el clasificador, definiendo los términos que aparecen y que son útilies para la comparación entre los distintos métodos de clasificación aplicados.

En la *Figura 3.23*, vemos que se han clasificado correctamente el 60.804 % de las observaciones, mientras que el 39.196 % lo han hecho de forma incorrecta. Además, se observa que la precisión media por clase es del 0.601.

Seguidamente se muestra el valor del *índice Kappa* (0.4085), indicando que en este caso tenemos una concordancia moderada. Esté índice es una medida de concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Si su valor fuese 1, la concordancia sería perfecta; si fuese 0, la concordancia es debida al azar; y si tomase un valor negativo, la concordancia es menor a la esperada por el azar.

Posteriormente aparecen ciertas medidas asociadas al *error de la clasificación*, cuyos indicadores asociados son:

- *Mean Absolute Error (error absoluto medio):* es la cantidad que se usa para medir las diferencias entre los cálculos previstos y los observados. El algoritmo ha conseguido un 0.2823.
- *Root Mean Squared Error (error cuadrático medio de la raíz):* es un número que permite medir la magnitud media del error, es decir, la diferencia existente entre lo que pronosticamos y sus valores observados. Este valor es siempre mayor o igual al obtenido en *Mean Absolute Error*. El valor aportado por el algoritmo es de 0.4396.
- *Relative absolute error (%):* es una forma de medir el rendimiento de un modelo predictivo. Se expresa como una razón, comparando un *error medio (residual)* con los errores producidos por un modelo trivial o ingenuo. El algoritmo ha conseguido un 63.8666 %.
- *Root relative squared error (%):* es relativo a lo que habría sido si se hubiera utilizado un predictor simple. Más específicamente, este predictor simple es solo el promedio de los valores reales. El algoritmo ha conseguido un 93.6258 %.

A continuación, aparecen el número total de casos y un cuadro donde se muestran una serie de indicadores relativos a la precisión de la clasificación. Estos coeficientes se basan en los resultados de la matriz de confusión que aparece al final de la salida. En este caso, la matriz de confusión muestra la siguiente información:

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 29 25  4 |  a = Malo
 21 42 17 |  b = Regular
  5  6 50 |  c = Bueno
```

Figura 3.24: Matriz de confusión

Por esta matriz, podemos observar que de los 199 casos han sido correctamente clasificados: 29 en el grupo *Malo*, 42 en el grupo *Regular* y 50 en el grupo *Bueno*.

También por estos valores podemos extraer para cada clase los siguientes indicadores de precisión: la **Tasa de verdaderos positivos**, la **Tasa de falsos positivos**, la **Precisión**, **Recall** y la **Medida-F**. Estos valores aparecen en la salida de resultados del algoritmo *(Figura 3.25)*.

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,500    0,184    0,527      0,500    0,513      0,321    0,742     0,521     Malo
                0,525    0,261    0,575      0,525    0,549      0,269    0,656     0,605     Regular
                0,820    0,152    0,704      0,820    0,758      0,642    0,843     0,674     Bueno
Weighted Avg.   0,608    0,205    0,601      0,608    0,603      0,399    0,738     0,602
```

Figura 3.25: Indicadores de precisión

3. **Método de clasificación OneR**

Este método, perteneciente a *One Rules*, es un algoritmo de clasificación que genera un árbol de decisión de un único nivel, cuya sencillez y rapidez puede lograr resultados muchos más complejos en comparación con otros métodos. Simplemente selecciona el atributo que mejor *"explica"* la clase de salida. Si hay atributos numéricos, busca los umbrales para hacer reglas con mejor tasa de aciertos.

Para su aplicación es necesario acceder a lapestaña **Classify**, situada en la parte superior de la interfaz **Explorer**, y pulsar el botón *Choose* de la sección *Classifier*. Seguidamente seleccionamos *Weka/ Classifiers/ Rules/ OneR*.

Seleccionado este método, podemos aplicarlo directamente en los diferentes métodos de evaluación. Además podemos modificar sus propiedades como sea necesario haciendo click en la etiqueta, en nuestro caso no se modificaran. Así pues tenemos:

- *Use training set (Conjunto de entrenamiento).* Seleccionando la opción *Use training set* en la sección *Test options*, y pulsando *Start*, podemos observar la iformación siguiente:

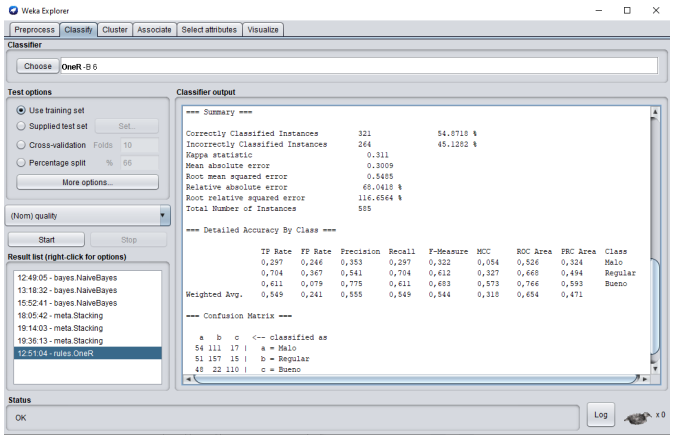


Figura 3.26: Salida de aplicar *OneR* usando la opción *Use training set*

A continuación analizaremos la salida resultante de aplicar el clasificador, definiendo los términos que aparecen y que son útilies para la comparación entre los distintos métodos de clasificación aplicados.

En la *Figura 3.26*, vemos que se han clasificado correctamente el 54.8718 % de las observaciones, mientras que el 45.1282 % lo han hecho de forma incorrecta. Además, se observa que la precisión media por clase es del 0.555.

Seguidamente se muestra el valor del *índice Kappa* (0.311), indicando que en este caso tenemos una concordancia mediana. Esté índice es una medida de concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Si su valor fuese 1, la concordancia sería perfecta; si fuese 0, la concor-

dancia es debida al azar; y si tomase un valor negativo, la concordancia es menor a la esperada por el azar.

Posteriormente aparecen ciertas medidas asociadas al *error de la clasificación*, cuyos indicadores asociados son:

- *Mean Absolute Error (error absoluto medio):* es la cantidad que se usa para medir las diferencias entre los cálculos previstos y los observados. El algoritmo ha conseguido un 0.3009.
- *Root Mean Squared Error (error cuadrático medio de la raíz):* es un número que permite medir la magnitud media del error, es decir, la diferencia existente entre lo que pronosticamos y sus valores observados. Este valor es siempre mayor o igual al obtenido en *Mean Absolute Error*. El valor aportado por el algoritmo es de 0.5485.
- *Relative absolute error (%):* es una forma de medir el rendimiento de un modelo predictivo. Se expresa como una razón, comparando un *error medio (residual)* con los errores producidos por un modelo trivial o ingenuo. El algoritmo ha conseguido un 68.0418 %.
- *Root relative squared error (%):* es relativo a lo que habría sido si se hubiera utilizado un predictor simple. Más específicamente, este predictor simple es solo el promedio de los valores reales. El algoritmo ha conseguido un 116.6564 %.

A continuación, aparecen el número total de casos (585) y un cuadro donde se muestran una serie de indicadores relativos a la precisión de la clasificación. Estos coeficientes se basan en los resultados de la matriz de confusión que aparece al final de la salida. En este caso, la matriz de confusión muestra la siguiente información:

Por esta matriz, podemos observar que de los 585 casos han sido correctamente

```
=== Confusion Matrix ===

   a   b   c   <-- classified as
  54 111  17 |   a = Malo
  51 157  15 |   b = Regular
  48  22 110 |   c = Bueno
```

Figura 3.27: Matriz de confusión

clasificados: 54 en el grupo *Malo*, 157 en el grupo *Regular* y 110 en el grupo *Bueno*.

También por estos valores podemos extraer para cada clase los siguientes indicadores de precisión: la **Tasa de verdaderos positivos**, la **Tasa de falsos positivos**, la **Precisión**, **Recall** y la **Medida-F**. Estos valores aparecen en la salida de resultados del algoritmo *(Figura 3.28)*.

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,297    0,246    0,353      0,297    0,322      0,054    0,526     0,324     Malo
                0,704    0,367    0,541      0,704    0,612      0,327    0,668     0,494     Regular
                0,611    0,079    0,775      0,611    0,683      0,573    0,766     0,593     Bueno
Weighted Avg.   0,549    0,241    0,555      0,549    0,544      0,318    0,654     0,471
```

Figura 3.28: Indicadores de precisión

- *Cross-validation (Validación cruzada).* Seleccionando la opción *Cross-validation* en la sección *Test options*, y pulsando *Start*, podemos observar la iformación siguiente:

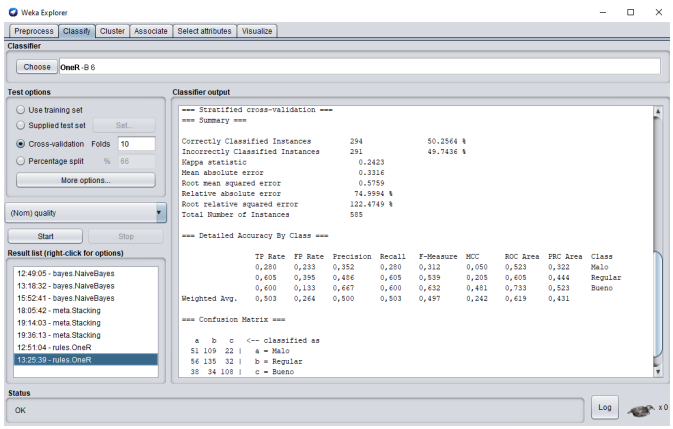


Figura 3.29: Salida de aplicar *OneR* usando la opción *Cross-validation*

A continuación analizaremos la salida resultante de aplicar el clasificador, definiendo los términos que aparecen y que son útilies para la comparación entre los distintos métodos de clasificación aplicados.

En la *Figura 3.29*, vemos que se han clasificado correctamente el 50.2564 % de las observaciones, mientras que el 49.7436 % lo han hecho de forma incorrecta. Además, se observa que la precisión media por clase es del 0.5.

Seguidamente se muestra el valor del *índice Kappa* (0.2423), indicando que en este caso tenemos una concordancia mediana. Esté índice es una medida de concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Si su valor fuese 1, la concordancia sería perfecta; si fuese 0, la concordancia es debida al azar; y si tomase un valor negativo, la concordancia es menor a la esperada por el azar.

Posteriormente aparecen ciertas medidas asociadas al *error de la clasificación*, cuyos indicadores asociados son:

- *Mean Absolute Error (error absoluto medio):* es la cantidad que se usa para medir las diferencias entre los cálculos previstos y los observados. El

algoritmo ha conseguido un 0.3316.

- *Root Mean Squared Error (error cuadrático medio de la raíz):* es un número que permite medir la magnitud media del error, es decir, la diferencia existente entre lo que pronosticamos y sus valores observados. Este valor es siempre mayor o igual al obtenido en *Mean Absolute Error*. El valor aportado por el algoritmo es de 0.5759.
- *Relative absolute error (%):* es una forma de medir el rendimiento de un modelo predictivo. Se expresa como una razón, comparando un *error medio (residual)* con los errores producidos por un modelo trivial o ingenuo. El algoritmo ha conseguido un 74.99994 %.
- *Root relative squared error (%):* es relativo a lo que habría sido si se hubiera utilizado un predictor simple. Más específicamente, este predictor simple es solo el promedio de los valores reales. El algoritmo ha conseguido un 122.4749 %.

A continuación, aparecen el número total de casos (585) y un cuadro donde se muestran una serie de indicadores relativos a la precisión de la clasificación. Estos coeficientes se basan en los resultados de la matriz de confusión que aparece al final de la salida. En este caso, la matriz de confusión muestra la siguiente información:

```
=== Confusion Matrix ===

  a   b   c   <-- classified as
 51 109  22 |   a = Malo
 56 135  32 |   b = Regular
 38  34 108 |   c = Bueno
```

Figura 3.30: Matriz de confusión

Por esta matriz, podemos observar que de los 585 casos han sido correctamente clasificados: 51 en el grupo *Malo*, 135 en el grupo *Regular* y 108 en el grupo

Bueno.

También por estos valores podemos extraer para cada clase los siguientes indicadores de precisión: la **Tasa de verdaderos positivos**, la **Tasa de falsos positivos**, la **Precisión**, **Recall** y la **Medida-F**. Estos valores aparecen en la salida de resultados del algoritmo *(Figura 3.31)*.

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,280    0,233    0,352      0,280    0,312      0,050    0,523     0,322     Malo
                 0,605    0,395    0,486      0,605    0,539      0,205    0,605     0,444     Regular
                 0,600    0,133    0,667      0,600    0,632      0,481    0,733     0,523     Bueno
Weighted Avg.    0,503    0,264    0,500      0,503    0,497      0,242    0,619     0,431
```

Figura 3.31: Indicadores de precisión

- *Percentage split (Dividiendo el fichero de datos).* Seleccionando la opción *Percentage split* en la sección *Test options*, y pulsando *Start*. A través de este método, el fichero de datos se divide en dos partes, de acuerdo al porcentaje indicado (que dejaremos en el 66 %). La primera se utilizará para construir el clasificador y la segunda para evaluar su rendimiento. La salida se puede observar en la *Figura 3.32*.

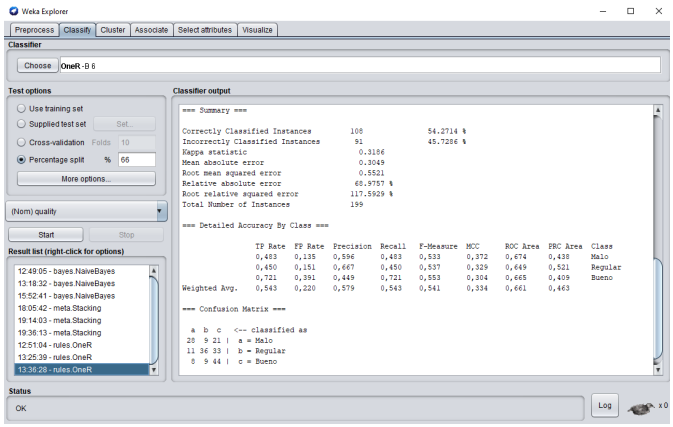


Figura 3.32: Salida de aplicar *OneR* usando la opción *Percentage split*

A continuación analizaremos la salida resultante de aplicar el clasificador, definiendo los términos que aparecen y que son útilies para la comparación entre los distintos métodos de clasificación aplicados.

En la *Figura 3.32*, vemos que se han clasificado correctamente el 54.2714 % de las observaciones, mientras que el 45.7286 % lo han hecho de forma incorrecta. Además, se observa que la precisión media por clase es del 0.579.

Seguidamente se muestra el valor del *índice Kappa* (0.3186), indicando que en este caso tenemos una concordancia mediana. Esté índice es una medida de concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Si su valor fuese 1, la concordancia sería perfecta; si fuese 0, la concordancia es debida al azar; y si tomase un valor negativo, la concordancia es menor a la esperada por el azar.

Posteriormente aparecen ciertas medidas asociadas al *error de la clasificación*, cuyos indicadores asociados son:

- *Mean Absolute Error (error absoluto medio):* es la cantidad que se usa para medir las diferencias entre los cálculos previstos y los observados. El algoritmo ha conseguido un 0.3049.
- *Root Mean Squared Error (error cuadrático medio de la raíz):* es un número que permite medir la magnitud media del error, es decir, la diferencia existente entre lo que pronosticamos y sus valores observados. Este valor es siempre mayor o igual al obtenido en *Mean Absolute Error*. El valor aportado por el algoritmo es de 0.5521.
- *Relative absolute error (%):* es una forma de medir el rendimiento de un modelo predictivo. Se expresa como una razón, comparando un *error medio (residual)* con los errores producidos por un modelo trivial o ingenuo. El algoritmo ha conseguido un 68.9757 %.

- *Root relative squared error (%):* es relativo a lo que habría sido si se hubiera utilizado un predictor simple. Más específicamente, este predictor simple es solo el promedio de los valores reales. El algoritmo ha conseguido un 117.5929 %.

A continuación, aparecen el número total de casos y un cuadro donde se muestran una serie de indicadores relativos a la precisión de la clasificación. Estos coeficientes se basan en los resultados de la matriz de confusión que aparece al final de la salida. En este caso, la matriz de confusión muestra la siguiente información:

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 28  9 21 |  a = Malo
 11 36 33 |  b = Regular
  8  9 44 |  c = Bueno
```

Figura 3.33: Matriz de confusión

Por esta matriz, podemos observar que de los 199 casos han sido correctamente clasificados: 28 en el grupo *Malo*, 36 en el grupo *Regular* y 44 en el grupo *Bueno*. También por estos valores podemos extraer para cada clase los siguientes indicadores de precisión: la **Tasa de verdaderos positivos**, la **Tasa de falsos positivos**, la **Precisión**, **Recall** y la **Medida-F**. Estos valores aparecen en la salida de resultados del algoritmo *(Figura 3.34)*.

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,483    0,135    0,596      0,483    0,533      0,372    0,674     0,438     Malo
                0,450    0,151    0,667      0,450    0,537      0,329    0,649     0,521     Regular
                0,721    0,391    0,449      0,721    0,553      0,304    0,665     0,409     Bueno
Weighted Avg.   0,543    0,220    0,579      0,543    0,541      0,334    0,661     0,463
```

Figura 3.34: Indicadores de precisión

4. **Algoritmo de clasificación J48**

Este método, implementación del algoritmo **C4.5**, es un algoritmo basado en clasificación por *árbol de decisión*. El algoritmo ofrece la posibilidad de poder parar antes de alcanzar las hojas en cada subárbol; esto dará lugar a árboles menos refinados, por decirlo de alguna forma, y ayudará a evitar el *overfitting*.

Los árboles de decisión entran dentro de los métodos de clasificación supervisada, es decir, se tiene una variable dependiente o clase, y el objetivo del clasificador es determinar el valor de dicha clase para casos nuevos.

Para su aplicación es necesario acceder a lapestaña **Classify**, situada en la parte superior de la interfaz **Explorer**, y pulsar el botón *Choose* de la sección *Classifier*. Seguidamente seleccionamos *Weka/ Classifiers/ Trees/ J48*.

Seleccionado este método, podemos aplicarlo directamente en los diferentes métodos de evaluación. Además podemos modificar sus propiedades como sea necesario haciendo click en la etiqueta, en nuestro caso no se modificaran. Así pues tenemos:

- *Use training set (Conjunto de entrenamiento).* Seleccionando la opción *Use training set* en la sección *Test options*, y pulsando *Start*, podemos observar la iformación siguiente:

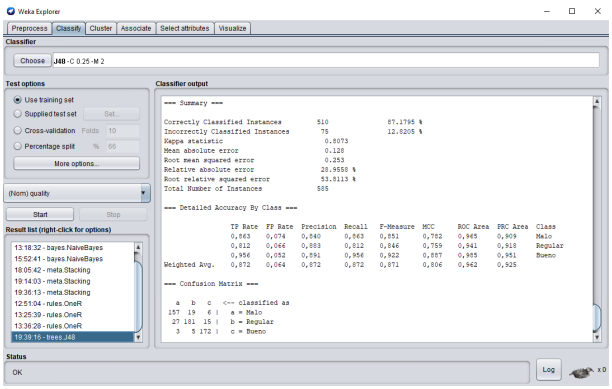
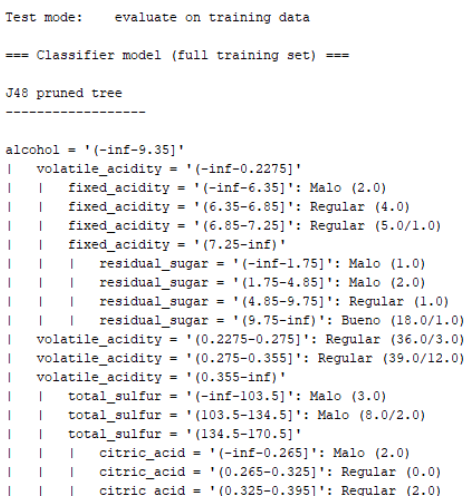


Figura 3.35: Salida de aplicar *J48* usando la opción *Use training set*

Además, tenemos que el programa genera el árbol de decisión para nuestros datos. Parte de ese árbol se muestra en la siguiente imagen.



```
Test mode:    evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree
------------------

alcohol = '(-inf-9.35]'
|   volatile_acidity = '(-inf-0.2275]'
|   |   fixed_acidity = '(-inf-6.35]': Malo (2.0)
|   |   fixed_acidity = '(6.35-6.85]': Regular (4.0)
|   |   fixed_acidity = '(6.85-7.25]': Regular (5.0/1.0)
|   |   fixed_acidity = '(7.25-inf)'
|   |   |   residual_sugar = '(-inf-1.75]': Malo (1.0)
|   |   |   residual_sugar = '(1.75-4.85]': Malo (2.0)
|   |   |   residual_sugar = '(4.85-9.75]': Regular (1.0)
|   |   |   residual_sugar = '(9.75-inf)': Bueno (18.0/1.0)
|   volatile_acidity = '(0.2275-0.275]': Regular (36.0/3.0)
|   volatile_acidity = '(0.275-0.355]': Regular (39.0/12.0)
|   volatile_acidity = '(0.355-inf)'
|   |   total_sulfur = '(-inf-103.5]': Malo (3.0)
|   |   total_sulfur = '(103.5-134.5]': Malo (8.0/2.0)
|   |   total_sulfur = '(134.5-170.5]'
|   |   |   citric_acid = '(-inf-0.265]': Malo (2.0)
|   |   |   citric_acid = '(0.265-0.325]': Regular (0.0)
|   |   |   citric_acid = '(0.325-0.395]': Regular (2.0)
```

Figura 3.36: Árbol de deciasión con la opción *Use training set*

A continuación analizaremos la salida resultante de aplicar el clasificador, definiendo los términos que aparecen y que son útilies para la comparación entre los distintos métodos de clasificación aplicados.

En la *Figura 3.35*, vemos que se han clasificado correctamente el 87.1795 % de las observaciones, mientras que el 12.8205 % lo han hecho de forma incorrecta. Además, se observa que la precisión media por clase es del 0.872.

Seguidamente se muestra el valor del *índice Kappa* (0.8073), indicando que en este caso tenemos una concordancia casi perfecta. Esté índice es una medida de concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Si su valor fuese 1, la concordancia sería perfecta; si fuese 0, la concordancia es debida al azar; y si tomase un valor negativo, la concordancia es menor a la esperada por el azar.

Posteriormente aparecen ciertas medidas asociadas al *error de la clasificación*,

cuyos indicadores asociados son:

- *Mean Absolute Error (error absoluto medio):* es la cantidad que se usa para medir las diferencias entre los cálculos previstos y los observados. El algoritmo ha conseguido un 0.128.
- *Root Mean Squared Error (error cuadrático medio de la raíz):* es un número que permite medir la magnitud media del error, es decir, la diferencia existente entre lo que pronosticamos y sus valores observados. Este valor es siempre mayor o igual al obtenido en *Mean Absolute Error*. El valor aportado por el algoritmo es de 0.253.
- *Relative absolute error (%):* es una forma de medir el rendimiento de un modelo predictivo. Se expresa como una razón, comparando un *error medio (residual)* con los errores producidos por un modelo trivial o ingenuo. El algoritmo ha conseguido un 28.9558 %.
- *Root relative squared error (%):* es relativo a lo que habría sido si se hubiera utilizado un predictor simple. Más específicamente, este predictor simple es solo el promedio de los valores reales. El algoritmo ha conseguido un 53.8113 %.

A continuación, aparecen el número total de casos (585) y un cuadro donde se muestran una serie de indicadores relativos a la precisión de la clasificación. Estos coeficientes se basan en los resultados de la matriz de confusión que aparece al final de la salida. En este caso, la matriz de confusión muestra la siguiente información:

```
=== Confusion Matrix ===

   a   b   c   <-- classified as
 157  19   6 |   a = Malo
  27 181  15 |   b = Regular
   3   5 172 |   c = Bueno
```

Figura 3.37: Matriz de confusión

Por esta matriz, podemos observar que de los 585 casos han sido correctamente clasificados: 54 en el grupo *Malo*, 157 en el grupo *Regular* y 110 en el grupo *Bueno*.

También por estos valores podemos extraer para cada clase los siguientes indicadores de precisión: la **Tasa de verdaderos positivos**, la **Tasa de falsos positivos**, la **Precisión**, **Recall** y la **Medida-F**. Estos valores aparecen en la salida de resultados del algoritmo *(Figura 3.38)*.

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,863    0,074    0,840      0,863    0,851      0,782    0,965     0,909     Malo
                 0,812    0,066    0,883      0,812    0,846      0,759    0,941     0,918     Regular
                 0,956    0,052    0,891      0,956    0,922      0,887    0,985     0,951     Bueno
Weighted Avg.    0,872    0,064    0,872      0,872    0,871      0,806    0,962     0,925
```

Figura 3.38: Indicadores de precisión

- *Cross-validation (Validación cruzada).* Seleccionando la opción *Cross-validation* en la sección *Test options*, y pulsando *Start*, podemos observar la iformación siguiente:

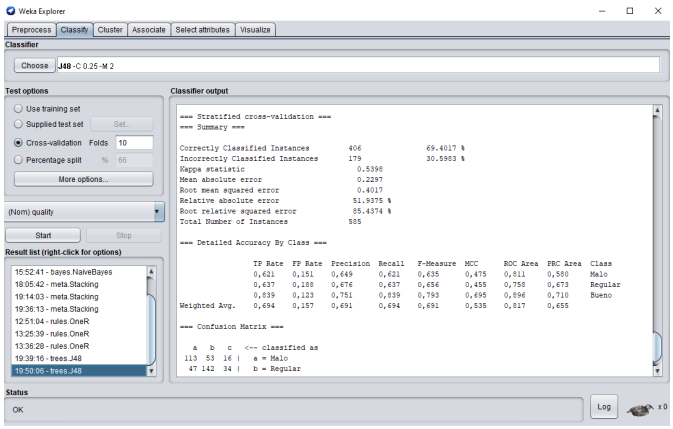


Figura 3.39: Salida de aplicar *J48* usando la opción *Cross-validation*

Además, tenemos que el programa genera el árbol de decisión para nuestros datos. Parte de ese árbol se muestra en la siguiente imagen.

```
=== Classifier model (full training set) ===

J48 pruned tree
------------------

alcohol = '(-inf-9.35]'
|   volatile_acidity = '(-inf-0.2275]'
|   |   fixed_acidity = '(-inf-6.35]': Malo (2.0)
|   |   fixed_acidity = '(6.35-6.85]': Regular (4.0)
|   |   fixed_acidity = '(6.85-7.25]': Regular (5.0/1.0)
|   |   fixed_acidity = '(7.25-inf)'
|   |   |   residual_sugar = '(-inf-1.75]': Malo (1.0)
|   |   |   residual_sugar = '(1.75-4.85]': Malo (2.0)
|   |   |   residual_sugar = '(4.85-9.75]': Regular (1.0)
|   |   |   residual_sugar = '(9.75-inf)': Bueno (18.0/1.0)
|   volatile_acidity = '(0.2275-0.275]': Regular (36.0/3.0)
|   volatile_acidity = '(0.275-0.355]': Regular (39.0/12.0)
|   volatile_acidity = '(0.355-inf)'
|   |   total_sulfur = '(-inf-103.5]': Malo (3.0)
|   |   total_sulfur = '(103.5-134.5]': Malo (8.0/2.0)
|   |   total_sulfur = '(134.5-170.5]'
|   |   |   citric_acid = '(-inf-0.265]': Malo (2.0)
|   |   |   citric_acid = '(0.265-0.325]': Regular (0.0)
|   |   |   citric_acid = '(0.325-0.395]': Regular (2.0)
|   |   |   citric_acid = '(0.395-inf)': Regular (10.0)
```

Figura 3.40: Árbol de deciasión con la opción *Use training set*

A continuación analizaremos la salida resultante de aplicar el clasificador, definiendo los términos que aparecen y que son útilies para la comparación entre los distintos métodos de clasificación aplicados.

En la *Figura 3.39*, vemos que se han clasificado correctamente el 69.4017 % de

las observaciones, mientras que el 30.5983 % lo han hecho de forma incorrecta. Además, se observa que la precisión media por clase es del 0.691.

Seguidamente se muestra el valor del *índice Kappa* (0.5398), indicando que en este caso tenemos una concordancia moderada. Esté índice es una medida de concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Si su valor fuese 1, la concordancia sería perfecta; si fuese 0, la concordancia es debida al azar; y si tomase un valor negativo, la concordancia es menor a la esperada por el azar.

Posteriormente aparecen ciertas medidas asociadas al *error de la clasificación*, cuyos indicadores asociados son:

- *Mean Absolute Error (error absoluto medio):* es la cantidad que se usa para medir las diferencias entre los cálculos previstos y los observados. El algoritmo ha conseguido un 0.5398.
- *Root Mean Squared Error (error cuadrático medio de la raíz):* es un número que permite medir la magnitud media del error, es decir, la diferencia existente entre lo que pronosticamos y sus valores observados. Este valor es siempre mayor o igual al obtenido en *Mean Absolute Error*. El valor aportado por el algoritmo es de 0.2297.
- *Relative absolute error (%):* es una forma de medir el rendimiento de un modelo predictivo. Se expresa como una razón, comparando un *error medio (residual)* con los errores producidos por un modelo trivial o ingenuo. El algoritmo ha conseguido un 51.9375 %.
- *Root relative squared error (%):* es relativo a lo que habría sido si se hubiera utilizado un predictor simple. Más específicamente, este predictor simple es solo el promedio de los valores reales. El algoritmo ha conseguido un 85.4374 %.

A continuación, aparecen el número total de casos (585) y un cuadro donde se muestran una serie de indicadores relativos a la precisión de la clasificación. Estos coeficientes se basan en los resultados de la matriz de confusión que aparece al final de la salida. En este caso, la matriz de confusión muestra la siguiente información:

```
=== Confusion Matrix ===

   a   b   c   <-- classified as
 113  53  16 |   a = Malo
  47 142  34 |   b = Regular
  14  15 151 |   c = Bueno
```

Figura 3.41: Matriz de confusión

Por esta matriz, podemos observar que de los 585 casos han sido correctamente clasificados: 113 en el grupo *Malo*, 142 en el grupo *Regular* y 151 en el grupo *Bueno*.

También por estos valores podemos extraer para cada clase los siguientes indicadores de precisión: la **Tasa de verdaderos positivos**, la **Tasa de falsos positivos**, la **Precisión**, **Recall** y la **Medida-F**. Estos valores aparecen en la salida de resultados del algoritmo *(Figura 3.42)*.

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,621    0,151    0,649      0,621    0,635      0,475    0,811     0,580     Malo
                 0,637    0,188    0,676      0,637    0,656      0,455    0,758     0,673     Regular
                 0,839    0,123    0,751      0,839    0,793      0,695    0,896     0,710     Bueno
Weighted Avg.    0,694    0,157    0,691      0,694    0,691      0,535    0,817     0,655
```

Figura 3.42: Indicadores de precisión

- *Percentage split (Dividiendo el fichero de datos).* Seleccionando la opción *Percentage split* en la sección *Test options*, y pulsando *Start*. A través de este método, el fichero de datos se divide en dos partes, de acuerdo al porcentaje

indicado (que dejaremos en el 66 %). La primera se utilizará para construir el clasificador y la segunda para evaluar su rendimiento. La salida se puede observar en la *Figura 3.43*.

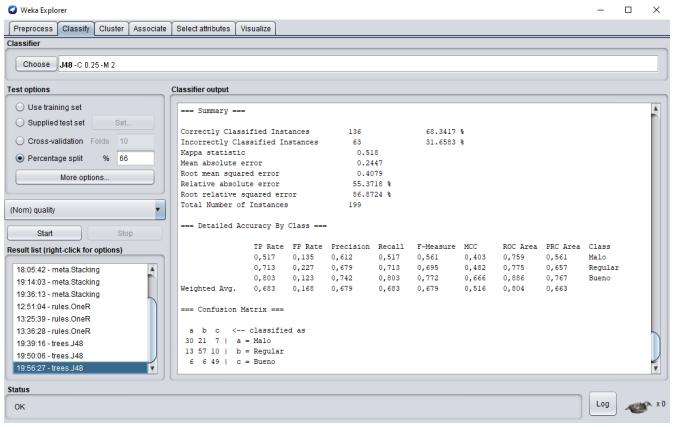


Figura 3.43: Salida de aplicar *J48* usando la opción *Percentage split*

Además, tenemos que el programa genera el árbol de decisión para nuestros datos. Parte de ese árbol se muestra en la siguiente imagen.

```
=== Classifier model (full training set) ===

J48 pruned tree
------------------

alcohol = '(-inf-9.35]'
|   volatile_acidity = '(-inf-0.2275]'
|   |   fixed_acidity = '(-inf-6.35]': Malo (2.0)
|   |   fixed_acidity = '(6.35-6.85]': Regular (4.0)
|   |   fixed_acidity = '(6.85-7.25]': Regular (5.0/1.0)
|   |   fixed_acidity = '(7.25-inf)'
|   |   |   residual_sugar = '(-inf-1.75]': Malo (1.0)
|   |   |   residual_sugar = '(1.75-4.85]': Malo (2.0)
|   |   |   residual_sugar = '(4.85-9.75]': Regular (1.0)
|   |   |   residual_sugar = '(9.75-inf)': Bueno (18.0/1.0)
|   volatile_acidity = '(0.2275-0.275]': Regular (36.0/3.0)
|   volatile_acidity = '(0.275-0.355]': Regular (39.0/12.0)
|   volatile_acidity = '(0.355-inf)'
|   |   total_sulfur = '(-inf-103.5]': Malo (3.0)
|   |   total_sulfur = '(103.5-134.5]': Malo (8.0/2.0)
|   |   total_sulfur = '(134.5-170.5]'
|   |   |   citric_acid = '(-inf-0.265]': Malo (2.0)
|   |   |   citric_acid = '(0.265-0.325]': Regular (0.0)
|   |   |   citric_acid = '(0.325-0.395]': Regular (2.0)
|   |   |   citric_acid = '(0.395-inf)': Regular (10.0)
```

Figura 3.44: Árbol de deciasión con la opción *Use training set*

A continuación analizaremos la salida resultante de aplicar el clasificador, de-

finiendo los términos que aparecen y que son útilies para la comparación entre los distintos métodos de clasificación aplicados.

En la *Figura 3.43*, vemos que se han clasificado correctamente el 68.3417 % de las observaciones, mientras que el 31.6583 % lo han hecho de forma incorrecta. Además, se observa que la precisión media por clase es del 0.679.

Seguidamente se muestra el valor del *índice Kappa* (0.518), indicando que en este caso tenemos una concordancia moderada. Esté índice es una medida de concordancia entre las categorías pronosticadas por el clasificador y las categorías observadas, que tiene en cuenta las posibles concordancias debidas al azar. Si su valor fuese 1, la concordancia sería perfecta; si fuese 0, la concordancia es debida al azar; y si tomase un valor negativo, la concordancia es menor a la esperada por el azar.

Posteriormente aparecen ciertas medidas asociadas al *error de la clasificación*, cuyos indicadores asociados son:

- *Mean Absolute Error (error absoluto medio):* es la cantidad que se usa para medir las diferencias entre los cálculos previstos y los observados. El algoritmo ha conseguido un 0.2447.
- *Root Mean Squared Error (error cuadrático medio de la raíz):* es un número que permite medir la magnitud media del error, es decir, la diferencia existente entre lo que pronosticamos y sus valores observados. Este valor es siempre mayor o igual al obtenido en *Mean Absolute Error*. El valor aportado por el algoritmo es de 0.4079.
- *Relative absolute error (%):* es una forma de medir el rendimiento de un modelo predictivo. Se expresa como una razón, comparando un *error medio (residual)* con los errores producidos por un modelo trivial o ingenuo. El algoritmo ha conseguido un 55.3718 %.
- *Root relative squared error (%):* es relativo a lo que habría sido si se hubiera

utilizado un predictor simple. Más específicamente, este predictor simple es solo el promedio de los valores reales. El algoritmo ha conseguido un 86.8724 %.

A continuación, aparecen el número total de casos y un cuadro donde se muestran una serie de indicadores relativos a la precisión de la clasificación. Estos coeficientes se basan en los resultados de la matriz de confusión que aparece al final de la salida. En este caso, la matriz de confusión muestra la siguiente información:

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 30 21  7 |  a = Malo
 13 57 10 |  b = Regular
  6  6 49 |  c = Bueno
```

Figura 3.45: Matriz de confusión

Por esta matriz, podemos observar que de los 199 casos han sido correctamente clasificados: 30 en el grupo *Malo*, 57 en el grupo *Regular* y 49 en el grupo *Bueno*.

También por estos valores podemos extraer para cada clase los siguientes indicadores de precisión: la **Tasa de verdaderos positivos**, la **Tasa de falsos positivos**, la **Precisión**, **Recall** y la **Medida-F**. Estos valores aparecen en la salida de resultados del algoritmo *(Figura 3.46)*.

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,517    0,135    0,612      0,517    0,561      0,403    0,759     0,561     Malo
                0,713    0,227    0,679      0,713    0,695      0,482    0,775     0,657     Regular
                0,803    0,123    0,742      0,803    0,772      0,666    0,886     0,767     Bueno
Weighted Avg.   0,683    0,168    0,679      0,683    0,679      0,516    0,804     0,663
```

Figura 3.46: Indicadores de precisión

Finalmente hay que indicar que podemos visualizar el árbol de forma gráfica, pulsando el botón derecho sobre el texto *trees.J48* de la sección *Result-list* y seleccionamos la opción *Visualize Tree*. Sin embargo y debido al gran número de ramas y hojas que se generan con nuestros datos no se visualiza de forma correcta.

3.5.6. Conclusiones

En este capítulo, hemos aplicado diferentes algoritmos de clasificación, con el objetivo de poner en práctica los más utilizados y analizar sus resultados. Además se ha comparado la clase predicha con la clase real de las instancias de diferentes formas, seleccionando alguno de los modos de evaluación *(Use training set, Cross-validation y Percentage split)*. La opción *Supplied test set* no se ha aplicado al no disponer de otro conjunto de datos.

Aplicados los algoritmos, pasamos a realizar una comparación entre ellos con el objetivo de identificar el modelo que mejor clasifica los datos. Para ello, nos fijaremos en el *error absoluto*, el *índice de Kappa*, pertenecientes a las medidas asociadas al *error de la clasificación*, y en los porcentajes de instancias bien y mal clasificadas.

Indicar que las medidas asociadas al *error de la clasificación* se calculan a partir de unos valores (d_i) que se obtienen de la siguiente forma para cada instancia: se construye un vector binario que tiene un uno en la posición de la clase a la que pertenece la instancia y ceros en las demás; se determina el vector de probabilidades de asignación a las distintas clases que proporciona el clasificador; se realiza la diferencia entre el par de vectores asociados; y las componentes de los vectores diferencia proporcionan los valores d_i. Para su visualización es necesario pulsar el botón *More options* y seleccionar en la opción *Output predictions*, *Choose/ weka/ classifiers/ evaluation/ output/ prediction/ CSV*.

Los resultados obtenidos en los anteriores procesos de clasificación se resumen en la siguiente tabla:

Classifier	Test Options	Correctly Classified Instances (%)	Incorrectly Classified Instances (%)	Kappa statistic	Mean absolute error
Naïve Bayes	Use training set	68.0342	31.9658	0.5169	0.2453
	Cross-validation	65.1282	34.8718	0.4729	0.258
	Percentage split	71.3568	28.6432	0.5652	0.2378
Stacking	Use training set	82.3932	17.6068	0.7343	0.1925
	Cross-validation	68.0342	31.9658	0.5183	0.2649
	Percentage split	60.804	39.196	0.4085	0.2823
OneR	Use training set	54.8718	45.1282	0.311	0.3009
	Cross-validation	50.2564	49.7436	0.2423	0.3316
	Percentage split	54.2714	45.7286	0.3186	0.3049
J48	Use training set	87.1795	12.8205	0.8073	0.128
	Cross-validation	69.4017	30.5983	0.5398	0.2297
	Percentage split	68.3417	31.6583	0.518	0.2447

Tabla 3.3: Tabla resumen con los resultados de cada clasificador

Tanto el clasificador **Stacking**, como el **J48** ofrecen el mayor porcentaje de instancias correctamente clasificadas superior al 80 %, al evaluar el clasificador sobre el mismo conjunto sobre el que se construye el modelo predictivo. En caso de obtener el modelo por evaluación cruzada o por división del fichero, obtenemos un porcentaje entre el 60 % y 70 %.

Indicar que para el clasificador **Stacking** no es útil usar un conjunto de entrenamiento para calcular el modelo predictivo, puesto que este método trata de corregir sus *sesgos*, aprende cómo cometen errores y pueden tener memoria del conjunto de entrenamiento. Por lo tanto es recomendable utilizar para el análisis los otros modos de prueba.

Para el clasificador **Naïve Bayes** y **OneR** observamos un porcentaje de instancias correctamente clasificadas del 68.0342 % y 54.8718 % respectivamente, al evaluar el clasificador sobre el mismo conjunto sobre el que se construye el modelo predictivo. En caso de obtener el modelo por evaluación cruzada o por división del fichero, obtenemos los siguientes porcentajes: 65.1282 % y 71.3568 % por el **Naïve Bayes**; y 50.2464 % y 54.2714 % por el **OneR**.

Como se ha podido observar, el porcentaje de instancias correctamente clasificadas es mayor al evaluar el clasificador sobre el mismo conjunto sobre el que se construye el modelo predictivo, es decir la opción *Use training set*. Esto es lógico puesto que este clasificador se evalúa en el mismo conjunto sobre el que se creó el modelo de clasificación produciendo una sobreestimación de los resultados. Sin embargo esta situación no es ratificada por el clasificador **Naïve Bayes** donde el porcentaje de instancias bien clasificadas mediante una división del fichero (71.3568 %) es superior al obtenido mediante el conjunto de entrenamiento (68.0342 %).

En relación al tipo de clasificador, es el **J48** el que mejores resultados ofrece, concretamente los valores de los *índices Kappa* para cada modo de prueba (0,8073, 0,5398 y 0,518) son superiores a los correspondientes modos de prueba de los otros clasificadores, salvo en el modo división del fichero por ser el clasificador **Naïve Bayes** el que posee el mayor valor.

Finalmente, es el clasificador de **Naïve Bayes** el que peores resultados ha ofrecido, aún así, los resultados obtenidos por este estimador resultan de gran utilidad, ya que este clasificador ofrece una medida probabilística de la importancia de las variables que intervienen en el problema.

Bibliografía

[1] **Apte, C.** (February, 2003). *The big (data) dig, OR/MS Today.* http://www.lionhrtpub.com/orms/orms-2-03/frdatamining.html

[2] **Berthold, M.** y **Hand, D.J.** (1999). *Intelligent Data Analysis: An Introduction. Springer*

[3] **Bouckaert, R.R.**, **Frank, E.**, **Hall, M.**, **Kirkby, R.**, **Reutemann, P.** y **Seewald, A.** (2013). *WEKA Manual Version 3.7.8. Hamilton, New Zcaland*, University of Waikato.

[4] **Cevallos, J.C.**, **Escobar, M.C.**, **Falcones, J.E.** y **Cevallos, W.J.**, Vol. 32(6), 111-122 (2021). *Modelado laboral de los egresados de la Facultad de Cencias Informáticas de la Universidad Técnica de Manabí (Ecuador).*

[5] **Cubero, J.L.**, **Herrera, F.** y **Berzal, F.** (2010), *Fundamentos de Minería de datos. Máster Oficial de la Universidad de Granada en Soft Computing y Sistemas Inteligentes.*

[6] **Corso, I.** y **Lorena, C.**. *Aplicación de algoritmos de clasificación supervisada usando Weka*, Universidad Tecnológica Nacional.

[7] **Garcia Gutiérrez, J.A.** (2016). *Comenzando con Weka: Filtrado y selección de subconjuntos de atributos basada en su relevancia descriptiva para la clase*, ETSI Informática, Universidad Nacional de Educación a Distancia.

[8] **García, D.** (2005) *Manual de WEKA.* https://knowledgesociety.usal.es/sites/default/files/MANUAL%20WEKA.pdf

[9] **Gonzalez, J.C.**, **Castellón, M.** y **Castejón, M. J.** (2009). *Técnicas de clasificación en el entorno de Weka para la determinación de cultivos de regadío (cítricos) en Librilla, Murcia (España).* http://www.aet.org.es/congresos/xiii/cal20.pdf

[10] **Hand, D.**, *Mannila, H.* y *Smyth, P.* (2001). *Principles of Data Mining. The MIT Press.*

[11] **Hastie, T.** y **Tibshirani, J.** (2001). *The elements of Statistical Learning. Springer.*

[12] **Holmes, G.**, pag 225. ACM (2012). *Developing data mining applications. In International Conference on Knowledge Discovery and Data Mining.*

[13] **Ian, H. W.**, **Eibe, F.** y **Mark, A. H.** (2011). *Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, Burlington, MA, 3 edition.*

[14] **Romero, C.A.** (2014). *Interfaces de usuario Explorer y knowledge Flow en Weka.*

[15] **Navas, F.** (2016). *Introducción a la minería de datos con Weka: Aplicación a un problema económico*, Universidad de Jaén.

[16] **Moujahid, A.** y **Inza, I.** (2010). *Manual de prácticas de minería de datos usando el software WEKA*, Universidad del País Vasco.

[17] *Tema 3: Clasificadores y predictores.* https://www.uv.es/mlejarza/datamine/T3.pdf

[18] *UCI Marchine Learning Repositury* (2007). https://archive.ics.uci.edu/ml/about.html

[19] **Wiliams, G.** (2011). *Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery.*

[20] **Zamora Ruiz, J.**. *Comparativa y análisis de algoritmos de aprendizaje automático para la predicción del tipo predominante de cubierta arbórea*, Universidad Complutense de Madrid.