Facultad de Informática / Informatika Fakultatea

TITULACIÓN: Ingeniería Informática

Análisis del comportamiento del CTC con problemas de clases muy desbalanceadas del repositorio KEEL

Alumno: **Javier Infante Izquierdo**

Director: **Txus Pérez de la Fuente**

Proyecto Fin de Carrera, Junio de 2013

RESUMEN

El desbalanceo de datos es uno de los principales retos del aprendizaje automático de hoy en día. Ocurre cuando el número de instancias de una de las clases es muy superior al número de instancias de la otra. El CTC fue diseñado en el contexto de la detección de clientes fraudulentos de una empresa aseguradora.

El objetivo de este trabajo es comparar el desempeño o rendimiento del algoritmo CTC con un conjunto de resultados disponibles en la bibliografía para 16 algoritmos de aprendizaje automático basados en genéticos y conocidos como GBML (para la inducción de reglas) y otros 6 clásicos no-evolutivos, empleando para ello el mismo conjunto de particiones para el entrenamiento/testeo de una validación basada en *5-fold-cross validation* empleada en 33 bases de datos.

Para ello, dos experimentos han sido realizados utilizando la misma metodología experimental empleada en el trabajo de referencia. Los resultados para el primero muestran que el CTC supera a la mayoría de los algoritmos, obteniendo la segunda posición en la clasificación promedio entre todos los algoritmos, pero sin diferencias estadísticamente significativas con el primero, el Ripper. Por lo tanto, podemos afirmar que el CTC logra resultados competitivos cuando se utiliza para tratar con bases de datos, reducidas y desequilibradas.

En la segunda experimentación sobre el mismo conjunto de 33 bases de datos pero habiéndoles aplicado la técnica SMOTE anteriormente, el CTC se sitúa en la zona media de la clasificación promedio, obteniendo la sexta posición. Esta segunda experimentación es menos extensa que la primera y simplemente busca ser punto de partida para futuras investigaciones, por considerar que los objetivos del proyecto quedan cumplidos con el primer experimento.

Para poder realizar ambas experimentaciones ha sido necesario adaptar la plataforma encargada de generar las submuestras (GureKS) que se emplearán posteriormente para la construcción de los árboles.

Palabras clave: Class imbalance problem, algoritmo CTC, árbol consolidado, árbol de decisión, genetics-based machine learning.

AGRADECIMIENTOS

Este trabajo no hubiera sido posible sin la valiosa ayuda de todas las personas que han colaborado directa o indirectamente en el proyecto y a las que quedo muy agradecido.

En primer lugar quisiera agradecer a mi tutor Txus la oportunidad que me ha brindado para realizar este proyecto y aprender de él, la disponibilidad que ha mostrado en todo momento, el trato recibido y la confianza depositada en mí para la realización de este proyecto.

En global, al grupo de investigación del que forma parte, el grupo ALDAPA, el permitirme realizarlo, los recursos puestos en mi mano (4 máquinas) y la oportunidad que me ha surgido de ir al congreso CEDI con los resultados obtenidos en este proyecto.

En el plano personal, no puedo dejar de agradecer a mi madre Conce y mi padre Antoño, aunque ya no esté entre nosotros, todo lo que han hecho para que pueda llegar hasta este punto, y la paciencia que ha tenido mi madre sobre todo en los primeros años en los que los resultados académicos no fueron los esperados.

A mi hermano, Carlos, que también ha aportado su granito de arena y que en estos momentos está acabando también su Licenciatura en Farmacia, empezamos juntos y acabaremos a la vez.

Por último no me quiero olvidar de mis compañeros de facultad, y amigos, a los que a partir de ahora podré dedicar más tiempo.

A todos, muchas gracias.

Irún, Junio 2013.

INDICE

Capítulo 0

Introducción

Capítulo 1

Minería de datos

Capítulo 2

Aprendizaje automático

Capítulo 3

Árboles de clasificación

Capítulo 4

Problema del desbalanceo de datos

Cápitulo 5

Material, Desarrollo y Métodos

Cápitulo 8

INDICE DE LAS FIGURAS Y TABLAS

Capítulo 0

Introducción

En este capítulo se hace una introducción al trabajo realizado en este proyecto, la motivación del mismo y el estado del arte, así como una descripción de los objetivos que se pretenden abordar. Finalmente se describe capítulo a capítulo la estructura que sigue esta memoria.

0.1 Introducción y motivación

El trabajo que se desarrolla en esta memoria se centra en el análisis del comportamiento del algoritmo CTC diseñado por el grupo de investigación ALDAPA http://www.sc.ehu.es/acwaldap/ para un conjunto de bases de datos muy desbalanceadas.

El objetivo que persigue el algoritmo CTC, es el de construir un clasificador que tuviera en cuenta una distribución de clases distinta a la original, y además que el paradigma fuera explicativo, para poder proporcionar al usuario el motivo de la clasificación. Estos clasificadores han sido llamados árboles consolidados *(Consolidated Trees, CT Trees o CTC).*

En concreto, se pretende realizar el estudio sobre las bases de datos disponibles en el sitio web asociado al proyecto KEEL (http://sci2s.ugr.es/keel/index.php) y que han sido ya utilizadas con diferentes algoritmos diseñados para afrontar el problema de clases desbalanceadas (*Class imbalance problem*) en [FGLBH10].

A. Fernandez, S. García, J. Luengo, E. Bernadó-Mansilla, F. Herrera, "Genetics-Based Machine Learning for Rule Induction: State of the Art, Taxonomy and Comparative Study". IEEE Transactions on Evolutionary Computation 14:6 (2010) 913-941, http://dx.doi.org/10.1109/TEVC.2009.2039140

Las bases de datos se nos facilitan con las muestras *cross-validation* ya preparadas, para reproducir el experimento bajo las mismas condiciones.

Junto a las bases de datos, también se encuentran los resultados obtenidos asociados a la experimentación de este trabajo y se pueden consultar en el sitio web creado a tal efecto: http://sci2s.ugr.es/gbml/.

Esto hace que los resultados del CTC obtenidos con estas muestras sean directamente comparables con los conseguidos por todos los algoritmos en el trabajo mencionado anteriormente.

Para poder construir los árboles de clasificación consolidados a partir de las muestras facilitadas, será necesario adaptar las plataformas de software del grupo que implementan el algoritmo CTC (GureKS y Haritza), para que acepten el formato de muestras utilizado en KEEL.

Para ello, habrá que analizar, comprender y estudiar el formato de datos empleado en KEEL, así como estudiar el formato de datos de la UCI, estándar empleado en *machine learning* y el propio de la herramienta GureKS, puesto que ésta es la generadora de muestras para construir un árbol consolidado en Haritza.

Con el fin realizar las experimentaciones de la manera más automatizada posible, se empleará la gramática generadora de scripts, desarrollada en el contexto del grupo ALDAPA. Esto nos ofrece una mayor comodidad a la hora de realizar las ejecuciones.

Nos apoyaremos en el lenguaje interpretado por Windows scripting host, VBScript, con el fin de generar los scripts que emplearemos en las plataformas de una manera automatizada.

Aunque el CTC ha sido ya analizado con problemas de clases desbalanceadas obteniendo buenos resultados por el grupo de investigación ALDAPA en [AAGMP11] y a priori puede parecer que este estudio carece de relevancia, cabe destacar que las bases de datos utilizadas en las experimentaciones hasta ahora eran considerablemente grandes y no muy desbalanceadas.

Las bases de datos de KEEL son mucho más desbalanceadas y mucho más pequeñas, por tanto, suponen un problema de aprendizaje mucho más difícil y es interesante estudiar y analizar el comportamiento del CTC en estos casos que podríamos denominar "extremos". Resaltar el hecho de que son públicas, con resultados de 22 algoritmos diferentes, posibilitando al resto de investigadores que añadan los resultados de sus algoritmos a esta comparativa.

0.2 Antecedentes y estado del arte

En este proyecto se hace un extenso análisis del comportamiento del algoritmo de construcción de árboles consolidados en el ámbito del *Class Imbalance*.

Este algoritmo es una modificación del algoritmo propuesto por Quinlan, ingeniero informático, investigador pionero en los campos de la teoría de decisión, y que ha contribuido de una manera más que destacada en el desarrollo de algoritmos de creación de árboles de decisión.

Pese al paso de los años, sus obras siguen siendo de obligada lectura y pueden servir como punto de partida y primer contacto a este modelo de predicción [Qui86] y [Qui92].

Los árboles de clasificación o árboles de decisión son un modelo de predicción utilizado en el ámbito del aprendizaje automático (*Machine Learning*) y de la inteligencia artificial (*Artificial Intelligence*), en el que dada una base de datos se crean diagramas de construcciones lógicas que ayudan a resolver un problema a través de una serie de condiciones que ocurren de forma sucesiva.

Esta modificación del algoritmo por excelencia dentro de la familia de los árboles de clasificación fue diseñada e implementada como resultado de la tesis doctoral de mi tutor en [Per06] y que puede consultarse en la web del grupo.

La motivación principal del desarrollo de este algoritmo surge con el objetivo de construir un sistema automático de detección del fraude.

El motivo principal por el que se parte del paradigma de clasificación de los árboles de decisión es que, además de clasificar un elemento nuevo como el resto de

paradigmas, en este contexto, tiene una importancia vital el ofrecer una explicación de la clasificación realizada en función de las características descriptivas de los casos recogidos en las bases de datos.

Este algoritmo se basa en el C4.5, al que se decida una sección en el capítulo 3, y más concretamente en la sección 3.3.4 Algoritmo C4.5, pero extrae el conocimiento de un conjunto de muestras para construir un árbol de decisión individual. El objetivo era generar muestras que contengan la proporción de los casos de fraude (clase minoritaria) estimada por los expertos de la compañía de seguros que se produjo en la realidad.

Por lo tanto, el algoritmo CTC permite extraer conocimiento de un conjunto de muestras con una distribución de clase distinta de la existente en la muestra original.

La idea se basa en generar un número de submuestras con el fin de cubrir la mayor parte de los casos de la base de datos original e ir generando nodo a nodo el árbol de clasificación de cada muestra.

Puesto que no todos los árboles van a elegir la misma variable para dividir el nodo, se realiza un proceso de votación y se fuerza a dividir el nodo por la variable más votada, dicho de otra manera, se consolida la variable y de ahí subyace el nombre de este algoritmo.

Gracias a los trabajos desarrollados por el grupo de investigación y los resultados obtenidos en éstos, se demostró que el paradigma de consolidación obtiene buenos resultados en comparación con otros clasificadores simples.

Entre estos trabajos y basándose en la siguiente afirmación de Weiss y Provost, que se puede consultar en la obra [WP03], cada dominio tiene una distribución óptima de la clase que se utilizará para el aprendizaje, sea cual sea la distribución original de la clase. El CTC fue aplicado a muestras generadas con la distribución de clases óptima del C4.5 en un nuevo estudio, concluyendo que los resultados fueron, en promedio, mejores. Los autores sospecharon que la distribución de clases óptima para CTC podría ser diferente.

Como resultado de esta investigación y siguiendo este camino, se dio comienzo a otra tesis doctoral desarrolla por Iñaki Albisua, cuyo principal objetivo fue valorar el comportamiento de los árboles consolidados frente a los que son a día de hoy algunos de los grandes retos del aprendizaje automático, y encontrar la distribución de clases óptima para el CTC en una experimentación amplia con 30 conjuntos de datos y 14 distribuciones diferentes de clase, que van desde el 2% al 98%.Este trabajo puede consultarse en la web del grupo de investigación [ALB12].

Uno de estos retos, en los que los investigadores están realizando un mayor esfuerzo, es en el denominado *Class Imbalance Problem* [YW06] [HG09],

Capítulo 0
Introducción

En diversos problemas de reconocimiento de patrones, se ha observado que el desequilibrio de clases puede disminuir el desempeño del clasificador, principalmente en los patrones de las clases minoritarias.

Esto ocurre puesto que el clasificador intenta reducir el error global, de forma que el error de clasificación no tiene en cuenta la distribución de los datos.

En estos dominios la clase minoritaria es la que suscita mayor interés y sin embargo, es la clase donde peores resultados de clasificación se obtienen con los clasificadores estándares.

Esta es la razón por la que se han realizado numerosas investigaciones, intentando desarrollar métodos de remuestreo inteligentes, con el fin de equilibrar la clase de las muestras de entrenamiento antes de llevar a cabo el aprendizaje. Entre ellas las expuestas en la tesis doctoral mencionada anteriormente, a la que se pueden sumar los siguientes artículos [AMP10], [BPM04].

Los resultados de la Tesis Doctoral de Iñaki Albisua sugieren que, el algoritmo de construcción de árboles consolidados es una buena opción a la hora de afrontar problemas de clases desbalanceadas, más aún, si tenemos en cuenta el ahorro que supone desde el punto de vista computacional.

En este proyecto se pretende refrendar estas conclusiones y presentar un estudio sobre las bases de datos disponibles en el sitio web asociado al proyecto KEEL [AFLDGSH11] que han sido ya utilizadas con diferentes algoritmos diseñados para afrontar el problema de clases desbalanceadas y comparar los resultados obtenidos por el CTC, con los resultados publicados en el artículo **"Genetics-Based Machine Learning for Rule Induction: State of the Art, Taxonomy and Comparative Study".**

Los autores proporcionan el estado del arte sobre los algoritmos de aprendizaje automático *(Genetic Based Machine Learning)* basado en genéticos. Un enfoque sobre la inducción de reglas y a continuación, se propone una taxonomía de este tipo de métodos. También incluyen un estudio comparativo de los algoritmos estudiados con algunos conocidos algoritmos no evolutivos como CART, AQ, CN2, C4.5, C4.5 Reglas y Ripper.

Para hacer frente a esto, se llevó a cabo una metodología basada en comparaciones jerárquicas donde, en primer lugar se identifican los algoritmos de mejor desempeño dentro de cada categoría y, a continuación, selecciona como representantes de la categoría para la comparación al mejor de cada familia.

Con el fin de estimar el rendimiento de los árboles CTC hemos seguido la misma metodología utilizada por Fernández et al en dicho artículo, la validación cruzada. Se han empleado los mismos conjuntos de entrenamiento y test que se encuentran en la web asociada.

Como resultado de la investigación realizada, se presentará un informe que puede verse en esta memoria. A modo de finalización, se realizará una pequeña conclusión sobre los temas tratados, presente en la última parte de la memoria.

0.3 Objetivos

Como se ha dicho anteriormente, el objetivo principal de este proyecto, es el análisis del comportamiento del algoritmo de construcción de árboles consolidado en el marco de la problemática *Class Imbalance* y realizar un detallado estudio comparativo con los resultados publicados en KEEL para el mismo conjunto de datos en las mismas condiciones.

Para conseguir este objetivo global se han llevado a cabo los siguientes objetivos específicos:

- Adquirir los conocimientos y fundamentos necesarios sobre los principales conceptos de los algoritmos de construcción de árboles de clasificación.

- Comprender los fundamentos necesarios sobre el problema del desbalanceo de clases conociendo las principales técnicas que en su aplicación han obtenido buenos resultados.

- Aprender a utilizar correctamente la librería *Microsoft Foundation Classes*. MFC es un conjunto de clases interconectadas por múltiples relaciones de herencia, que proveen un acceso más sencillo a las API de Windows.

- Estudiar y comprender el formato de las Bases de datos empleado por KEEL.

- Conocer y familiarizarse con las plataformas del grupo ALDAPA en las que se desarrolla la experimentación.

- Estudiar, analizar e interiorizar cómo se encuentran estructuradas las diferentes secciones de la aplicación GureKS, generados de submuestras.

- Implementar en el lenguaje de programación Visual C++ las funciones necesarias para adaptar la plataforma GureKS al tipo de datos de KEEL y generar los ficheros necesarios para su ejecución en Haritza.

- Estudiar y comprender el formato de las bases de datos de la UCI, que es el empleado por la plataforma Haritza para la creación de los árboles *(.names* y *.data*)

- Comprender los distintos métodos para evaluar la bondad y medir la capacidad de aprendizaje de un algoritmo.

- Saber aplicar los principales test de significancia con el fin de comparar los algoritmos y poder deducir si hay diferencias estadísticamente significativas o no.

- Aprender a explotar las ventajas que ofrece Excel y más concretamente el lenguaje Visual Basic scripting, para automatizar la creación de los scripts necesarios para la ejecución del experimento (Haritza, GureKS, y Kurbak)

- Aprender a generar los experimentos sobre las bases de datos mediante el uso de la gramática creada por el grupo de investigación que facilita la ejecución.

- Desarrollar diferentes "*scripts*", capaces de automatizar las comparativas entre los distintos algoritmos del estudio de forma sencilla e intuitiva.

- Probar, experimentar y determinar la validez del algoritmo CTC.

0.4 Estructura de la memoria

Esta memoria se divide en 8 capítulos y 3 apéndices. En el capítulo 0, se hace una breve introducción al núcleo principal de este proyecto, el algoritmo de construcción de árboles consolidados o CTC, recordando la motivación, antecedentes y detallando los objetivos que se persiguen abordar.

En los siguientes 4 capítulos, se desarrolla la base teórica en la que se asienta este proyecto. Se comienza con una introducción a la disciplina de la minería de datos, campo de la computación referido al proceso que intenta descubrir patrones en grandes volúmenes de datos, en el capítulo 1, se describen las técnicas principales y se nombran las aplicaciones más conocidas.

En el capítulo 2, nos centramos en la técnica de aprendizaje supervisado, haciendo una introducción al propio contexto, las técnicas principales y los conceptos teóricos para evaluar estas técnicas, como son la bondad de un clasificador y los test de significancia.

En el 3, se hace una introducción al modelo de predicción en que se basa el algoritmo a estudio. Las características, fundamentos y conceptos de los árboles de clasificación. Así como describir brevemente los algoritmos principales de esta familia. Para concluir el capítulo, se realiza una descripción detallada del algoritmo de construcción de árboles consolidados y sus particularidades.

En el capítulo cuatro, con el que se finaliza la base teórica, nos encontraremos el planteamiento del problema del desbalanceo de clases, así como las distintas técnicas desarrolladas por la comunidad de investigadores para luchar contra este fenómeno.

Así mismo se pretende introducir todos los términos, técnicas del aprendizaje supervisado, contexto de los árboles de clasificación y técnicas de balanceo de datos que se emplearán a lo largo del presente trabajo.

El capítulo 5 lo forma el desarrollo, y abarca desde la implementación de las funciones necesarias para poder llevar a cabo la experimentación, pasando por las herramientas que la harán posibles, su funcionamiento y las técnicas empleadas.

En los capítulos 6 y 7 se realiza una descripción detallada de los experimentos que se llevaron a cabo, con el objetivo de evaluar el comportamiento del algoritmo CTC en el conjunto de bases de datos desbalanceadas proporcionadas por KEEL.

Finalmente se encuentran las conclusiones obtenidas en este trabajo y las líneas abiertas de cara a una futura investigación.

Tras las conclusiones, se presenta la bibliógrafa y algunos apéndices de interés como pueden ser un resumen de los resultados, el código de programación que hace posible la lectura de bases de datos KEEL y una pequeña introducción a los entornos de desarrollo Microsoft Visual C+ 6.0 y VisualBasicScripting.

Capítulo 1

Minería de datos

En este capítulo se hace una introducción a la disciplina de la minería de datos, comenzando desde el proceso de extracción de conocimiento a partir de un gran volumen de datos, para centrarnos después en los procesos propiamente dichos y terminando con las aplicaciones típicas de esta área de la computación.

Capítulo 1
Minería de datos

1.1 Introducción y motivación

En este capítulo se hace una introducción a los conceptos fundamentales relacionados con la minería de datos (*data mining*) que nos ayudarán a hacernos una idea general sobre esta disciplina.

Estamos en una era que podríamos denominar como la era de la información. Consideramos que la información conduce al poder y al éxito y gracias a las tecnologías más modernas somos capaces de recolectar una gran cantidad de datos.

La información almacenada es útil para explicar el pasado, entender el presente y predecir la información futura. Desafortunadamente, esta colección de datos está almacenada en estructuras dispares y muy rápidamente se convirtió en abrumadora.

Este caos inicial llevó a la creación de bases de datos estructuradas y sistemas de gestión de bases de datos (DBMS). La proliferación de sistema de gestión de bases de datos también ha contribuido a la recolección masiva de datos.

En los últimos años, ha existido un gran crecimiento en nuestras capacidades de generar y colectar datos, debido básicamente al gran poder de procesamiento de las máquinas como a su bajo costo de almacenamiento.

Sin embargo, dentro de estas enormes masas de datos existe una gran cantidad de información "oculta", a la que no se puede acceder por las técnicas clásicas de recuperación de la información.

- Los sistemas clásicos de estadística son difíciles de usar y no escalan al número y tipo de datos que se suelen encontrar en bases de datos.
- El usuario final no es un experto en aprendizaje automático ni en estadística.
- El usuario no puede perder más tiempo analizando los datos.

La enorme abundancia de información desborda la capacidad humana de comprenderla y éste es el principal cometido de la minería de datos. Resolver problemas analizando los datos presentes en las bases de datos. El sistema encuentra y sugiere modelos.

Ventajas que aporta:

• Generar un modelo requiere menos esfuerzo manual y permite evaluar cantidades ingentes de datos.

• Se pueden evaluar muchos modelos generados automáticamente, y esto aumenta la probabilidad de encontrar un buen modelo.

• El analista necesita menos formación sobre construcción de modelos y menos experiencia.

1.2 Proceso de extracción de conocimiento

¿Qué es la minería de datos y el *knowledge discovery in databases*?`

Debido a este almacenamiento masivo de la información, bien sea en bases de datos, repositorios y archivos, se hace especialmente importante y necesario desarrollar técnicas nuevas que faciliten el análisis de los datos, la interpretación y la extracción de conocimiento relevante que pueda ayudar en la toma de decisiones.

Veamos tres definiciones sobre el proceso de descubrir conocimiento en las bases de datos:

- *El knowledge discovery in databases o KDD a partir de ahora, es el proceso de identificación de información, válida, novedosa, útil y comprensible. La minería de datos es el núcleo matemático del proceso del KDD y una etapa dentro de este proceso* [MR10].

- *Proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y en última instancia comprensibles a partir de los datos* [Fay96].

- *El descubrimiento de conocimiento en bases de datos, es un proceso que consta de unos pasos, que traducen las colecciones de datos en bruto a nuevos conocimientos* [OZ99].

KDD nace como interfaz y se nutre de diferentes disciplinas:

- Estadística.
- Sistemas de información / bases de datos.
- Aprendizaje automático / IA.
- Visualización de datos.
- Computación paralela / distribuida.
- Interfaces de lenguaje natural.

Básicamente, el KDD está compuesto por los pasos de selección de datos (los datos relevantes para el análisis se recuperan de la base de datos), la limpieza de los mismos (limpiar y preparar los datos), *data mining* (construir modelos descriptivos/predictivos) y evaluación del modelo (conseguir los modelos descriptivos/predictivos que mejor solucionen el problema) [HK06].

Ilustración 1 Etapas que componen el proceso de descubrimiento de conocimiento a partir de bases de datos (KDD)

Filtrado de datos: Mediante el preprocesado, se filtran los datos (se eliminan valores incorrectos, no válidos, desconocidos, etc.), se obtienen muestras de los mismos, o se reducen el número de valores posibles (mediante redondeo, agrupamiento, etc.).

Selección de variables: La selección de características reduce el tamaño de los datos, eligiendo las variables más influyentes en el problema, sin apenas sacrificar la calidad del modelo de conocimiento obtenido del proceso de minería. Los métodos para la selección de características son dos:

1. Los basados en la elección de los mejores atributos del problema.

2. Los que buscan variables independientes mediante test de sensibilidad, algoritmos de distancia o heurísticos.

Extracción de Conocimiento: Se obtiene un modelo de conocimiento, que representa patrones de comportamiento observados en los valores de las variables del problema o relaciones de asociación entre dichas variables.

Interpretación y evaluación: Finalmente se procede a su validación, comprobando que las conclusiones son válidas y satisfactorias.

Minería de datos es un término genérico que engloba resultados de investigación, técnicas y herramientas usadas para extraer información útil de grandes bases de datos.

Situaremos la minería de datos mediante algunas definiciones que se han dado sobre la misma:

"MD es la extracción no trivial de información implícita, desconocida previamente, y potencialmente útil desde los datos" G. Piatesky-Shapiro and W.J. Frawley.

"MD es el proceso de extracción y refinamiento de conocimiento útil desde grandes bases de datos" E. Simoudis, B. Livezey and R. Kerber.

"MD es el proceso de extracción de información previamente desconocida, válida y procesable desde grandes bases de datos para luego ser utilizada en la toma de decisiones" P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, A.Zanasi.

"MD es la exploración y análisis, a través de medios automáticos y semiautomáticos, de grandes cantidades de datos con el fin de descubrir patrones y reglas significativos" M.Berry, and G.Linoff.

«Iterative process of extracting hidden predictive patterns from large databases, using AI technologies as well as statistics techniques» Jesus Mena.

La última definición enfatiza cuáles son las raíces o los fundamentos en los que se sustenta la minería de datos: la inteligencia artificial y la estadística. Comúnmente se ha calificado la minería de datos como la bisagra entre la estadística y la computación ya que hace uso de todas las técnicas que puedan aportar información útil, desde un sencillo análisis gráfico, pasando por métodos estadísticos, complementados con métodos y algoritmos del campo de la inteligencia artificial y el aprendizaje automático que resuelven problemas típicos de agrupamiento automático, clasificación, detección de patrones, asociación de atributos, etc.

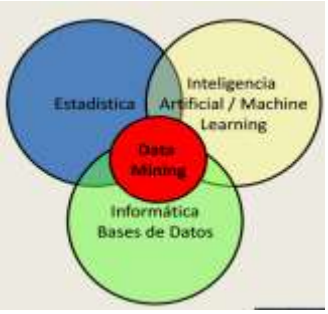


Ilustración 2 Gráfico que ilustra las disciplinas relacionadas con la minería de datos

1.3 Técnicas de minería de datos

Las técnicas de la minería de datos provienen de la inteligencia artificial y de la estadística, dichas técnicas, no son más que algoritmos, más o menos sofisticados que se aplican sobre un conjunto de datos para obtener unos resultados. Resumiendo, se podría definir la IA como la disciplina que se ocupa en ofrecer soluciones algorítmicas con un coste computacional aceptable, mientras que la estadística se ha preocupado más del poder de generalización de los resultados obtenidos con el fin de poder inferir los resultados a situaciones más generales.

No existe una técnica para resolver un problema de minería de datos cualquiera que sea éste, sino distintas aproximaciones que dependiendo del dominio tendrán mejores resultados.

Nos centraremos más profundamente en este tipo de algoritmos en el capítulo 2, explicando más detenidamente los fundamentos y conceptos básicos del aprendizaje supervisado.

Según el objetivo del análisis de los datos, los algoritmos utilizados se clasifican en supervisados y no supervisados *(Weiss y Indurkhya, 1998):*

Algoritmos supervisados (o predictivos): Predicen un dato o un conjunto de ellos, desconocido a priori, a partir de otros conocidos.

Los algoritmos predicen el valor de un atributo de un conjunto de datos, conocidos otros atributos. A partir de datos cuya etiqueta se conoce se induce una relación entre dicha etiqueta y otra serie de atributos. Esas relaciones sirven para realizar la predicción en datos cuya etiqueta es desconocida. Esta forma de trabajar consta de dos fases:

- Entrenamiento: Construcción de un modelo usando un subconjunto de datos con etiquetas conocidas.

- Prueba o Test: Prueba del modelo sobre el resto de los datos.

Algunos de los algoritmos más conocidos son:

- Árboles de clasificación (Quinlan, 1986; Breiman y col. 1984): Puede verse como la estructura resultante de la partición recursiva del espacio de representación a partir del espacio muestral. Esta partición recursiva se traduce en una organización jerárquica del espacio de representación que puede modelarse mediante una estructura de tipo árbol. Cada nodo interior contiene una pregunta sobre un atributo concreto con un hijo por cada posible respuesta y cada nodo hoja se refiere a una decisión clasificación.

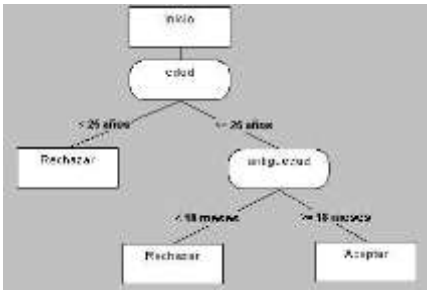


Ilustración 3 Representación gráfica de un árbol de clasificación

- Knn (Covert y Hart, 1967; Dasarathy, 1991): La idea básica sobre la que se fundamenta este paradigma es que un nuevo caso se va a clasificar en la clase más frecuente a la que pertenecen sus *K* vecinos más cercanos. El paradigma se fundamenta por tanto en una idea muy simple e intuitiva.

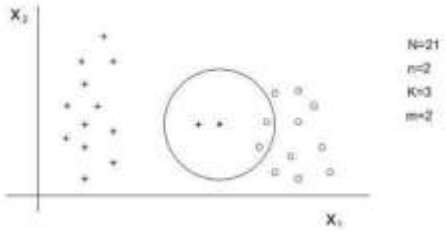


Ilustración 4 Ejemplo de aplicación del clasificador KNN

- Redes neuronales (McCulloch y Pitts, 1943): Sistema compuesto de muchos elementos simples de procesamiento que operan en paralelo, cuya función es determinada por estructura de la red, fuerzas de conexión, y el procesamiento realizado en computación.

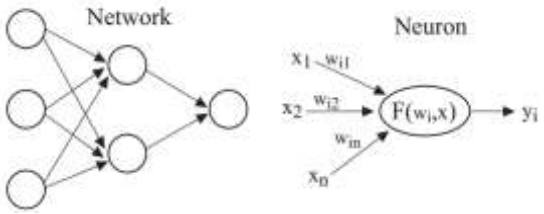


Ilustración 5 Esquema de una red neuronal y a su derecha de una neurona

- Redes bayesianas (Mitchell, 1997): Estructura de datos para representación de conocimiento incierto. Representa la dependencia entre variables, y especifica en forma concisa la distribución de probabilidad conjunta mediante una representación gráfica.

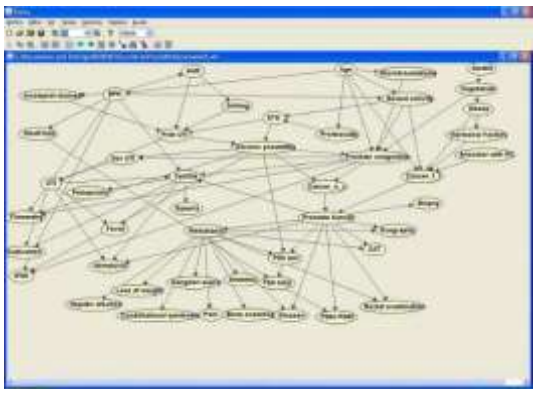


Ilustración 6 Red bayesiana para la detección de cancer de próstata

- Máquinas de vectores soporte (Cristianini y Shawe–Taylor, 2000): Las máquinas de vectores soporte son sistemas de aprendizaje que utilizan como espacio de hipótesis, funciones lineales en espacios característicos de dimensión muy alta, ensayando algoritmos de aprendizaje de la teoría de la optimización que implementan un aprendizaje sesgado derivado a partir de la teoría del aprendizaje estadístico.

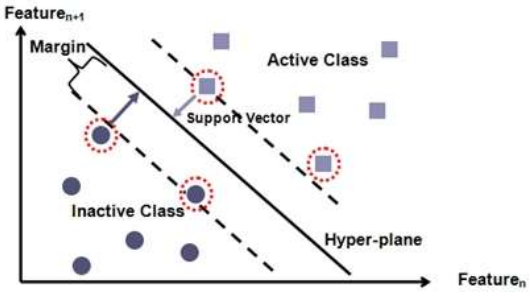


Ilustración 7 Separación de un conjunto de datos mediante SVN

Algoritmos no supervisados: Agrupan los casos o patrones en grupos o clústeres en base a las características que los definen. No existe esa variable especial que es la clase o variable dependiente. Es precisamente el objetivo: buscar la clase, es decir, los clústeres.

Descubren patrones y tendencias en los datos actuales. El descubrimiento de esa información sirve para llevar a cabo acciones y obtener un beneficio (científico o de negocio) de ellas.

Los algoritmos más destacados son los que se presentan a continuación:

- K-means (MacQueen, 1967): Es una manera simple y fácil clasificar un determinado conjunto de datos a través de un cierto número de clúster fijado a priori.

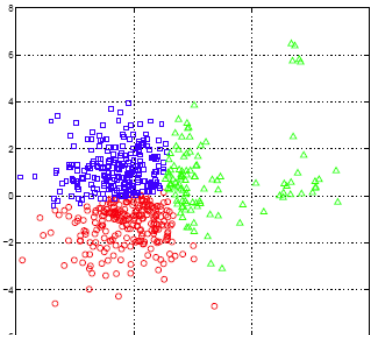


Ilustración 8 Ejemplo de aplicación del K-means

- Mapas autorganizados de Kohonen (T.Kohonen, 1982): Son un algoritmo, que a partir de un proceso iterativo de comparación con un conjunto de datos y cambios para aproximarse a los mismos, crea un modelo de esos mismos datos que puede servir para agruparlos por criterios de similitud. Esta proyección de los datos sobre el mapa distribuye sus características de forma gradual.

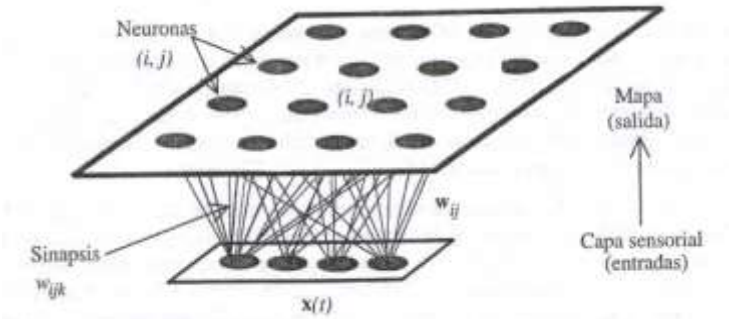


Ilustración 9 Arquitectura típica de un mapa SOM

1.4 Áreas de aplicación

La minería de datos se presenta como una tecnología emergente. Hoy en día existen aplicaciones que utilizan agentes basados en aprendizaje en numerosas ramas de la industria y de la ciencia. Por ejemplo:

- **Procesamiento del Lenguaje Natural** Para el análisis sintáctico y morfológico de los textos. Extracción de Información, Clasificación Automática de Documentos.

- **Sistemas de Recuperación de Información** Los buscadores de Internet utilizan estas técnicas para mejorar el rendimiento de sus búsquedas y confeccionar rankings personalizados según la experiencia de los usuarios.

- **Diagnóstico Médico** Se emplean para asistir a médicos en el diagnóstico según la historia clínica y los síntomas que presenta el paciente.

- **Ciencias biológicas** Para la clasificación de especies, reconocimiento de tumores o arritmias o patrones en cadenas de ADN. Descubrimiento y predicción de patrones de movilidad.

- **Finanzas e Industria bancaria** La industria bancaria utiliza modelos de riesgo crediticio para calificar a los solicitantes según su nivel de cumplimiento esperado en el pago de las cuotas del crédito. También existen modelos de fraude de consumo de tarjetas de crédito, compañías de seguros y de predicción de comportamiento en el mercado de valores.

- **Análisis de imágenes** Reconocer objetos dentro de una imagen, como personas, rostros, dígitos, monumentos arquitectónicos o accidentes geográficos, etc.

- **Juegos** Existen muchos juegos, como el Backgammon y las Damas, donde programas de computadoras que han sido entrenadas con técnicas de ML superan a los campeones mundiales.

- **Robótica** Se utilizan modelos de ML para regular el desplazamiento de robots, planificación de tareas, entre otras tareas.

Capítulo 2

Aprendizaje automático

Este segundo capítulo se centra en el aprendizaje automático, la rama de la inteligencia artificial que desarrolla técnicas que permiten aprender. Existen distintos tipos de algoritmos empleado para la generación de conocimiento, entre ellos se destacan el aprendizaje supervisado y el no supervisado. Se detallan las técnicas que lo componen, y los principales conceptos sobre las mismas.

Capítulo 2
Aprendizaje automático

2.1 Introducción

Las técnicas de aprendizaje automático constituyen un tema vigente de investigación actual, especialmente en la minería de datos con el fin de obtener información relevante que permita obtener conocimiento a partir de los mismos.

El objetivo que persiguen es desarrollar métodos computacionales que implementen formas de aprendizaje, en particular, mecanismos capaces de inducir conocimiento a partir de los datos.

Lógicamente, las áreas del aprendizaje automático y la minería de datos están estrechamente relacionadas y es muy común confundirlas. No obstante, la principal diferencia que los caracteriza es que la minería de datos tiene un mayor enfoque en el conocimiento comprensible a partir de grandes cantidades de información, mientras que el aprendizaje automático se orienta más a la tarea del aprendizaje propiamente.

Una vez aclarada la diferencia entre las dos disciplinas, se muestran unas definiciones con el fin de entender mejor este concepto.

El aprendizaje automático (*Machine Learning*) es una rama de la inteligencia artificial donde su objetivo es desarrollar técnicas que permitan a las máquinas *aprender*.

Podíamos calificar el concepto genérico de "aprender" como el proceso de adquisición de conocimiento, habilidades, valores y aptitudes, posibilitado mediante el estudio, la enseñanza o la experiencia. En las teorías del aprendizaje se intenta explicar la forma en la que se estructuran los significados y se aprenden conceptos nuevos.

"Aprende a construir o modificar representaciones de aquello con lo que se está experimentando" (McCarthy)

Aplicado este concepto a la informática:

"Un programa de ordenador APRENDE a partir de una experiencia E a realizar una tarea T (de acuerdo con una medida de rendimiento P), si su rendimiento al realizar T, medido con P, mejora gracias a la experiencia E" (Mitchell, 97)

Una vez aclarado el concepto de aprender se muestra una definición sobre el aprendizaje automático:

"Proceso de extracción de conocimiento útil e inteligible anteriormente desconocido a partir de grandes cantidades da datos guardados en diferentes formatos". (Witten & Frank 2005)

El área del aprendizaje automático es relativamente amplio, dando lugar a varias técnicas o paradigmas diferentes de aprendizaje que se detallan en el siguiente punto.

2.2 Técnicas de aprendizaje: Tipos y conceptos básicos

No existe una única forma de clasificar los métodos de aprendizaje automático, varía en función del concepto en el que nos basemos para realizar esta distinción, es decir, la naturaleza del conocimiento, la forma de aprendizaje o las técnicas empleadas.

Si nos guiamos por la naturaleza del conocimiento distinguimos el aprendizaje simbólico y el subsimbólico:

- Simbólico: Representación explícita.
- Subsimbólico: Representación no directamente interpretable por el humano.

Si por lo contrario nos basamos en la forma de aprendizaje, se divide en:

- Supervisado: Se conoce la clase a la que pertenece cada uno de los ejemplos.
- No supervisado: Desconocemos la clase y se agrupan automáticamente en clústeres.

Si finalmente nos basamos en las técnicas empleadas de aprendizaje:

- Estadístico: Obtención de un modelo a partir de un conjunto de datos.
- Inductivo: Obtener conclusiones generales a partir de premisas que contienen datos particulares.

Centrándonos en la división propuesta por *(Weiss y Indurkhya, 1998)* distinguimos dos escenarios de aprendizaje, la clasificación supervisada y clasificación no supervisada, que requieren distintas soluciones.

La clasificación siempre ha sido una de las actividades inherentes al ser humano ya que es la base de conocimiento humano. Un ejemplo de ello es la naturaleza, que ofrece una amplia diversidad de poblaciones susceptibles a ser agrupadas.

Clasificación supervisada: También conocido como categorización. Se basa en ejemplos ya clasificados para construir el conocimiento y ser capaz de clasificar los nuevos elementos. Dicho de otra manera, cuentan con conocimiento a priori.

Existen diversas técnicas capaces de hacer una clasificación supervisada y todas tienen en común el siguiente razonamiento: Construir de alguna manera un modelo capaz de decidir a qué clase o categoría pertenece cada uno de los elementos y aplicar alguna función que permita estimar el parecido o similitud entre los casos.

La clasificación consiste en la asignación de los datos en una serie de clases o categorías conceptuales prediseñadas a priori basadas en un atributo especial denominado variable clase o dependiente.

Cada dato contiene un conjunto de atributos, uno de los cuales es el atributo clase, como se muestra en la ilustración que se encuentra a la derecha. La clase, describe el fenómeno que se quiere aprender o sobre el que se desea hacer predicciones.

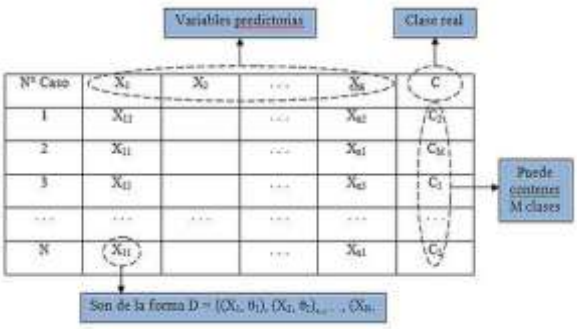


Ilustración 10 Notación para el paradigma de clasificación

Parten de la elaboración de un modelo. Por cada una de las clases se distinguen dos fases: La fase de entrenamiento y la fase de test. La ilustración inferior muestra el esquema general de un modelo de clasificación.

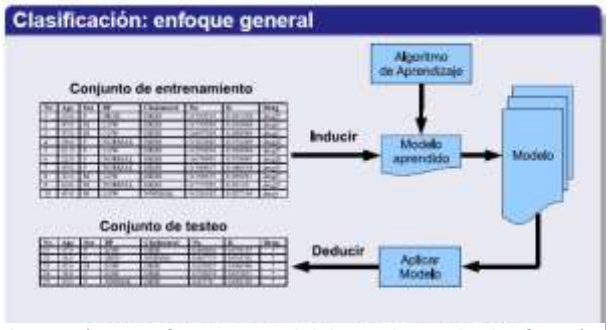


Ilustración 11 Enfoque general del paradigma de clasificación supervisada

El conjunto de entrenamiento sirve para construir el clasificador y el de test para evaluar su precisión o efectividad. En la fase de entrenamiento el objetivo es utilizar el contenido de la base de datos, datos históricos, y así podrá predecir un comportamiento futuro.

El objetivo del algoritmo, es maximizar la razón de precisión de la clasificación de las nuevas instancias, con diferentes medidas de bondad -se detallan en posteriores secciones de este capítulo-, después de realizar una breve descripción del concepto de la clasificación no supervisada.

Clasificación no supervisada: Se diferencia de la clasificación supervisada en que en esta modalidad no dispone de la clase de la instancia. En este tipo de clasificación se dispone de muestras que tienen un conjunto de características. La finalidad es el descubrimiento de grupos de casos que los agrupen. Las características afines a los mismos nos permiten separar las distintas clases. La similitud entre observaciones se establece en términos de distancias. Se busca crear grupos homogéneos y diferenciados a los demás.

Este tipo de clasificación también es denominada agrupamiento o *clustering*. Se diferencian en dos grandes grupos:

- Algoritmos incrementales o bottom-up: Parten de patrones aislados y tienden unirlos mediante grupos en función de algún umbral concreto. Pueden partir de un número de grupos conocidos o desconocidos.

- Algoritmos decrementales o top-down: Parten de grupos ya preestablecidos con el fin de obtener grupos más conocidos.

2.3 Evaluación en la clasificación supervisada

Existen diversos criterios para la evaluación de los clasificadores construidos. Su elección dependerá del dominio del problema.

La matriz de confusión es una herramienta visual muy empleada en el aprendizaje supervisado. En esta matriz, se distinguen filas de columnas: en las filas encontramos las instancias en la clase real, y en las columnas el número de predicciones de cada clase.

clase real / predicha	*verdadero*	*falso*
verdadero	verdaderos positivos	falsos positivos
falso	falsos negativos	verdaderos negativos

Ilustración 12 Tabla de contigencia o matriz de confusión

Esta matriz es útil, ya que a partir de ella, somos capaces de extraer información necesaria para estimar la bondad de un clasificador y mediante ella conseguimos detallar el resultado de la clasificación.

En la tarea de clasificación por cada instancia se producen cuatro valores de salida posibles como son:

- Verdadero positivo o TP: Número de veces que una instancia positiva ha sido correctamente clasificada.

- Falso positivo o FP: Indica el número de veces que una instancia negativa ha sido incorrectamente clasificada, dicho de otra manera, ha sido clasificada como positivo sin serlo.

- Verdadero negativo o TN: Número de veces que una instancia negativa ha sido clasificada como tal.

- Y finalmente falso negativo o FN: Número de veces que una instancia positiva ha sido clasificada incorrectamente.

Cabe destacar que, en estos problemas de clasificación con elementos que solo poseen dos clases, por convenio se hace referencia a la clase positiva como la minoritaria y la negativa como la mayoritaria.

A partir de esta matriz se pueden extraer medidas cuantitativas que permiten entender la naturaleza y distribución del error cometido.

2.3.1 AUC area under curbe

Mide la capacidad de un clasificador para evitar una clasificación errónea, teniendo en cuanta ambas clases, la negativa y la positiva. Es una aproximación del área bajo la curva ROC (Receiver Operating Characteristic), o dicho de otra manera, una representación gráfica de la relación entre el porcentaje de verdaderos positivos y el de falsos positivos.

Puede describirse como una porción del área de un cuadrado de lado la unidad, con valores entre 0 y 1, véase la ilustración inferior. Desde un punto de vista estadístico se define como la probabilidad, de que al elegir aleatoriamente un caso de la clase negativa y otro de la clase positiva, el clasificador de una mayor puntuación al ejemplo de la clase positiva.

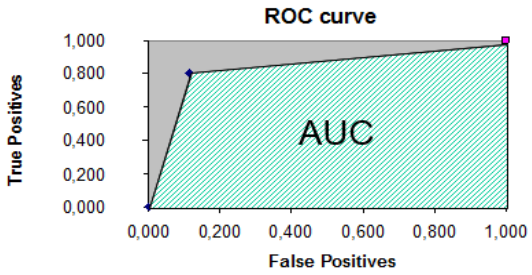


Ilustración 13 Representación de un curva ROC

2.3.2 Error o tasa de error

Es la opuesta de la tasa de acierto: Representa el número de casos clasificados erróneamente. No se distingue entre clases. $\frac{FP+FN}{TP+FP+FN+TN}$

2.3.3 GM

En el caso que pretendemos analizar, la problemática del desbalanceo de datos, la medida *AUC* no puede ser considerada como una medida de bondad del algoritmo, ya que como se ha descrito anteriormente no distingue el número correcto de elementos clasificados entre las clases, lo que puede conducirnos a conclusiones erróneas.

Para verlo de una manera más clara nos basamos en el siguiente caso:

- Si el nivel de desbalanceo de la clase es de un caso minoritario entre 100 casos, tenemos 1 instancia positiva y 99 negativas, en el supuesto de que todos los casos a clasificar se clasifiquen como negativas, tendríamos un 99 % de tasa de acierto pero, paradójicamente no sería un buen clasificador ya que no sería capaz de clasificar ningún caso de la clase minoritaria correctamente, que suele ser, además la de mayor interés.

El objetivo que pretende esta métrica es mantener equilibrado el porcentaje de acierto de ambas clases. Este método consigue considerar la precisión en cada una de las dos clases con el mismo peso. $GM = \sqrt{\frac{TP}{TP+FN} \cdot \frac{TN}{TN+FP}}$

$\frac{TP}{TP+FN}$ Representa el número correcto de casos clasificados pertenecientes a la clase minoritaria o la positiva, con respecto al nº total de casos de esta clase.

$\frac{TN}{TN+FP}$ Representa el número correcto de casos clasificados pertenecientes a la clase mayoritaria o negativa, con respecto al nº total de casos de esta clase.

2.3.4 Kappa

En esta métrica los valores posibles van en el intervalo [-1,+1]. Siendo +1 una tasa de acierto total y -1 una tasa de error total.

$$\frac{TP+TN-E(TP+TN)}{TP+TN+FP+FN-E(TP+TN)} \quad \text{con} \quad E(TP+TN)=\frac{(TP+FN)(TP+FP)+(FP+TN)(FN+TN)}{N}$$

El objetivo es realizar una comparación entre la tasa de acierto obtenida y la que podría esperarse de una clasificación aleatoria.

2.3.5 TP rate

También denominado *Recall*, Sensibilidad o *True Positive Rate*. Es el porcentaje de elementos correctamente clasificados de la clase positiva con respecto a total de elementos existentes de dicha clase, es decir, la efectividad que posee el clasificador a la hora identificar correctamente los elementos de la clase positiva.

$$\frac{TP}{TP+FN}$$

2.3.6 Precisión o precisión positiva

Porcentaje de elementos correctamente clasificados de la clase positiva con respecto a todos los elementos que el clasificador predijo como positivos.

$$\frac{TP}{TP+FP}$$

La inversa sería la precisión negativa comúnmente denominada especificidad.

2.3.7 F-Value

El objetivo que persigue esta métrica es equilibrar los aciertos en la clasificación de ambas clases. Se basa en dos medidas expuestas anteriormente como son el *Recall* o TPR y la precisión o P, a partir de las cuales calcula la media armónica. Los valores que puede tomar van del intervalo [0,1] indicando el 1 un total de aciertos.

$$\frac{2}{\frac{1}{P}+\frac{1}{TPR}}=2\cdot\frac{P\cdot TPR}{P+TPR}$$

2.4 Técnicas de validación

El objetivo que persiguen las técnicas de validación es poder probar un modelo bajo todas las circunstancias. Esto es imposible, ya que no disponemos de datos de entrenamiento para simular todas las situaciones posibles, pero podemos estimar el comportamiento que tendrá un modelo mediante técnicas estadísticas de muestreo.

Una medida posible de validación de un modelo puede ser determinar la tasa de error cometida por el mismo para los nuevos casos. Y posteriormente comparar esta tasa frente a otras tasas de error de otros modelos.

Se va a emplear el error como medida de bondad para la descripción de las técnicas de validación, pero igualmente se podría emplear cualquier otra de las medidas que se acaban de explicar.

La idea básica detrás de la validación es construir un clasificador con un conjunto de ejemplos y estimar lo bien que clasifica con otro conjunto diferente. No se trata de que el modelo memorice un conjunto, sino de que generalice lo mejor que pueda.

Se distinguen dos tipos de valores de error; el error aparente o el error real:

- El error aparente no es una tasa fiable, ya que se estima la tasa de error con los mismos casos o instancias con las que ha aprendido el clasificador. Esta estimación es demasiado optimista puesto que lo que busca un clasificador es predecir la clase de un nuevo caso correctamente y no predecir las clases de casos ya clasificados.

- El error real es una tasa mucho más realista. Estima la tasa de error se emplean diferentes casos con los que el algoritmo ha aprendido. Existen diversas técnicas para llevar a cabo esta estimación, las principales y más destacadas se describen brevemente a continuación:

2.4.1 Holdout

El método *Holdout* o método H, particiona los datos en dos conjuntos disjuntos: el conjunto de entrenamiento y el conjunto de testeo. De los métodos que se detallan en esta sección es el más sencillo de todos. El conjunto de entrenamiento es empleado para inducir el modelo clasificatorio, mientras que el conjunto de testeo es empleado para estimar la predicción verdadera.

El conjunto de entrenamiento suele ser aproximadamente dos tercios de los datos totales y el tercio restante es empleado para el testeo.

En esta técnica tiene una vital importancia el tamaño de la muestra, pues mientras que con muestras grandes obtendremos buenos resultados, en el caso de disponer de muestras pequeñas, los resultados obtenidos no son los más óptimos. En el conjunto de

testeo existen casos que resultan importantes desde el punto de vista del aprendizaje, obteniendo una estimación pesimista, porque el algoritmo de inducción sólo utiliza una porción del conjunto de datos. Cuantas más instancias se dejen para el conjunto de validación, más pesimista será la predicción. En la siguiente ilustración se muestra un gráfico que detalla el método.

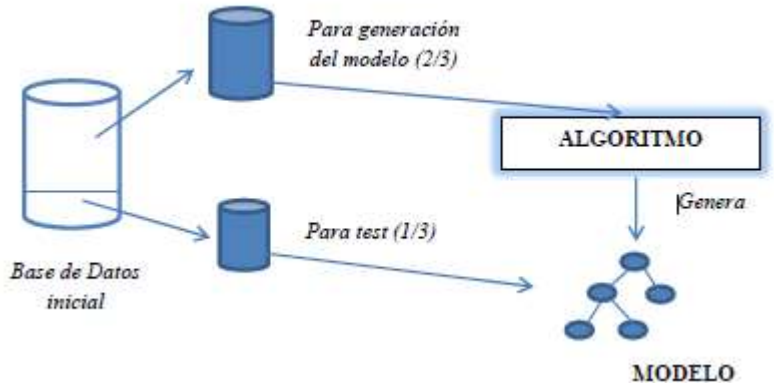


Ilustración 14 Método de validación Holdout H

2.4.2 Submuestreo aleatorio

El método de su muestreo aleatorio o random subsampling es un método que tiene su origen en el método *Holdout* y que podría calificarse como una mejora o modificación del mismo.

El procedimiento se basa en repetir múltiples veces el método H sobre diferentes particiones independientes del conjunto de entrenamiento y conjunto de validación. Para estimar el error en este método, se calcula el error en cada una de las ejecuciones y se haya la media de ellas.

2.4.3 Validación cruzada

Este método es el método de validación más empleado. También denominado método de k-rodajas o *k-fold cross-validation* en inglés. Tiene su origen en 1974 [Sto74].

El conjunto de datos se divide en "m" subconjuntos de datos denominado rodajas o *folds* de aproximadamente el mismo tamaño. Estos subconjuntos son mutuamente excluyentes. m-1 subconjuntos son destinados a entrenamiento mientras que el subconjunto restante es destinado a la validación o test. El clasificador es entrenado y testeado "m" veces. La estimación de la exactitud por medio de este método es la media de la medida de bondad utilizada.

La ilustración 15 muestra el proceso llevado a cabo para aplicar un *3-fold cross validation*. El conjunto de datos se divide en 3 subconjuntos, 2/3 son empleados para entrenar el clasificador, y el restante para el testo. Se repite el proceso 3 veces.

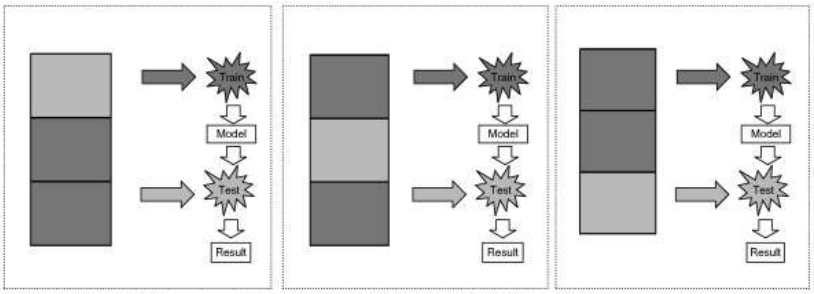


Ilustración 15 Validación cruzada aleataria con 3 folds

2.4.4 Validación cruzada dejando uno fuera (Leave one out)

Esta técnica podría verse como un caso especial del método anterior. La base de datos es particionada k veces, siendo k el número de casos originales n. Se forma un conjunto de entrenamiento con los n - 1 casos dejando uno caso fuera, este caso será el que emplearemos para el test. Este tipo de validación, generalmente es aplicado a bases de datos con relativamente pocos casos ya que presenta una varianza alta. En coste computacional es muy elevado. Se detalla el proceso de manera gráfica en la ilustración 16.

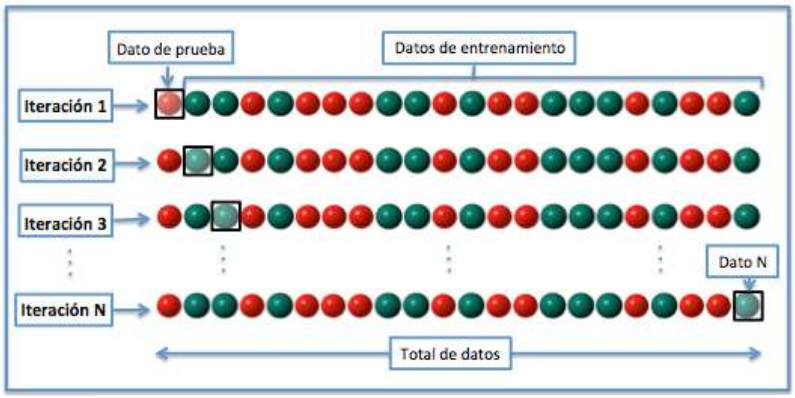


Ilustración 16 Validación cruzada dejando uno fuera

2.4.5 BootStrapping

Método introducido en 1993 por Efron y Tibshirani, su nombre viene de la expresión: *pulling oneself up by one's bootstraps* y se basa en el concepto de reemplazo, que consiste en extraer un número de casos, de forma que tras cada extracción el elemento extraído vuelve a introducirse y puede volver a ser seleccionado.

Este proceso se realiza repetidas veces, calculando la media de los valores obtenidos. Como contrapartida requiere de un mayor gasto computacional que el método *leave-one-out.*

2.5 Test de significancia

A pesar de realizar un gran número de pruebas para demostrar o evaluar el comportamiento de un modelo, las conclusiones derivadas de una comparación directa de los valores obtenidos para los diferentes modelos o escenarios, sobre el mismo conjunto de datos, puede ser erróneas. Es decir, que dichas diferencias pueden ser debidas al azar. Por otro lado, encontrar un algoritmo que sea el mejor en todas las situación resulta a día de hoy imposible, si bien es cierto que para determinados dominios unos se comportan mejor que otros.

Se necesita de alguna herramienta o método que sea capaz de determinar con un cierto grado de seguridad que los resultados obtenidos por un algoritmo son mejores o peores. Esto es posible gracias a la inferencia estadística.

Por ello, es recomendable realizar algún test de validación de los resultados obtenidos, que nos permita asegurar que las diferencias observadas son reales, dicho de otra manera estadísticamente significativas.

Se denomina estadísticamente significativo cuando no es probable que la diferencia observada haya sido debido al azar. Una "diferencia estadísticamente significativa" únicamente quiere decir que hay evidencias estadísticas de que hay una diferencia entre los métodos estudiados, ofreciendo una “garantía de calidad”. Esto no quiere decir que la diferencia sea grande en el sentido estricto de la palabra, indica que hay diferencias.

El nivel de significación de un test es un concepto estadístico asociado a la verificación de una hipótesis. Lo estableció el reputado estadístico Fisher quien señaló “es conveniente trazar una línea de demarcación a partir de la cual podamos decir….”.

Implica utilizar términos comparativos de dos hipótesis, estas comparaciones no son únicas puesto que, existen distintos métodos en función del objetivo deseado. No es lo mismo realizar una comparación entre múltiples clasificadores, que realizarla únicamente entre dos clasificadores.

Para comparar los resultados obtenidos en la experimentación llevada a cabo con los publicados en el trabajo [FGLBH10] hemos empleados los recomendados por Demšar a la hora de comparar múltiples clasificadores [Dem06].

A continuación, se describen, -sin pretender una explicación exhaustiva- los métodos que se emplearán en la metodología experimental de este trabajo.

Las pruebas estadísticas -para la comparación de múltiples algoritmos- han sido prácticamente ignoradas hasta el aporte realizado por Demšar [Dem06]. El test no paramétrico, que recomienda Demšar para comparar múltiples algoritmos es el test Friedman, propuesto por Milton Friedman.

El test de Friedman (Friedman, 1937, 1940) es un equivalente no paramétrico de las medidas repetidas ANOVA. Se clasifican los algoritmos para cada conjunto de datos por separado, elaborando un ranking de estos en función de la posición que ocupan, en el caso que los algoritmos empaten se hace una media de los rankings.

Se trata de un test que detecta si hay o no diferencias significativas entre los algoritmos, pero no entre qué algoritmos se dan. Para determinar entre qué algoritmos se dan dichas diferencias se aplican dos test post-hoc.

Test de *Nemenyi* : Se inscribe en el grupo de pruebas de comparación a posteriori. Considerando que se tienen *k* muestras que provienen de igual número de poblaciones, el proceder es el siguiente:

Nemenyi compara todos contra todos (nxn) y determina entre que par de algoritmos se dan las diferencias.

En la siguiente Tabla se muestran los valores críticos para q0 que se calculan en a través de una fórmula.

#classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.343	2.569	2.728	2.850	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	2.291	2.459	2.589	2.693	2.780	2.855	2.920

Tabla 1 Valores críticos del test de Nemenyi.

Test de Bonferroni-Dunn: El test de BonferroniDunn es similar al test de Tukey para ANOVA y se utiliza cuando queremos comparar un algoritmo frente a los demás. La calidad de dos algoritmos es significativamente diferente si la correspondiente media de rankings es tan diferente como su diferencia crítica.

Elige un algoritmo como control y la comparación se realiza entre este y el resto (1xn)

En la Tabla 2 se pueden consultar los valores críticos para este parámetro.

#classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.241	2.394	2.498	2.576	2.638	2.690	2.724	2.773
$q_{0.10}$	1.645	1.960	2.128	2.241	2.326	2.394	2.450	2.498	2.539

Tabla 2Valores críticos del test de Bonferroni-Dunn.

Ambos test dan la ventaja frente a otros métodos en que ofrecen una representación gráfica *(CD Diagram)*, en el que todas las comparaciones se realizan respecto a una misma diferencia crítica (diferente para los dos tests). En la ilustración 17 se encuentra un ejemplo de la representación gráfica que ofrecen estos métodos.

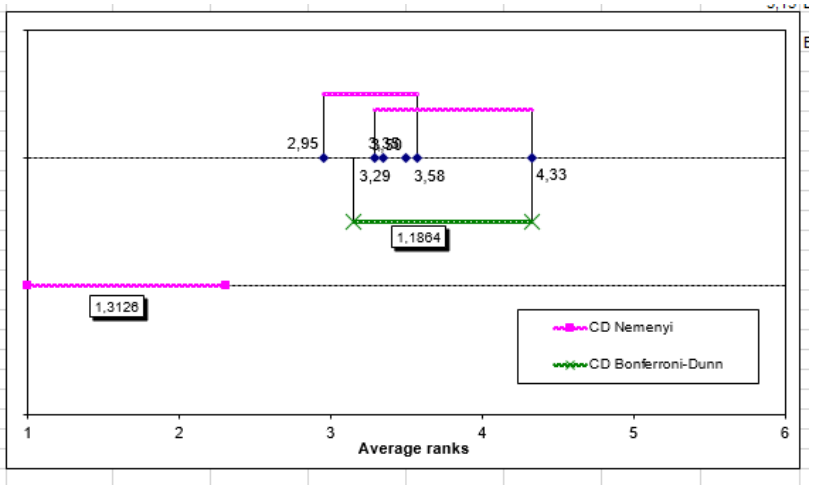


Ilustración 17 Diagrama CD para los test de Nemenyi y Bonferroni-Dunn

En el test de Nemenyi, no existen diferencias significativas si los dos algoritmos aparecen bajo una línea. Al tratarse de comparaciones entre todos los algoritmos podemos encontrar varias líneas. En la parte inferior de la ilustración se muestra la línea que indica el valor de la distancia crítica, 1,3126 en este caso.

En el test de Bonferroni-Dunn, sí existen con respecto al algoritmo de control para aquellos algoritmos cuyo punto este fuera de la línea. Además junto a la línea verde, se indica el valor de la distancia crítica, que es de 1,1864.

En los últimos años los métodos desarrollados por Francisco Herrera et al en [GFLH10] y [GH08] han relegado los test propuestos por Demšar años antes. Surgen ante la necesidad de aplicar procedimientos empíricos más poderosos, en beneficio de obtener más diferencias estadísticas entre los clasificadores comparados.

En cuanto al aporte realizado por Francisco Herrera, destaca el uso del test de Shaffer, siendo éste uno de los procedimientos más potentes que se aplican en la actualidad. Este test tiene mayor poder que el resto de alternativas posibles. Es una versión más poderosa del procedimiento de rechazo secuencial desarrollado por Holm (1979). Shaffer ofrece una mayor potencia del test estadístico asociado con el par de medias que difieren más entre sí.

Capítulo 3
Árboles de clasificación

En este capítulo se presenta el paradigma de los árboles de clasificación, siendo éste uno de los paradigmas clasificatorios más empleados en el ámbito de la clasificación supervisada en múltiples disciplinas. Comienza con una introducción del concepto de inducción para posteriormente profundizar en las características propias. Se presentan también los algoritmos de árboles de clasificación más comunes centrándose principalmente en el C4.5 y el ID3. La última sección del capítulo está dedicada en exclusiva al algoritmo CTC, un paradigma de clasificación con capacidades explicativas diseñado en el contexto de la tesis doctoral de Txus Pérez presentada en el 2006, partiendo del objetivo que persigue abordar y la motivación del mismo.

1

[1] 1[1]La última sección de este capítulo es una adaptación de una de las secciones del capítulo Árboles Consolidados de la tesis de Txus Pérez [Per06].

Capítulo 3
Árboles de clasificación

3.1 Introducción

Dentro de los distintos paradigmas que se han enunciado en el capítulo 1 de esta memoria, los árboles de clasificación o *decision trees* son los más empleados por un conjunto de motivos que los hacen especialmente atractivos. Entre ellos, cabe destacar la sencillez del modelo, la amplitud de implementaciones que existen, la rapidez de clasificación de nuevos patrones y sobre todo la posibilidad de representarlos gráficamente, aportando así una explicación de la división efectuada. Esto nos proporciona un alto grado de comprensión del conocimiento utilizado en la toma de decisiones. Puede resultar de gran utilidad en dominios como la medicina, la detección de clientes fraudulentos… etc.

Se enmarcan dentro de los modelos inductivos, supervisados y no paramétricos. La base en la que se sustenta es la inducción. Por inducción entendemos:

- Método de raciocinio que consiste en alcanzar un principio que se deriva lógicamente de unos datos o hechos particulares.

- Una definición más general sería definir la inducción como el movimiento del pensamiento que va de los hechos particulares a afirmaciones de carácter general.

El aprendizaje inductivo es un caso particular entre las técnicas de aprendizaje que obtienen el conocimiento a partir de ejemplos, siendo su cometido inducir reglas a partir de un conjunto de datos.

Hipótesis del aprendizaje inductivo: cualquier hipótesis encontrada que clasifique un número suficientemente grande de ejemplos de entrenamiento clasificará otros no observados.

Un árbol de clasificación es una forma de representar el conocimiento obtenido en este proceso.

Estos algoritmos tienen su inicio en los años 60. A continuación se muestra una lista de los principales algoritmos que conforman esta familia.

- CLS (Hunt et al., 1966)
- ID3 (Quinlan, 1979)
- CHAID (Kass, 1980)
- ACLS (Niblett et al., 1982)
- CART (Breiman et al., 1984)
- ASSISTANT (Cestnik et al., 1987)
- C4.5 (Quinlan, 1993).

3.2 Descripción general de los árboles de clasificación

El método de construcción de árboles consolidados tiene su origen en el aporte realizado por Hunt et al en 1966. El proceso se basa en la partición recursiva del conjunto de casos de entrenamiento con el fin de quedarnos con el más pequeño. Se fundamenta en el conocido paradigma algorítmico de "Divide y vencerás".

Los algoritmos de construcción de árboles de clasificación se engloban dentro de la familia TDIDT. La idea principal radica en lo siguiente:

Mientras que todos los patrones que se corresponden con una rama del árbol no pertenezcan a una misma clase, se seleccione la variable que de entre las no seleccionadas en esa rama sea la más idónea con respecto de un criterio previamente establecido. Mediante esta variable se expande el árbol en tantas ramas como posibles valores toma dicha variable. A continuación se presenta el pseudocódigo:

Input: D conjunto de N patrones etiquetados, cada uno de los cuales está caracterizado por n variables predictoras $X_1, \ldots, X_n$ y la variable clase C
Output: Árbol de clasificación
Begin TDIDT
  if todos los patrones de D pertenecen a la misma clase c
  then
    resultado de la inducción es un nodo simple (nodo hoja) etiquetado como c
  else
    begin
      1. Seleccionar la variable más informativa X_r con valores $x_r^1, \ldots, x_r^{nr}$
      2. Particionar D de acorde con los n_r valores de X_r en $D_1, \ldots, D_{nr}$
      3. Construir n_r subárboles $T_1, \ldots, T_{nr}$ para $D_1, \ldots, D_{nr}$
      4. Unir X_r y los n_r subárboles $T_1, \ldots, T_{nr}$ con los valores $x_r^1, \ldots, x_r^{nr}$
    end
  endif
End TDIDT

Ilustración 18 Pseudocódigo del algoritmo TDIDT

Se fundamenta en el particionamiento recursivo del dominio de definición de las variables predictorias.

3.2.1 Estructura de los árboles

Los árboles están formados por nodos internos, arcos y hojas.

Cada nodo del árbol representa una subpoblación del conjunto de datos y las ramas representan conjuntos de decisiones, las cuales generan reglas para la clasificación de un conjunto de datos. Las ramas se generan de forma recursiva y este proceso continúa hasta cumplirse el criterio de parada. La complejidad del árbol viene dada por el número de hojas que forman el árbol.

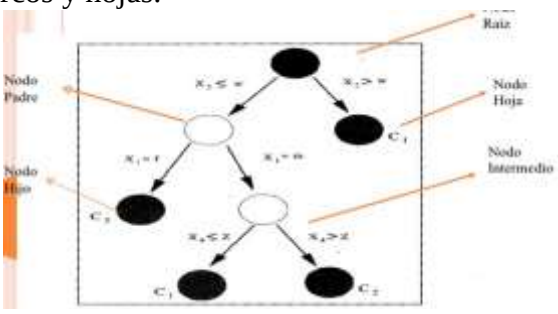


Ilustración 19 Estructura de un árbol de clasificación

En la ilustración 19 se aprecia esta estructura.

Los dos aspectos más destacados a la hora de construir árboles son:

- Cómo se decide la división del nodo.
- El criterio de parada.

Los distintos algoritmos de creación de árboles de clasificación se diferencian en función de dos taxonomías. La división en función de si la variable es discreta o continua, o bien el número de ramas o hijos que podemos encontrar en cada nodo.

Según esta segunda distinción podemos clasificar los algoritmos de construcción de árboles consolidados en:

- Univaluados, sólo contienen un atributo de prueba en cada nodo. Ejemplos de estos algoritmos son:

  ID3, C4.5, CART, FACT, QUEST y CTC.

- Multivaluados, que poseen a un subconjunto de atributos en cada uno de sus nodos.
  LMDT, GALE y C-DT.

¿Qué pasa si en alguno de los atributos falta el valor? Para dividir el árbol es necesario conocer los valores del atributo. Con el fin de evitar este problema se aplican distintas soluciones.

Una posible opción es calcular alguna medida como la media, o la moda y usar este valor. Otra opción empleada es, añadir un nodo hijo "especial" más a cada una de las divisiones y contemplar éste como el valor *missing*, esta opción es la empleada en CHAID, y la tercera opción consiste en repartir proporcionalmente el caso entre las ramas del nodo en las que se pregunta por el valor de la variable desconocida, por ejemplo C4.5.

Como desventaja principal encontramos que son clasificadores débiles, ya que un pequeño cambio en los datos puede tener como consecuencia grandes cambios en el modelo, este fenómeno es conocido como inestabilidad.

3.3 Algoritmos principales y sus características

A continuación se presenta los algoritmos más destacados de los árboles de clasificación, comenzando con el precursor en la década de los 60 el CLS, pasando por los algoritmos propuestos por Quinlan y acabando con los más actuales.

3.3.1 Concept Learning System (CLS)

Este algoritmo es el comienzo de los árboles de clasificación, expuesto en *experiments induction en 1966* por Hunt et al fue el comienzo a los algoritmos de la familia TIDTD expuesta anteriormente. Está enfocado en el diagnóstico médico.

Pueden parecer sencillos ya que sólo se basan en evaluar si ¿atributo = valor? teniendo únicamente dos únicas respuestas sí, y no. Emplea atributos binarios.

Su objetivo primordial es reducir el coste de clasificación de un elemento.

Utiliza una técnica similar al Minimax [2]. Explora el espacio de posibles árboles con un recorrido en profundidad. Como contrapartida requiere de un coste computacional muy elevado.

3.3.2 ID3

Algoritmo propuesto por Quinlan en 1986. Su nombre se deriva de Induction Decision Trees [Qui86].

Comienza el proceso de creación del árbol eligiendo el atributo que dividirá la raíz. Para dicha elección se basa en un test estadístico, con el fin de descubrir qué atributo de todos es el que mejor clasifica los elementos. Esto lo consigue mediante:

- La entropía de una variable: La entropía de una variable aleatoria *X* es una medida cuantitativa de la incertidumbre asociada a los valores tomados por esta variable, y se define como:

$$H(X) = -\sum_{x \in \chi} P_X(x) \log_2 P_X(x)$$

- La efectividad de un atributo: La efectividad para subdividir un conjunto de casos en n subconjuntos.

- La ganancia de información: Propiedad estadística que mide cómo clasifica ese atributo a los ejemplos.

Elige como variable para dividir el nodo del árbol aquella que tenga mayor ganancia de información, para después continuar expandiendo las ramas, considerando la nueva partición formado por el subconjunto de ejemplos que tienen ese valor para el atributo elegido.

---

[2] Es un método de decisión para *minimizar* la pérdida máxima esperada en juegos con adversario y con información perfecta, algoritmo clásico de la IA.

Como principal característica destaca que favorece la elección de variables con un mayor número de valores y que realiza una prepoda. Esta poda consiste en la aplicación de un test de independencia entre la variable predictora y la variable clase. Como consecuencia, solo emplea aquellas variables predictoras para las que se rechaza el test de hipótesis de independencia para la inducción del árbol.

Considera cada nodo del árbol como candidato a la poda. Podar una rama es el proceso de eliminar una rama del árbol, convirtiendo el árbol en más general. Se continúa con este proceso hasta que el árbol podado obtiene peores resultados que el no podado.

```
Id3(Ejemplos, Atributo-objetivo, Atributos )
   Si todos los ejemplos son positivos devolver un nodo positivo
   Si todos los ejemplos son negativos devolver un nodo negativo
   Si Atributos está vacío devolver el voto mayoritario del valor del atributo objetivo en
                                                                          Ejemplos
   En otro caso
        Sea A Atributo el MEJOR de atributos
        Para cada v valor del atributo hacer
             Sea Ejemplos(v) el subconjunto de ejemplos cuyo valor de atributo A es v
             Si Ejemplos(v) esta vacío devolver un nodo con el voto mayoritario del
                                                    Atributo objetivo de Ejemplos
             Sino Devolver Id3(Ejemplos(v), Atributo-objetivo, Atributos/{A})
```

Ilustración 20 Pseudocódigo del algoritmo ID3

3.3.4 Algoritmo C4.5

Surge como una modificación propuesta por Quinlan con el fin de mejorar el ID3 [Quin93].

Como diferencia principal introduce una medida alternativa, el ratio de ganancia. Selecciona en cada nodo el atributo con mayor ratio de ganancia de información. Con esto, evitamos favorecer la elección de variables con un mayor número de valores. Otra diferencia notable con respecto al ID3 es que permite el uso de valores *missing*. También admite variables continuas.

A diferencia del ID3 utiliza una técnica de poda, pero después de haber construido el árbol de clasificación.

La post poda se basa en aplicar un test de hipótesis con el objetivo de deducir si merece o no la pena expandir dicha rama.

Construye el árbol de clasificación a partir del conjunto de entrenamiento. Posteriormente puede convertir el árbol en un conjunto de reglas (*C4.5 Rules*).

Algoritmo 1 Algoritmo C4.5

Entrada:

R: Conjunto de atributos o variables independientes.
C: variable clase o variable dependiente.
S: conjunto de entrenamiento.

Inicio:

si S está vacío **entonces**
  Devolver un nodo único con Valor Error.
fin si

si Todos los patrones de S pertenecen a la misma clase **entonces**
  Devolver un nodo único con dicho valor.
fin si

si R está vacío **entonces**
  Devolver un nodo único con el valor más frecuente de la variable clase en los patrones de S.
fin si

si R no está vacío **entonces**
  D ← atributo con mayor *ratio de ganancia* (D,S) de entre los atributos de R.
  Sean $\{d_j \mid j=1,...,m\}$ los valores del atributo D.
  Sean $\{S_j \mid j=1,...,m\}$ los subconjuntos de S correspondientes a cada valor de d_j.
  Devolver un árbol con rafz nombrada como D y con los arcos nombrados d_1, d_2,..., d_m que van respectivamente a los árboles **C4.5**(R-{D}, C, S_1), **C4.5**(R-{D}, C, S_2),... **C4.5**(R-{D}, C, S_m)
fin si

Ilustración 21 Pseudocódigo del algoritmo C4.5

3.3.5 Algoritmo Chi Squared AID (CHAID)

Método de clasificación jerárquica y descendente. Este algoritmo se basa en seleccionar la variable que más separa los grupos a estudio, usando el criterio chi cuadrado o la razón de verosimilitud.

No aplica ninguna técnica de postpoda para evitar el sobrentrenamiento, sino que es, el propio algoritmo el encargado de decidir cuándo se para de desarrollar el árbol. Sólo es capaz de tratar con variables predictoras discretas. La aplicación fundamental comenzó en el ámbito del diagnóstico médico y la detección de interacción entre variables para posteriormente ser usada en el ámbito del marketing.

3.4 Introducción a los árboles de clasificación consolidados.

Los métodos basados en combinación de clasificadores (*Multiple Classifier System*) tienen una importante relevancia en paradigmas clasificatorios como son los árboles de clasificación, ya que mediante ellos, son capaces de minimizar el principal efecto negativo que produce la clasificación mediante árboles, y no es otra, que la inestabilidad. De ahí que reciban el nombre de clasificadores *weak* o débiles.

En la bibliografía típica se pueden encontrar los métodos más destacados para la combinación de clasificadores, entre ellos, el *Boosting* y el *Bagging* vistos de forma no exhaustiva en la asignatura métodos matemáticos de ciencias de la computación.

Capítulo 3
Árboles de clasificación

El algoritmo de construcción de árboles consolidados conocido como CTC fue desarrollado por Txus Pérez como resultado de su tesis doctoral [Per06], en la que pretendía desarrollar un método capaz de detectar los clientes fraudulentos de una empresa de seguros donde estos casos están representados con un número bajo de casos, en torno al 1% cuando realmente se estima que el porcentaje real de fraude aumenta hasta el 20%.

Por ello, la motivación principal surge en desarrollar un clasificador que tuviera en cuenta una distribución de clases distinta a la original y que aportara una explicación de la clasificación realizada.

El CTC es un meta algoritmo que tomando como base el algoritmo C4.5 de Quinlan, es capaz de aprovechar la información contenida en un conjunto de muestras generadas con la distribución de clases deseada para obtener un único árbol de clasificación.

Este enfoque podría calificarse como una combinación de las dos metodologías o paradigmas que se detallan en el próximo capítulo y que son más comunes a la hora de luchar contra el desbalanceo de datos.

- Trabaja a nivel de remuestreo.
- Trabaja a nivel de algoritmo.

El proceso de construcción de árboles consolidados consta de dos fases principales y diferenciadas:

- La primera fase genera las submuestras que se emplearán para construir el árbol: Extrae un conjunto de Number_Samples submuestras de la muestra de entrenamiento, en base a una de las distintas técnicas de remuestreo posibles, que se indica mediante Resampling_Mode. Podemos generar dichas submuestras con cualquier distribución de clases.

- Cada nodo del árbol es consolidado entre las distintas propuestas, como resultado del proceso de votación de la variable con la que dividir el nodo en cada submuestra. El nodo del árbol consolidado es dividido por la variable más votada. El algoritmo deja de expandir o desarrollar una rama cuando la mayoría de las propuestas para ese nodo sean no dividir. Al terminar de consolidar el árbol se calculan las probabilidades a posteriori de los nodos hojas, calculando la media de estas probabilidades en cada nodo de los árboles asociados al conjunto de submuestras.

El CTC emplea la idea principal del *bagging*, realizando una votación, la misma se aplica para escoger la variable que dividirá el nodo en cada paso a lo largo del proceso de generación del árbol. La elección de la variable se toma en función de las distintas

muestras. Este proceso se repite para cada nodo, o cada variable, dando como resultado un único árbol.

La función empleada para la división de los nodos se basa en el mismo criterio empleado por Quinlan en el C4.5, el *gain ratio*. También se han empleado el resto de parámetros por defecto del C4.5

En la siguiente ilustración se muestra de una manera gráfica el proceso de construcción y consolidación del árbol de clasificación.

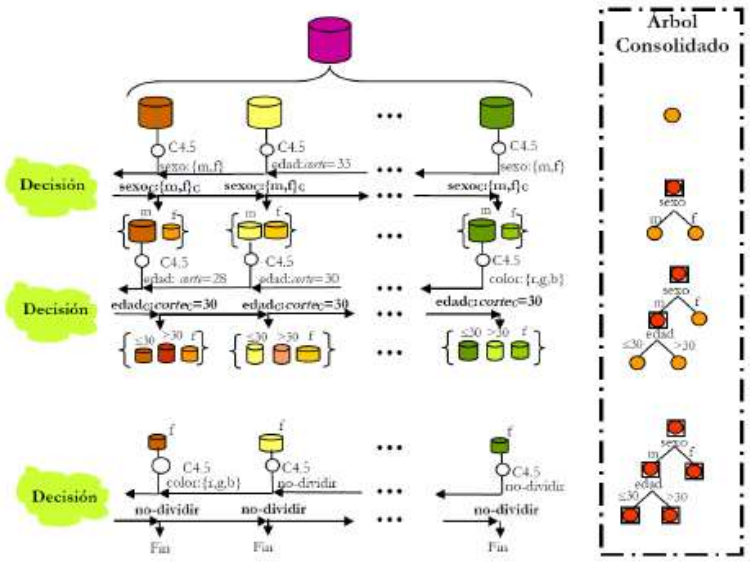


Ilustración 22 Ejemplo paso a paso del proceso de creación de un árbol consolidado

Se parte de la base de datos original -representada en morado-. A partir de esta se crean las submuestras, que se emplearán el proceso de consolidación. Para cada una de las submuestras se crea un árbol de clasificación estándar C4.5, paso a paso, consolidando la decisión nodo a nodo.

En este caso, en la ilustración, las submuestras están representadas mediante los colores naranja, amarillo y verde. Solo se han empleado 3 submuestras para simplificar el proceso pero esto es válido para cualquier número de submuestras.

- En el primer paso, la variable elegida mediante votación para dividir el nodo es la denominada "sexo". El primer árbol de clasificación, el correspondiente a la muestra naranja y el tercero eligen la variable "sexo" para dividir el nodo mientras que la segunda submuestra, la amarilla, elige la variable "edad". Todas las submuestras son divididas en ese nodo entre los dos posibles valores "f" y" m". En este primer paso, hemos consolidado el primer nodo del árbol y se procede de igual manera con el resto de nodos. El tamaño y distribución de clases de los nodos, serán la media de los tamaños y distribución de clases de los nodos correspondientes a las submuestras.

- En el segundo paso, cada una de las submuestras implicadas en el proceso de creación del árbol consolidado es dividida en dos submuestras, se crean tantas muestras por cada submuestra como valores puede tomar la variable. La escogida en esta segunda fase del algoritmo es "edad". El valor de corte elegido es calculado mediante la media y en este caso es 30. Se consolidada la variable y se vuelve a obligar a la muestra a dividir entre la misma.

- Así se continúa con el resto de nodos del árbol hasta se decide por votación mayoritaria que no se siga dividendo el nodo.

Una vez descrito el esquema gráfico, en la ilustración inferior se muestra el algoritmo CTC en pseudocódigo que se pasa a analizar a continuación.

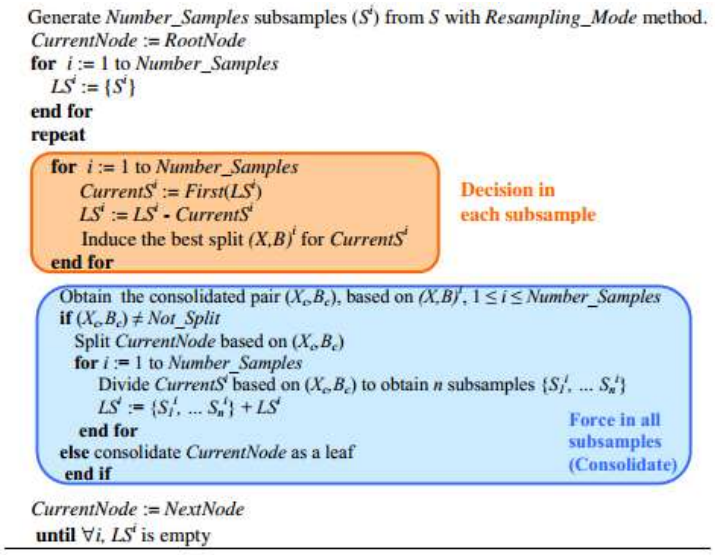


Ilustración 23 Algoritmo de construcción de árboles consolidados

Tal y como se ha comentado al inicio de este capítulo, el proceso de construcción de árboles consolidados se divide en dos fases:

- La creación de submuestras a partir de la muestra de entrenamiento.
- La creación del árbol consolidado a partir de estas submuestras.

Se comienza generando el número de submuestras a partir de la muestra de entrenamiento, este número de submuestras viene dado por la variable Number_Samples. Para ello se remuestrea empleando la técnica de remuestreo indicada por la variable Resampling_Method.

En la segunda fase podríamos diferenciar dos procesos, la elección de la variable que es empleada para dividir el nodo mediante un proceso de votación mayoritaria -dentro de la región naranja-, y el proceso de consolidación -dentro de la región azul cielo-.

Capítulo 4

Desbalanceo de datos

En este capítulo se introduce al lector en la problemática del class imbalance, planteando brevemente la causa del problema y exponiendo cuáles son los métodos más comunes descritos por la comunidad de investigadores para abordarlo.

Capítulo 4
Problema del desbalanceo de datos

4.1 Introducción a la problemática

Nos encontramos con un conjunto de datos desbalanceados, cuando en dicho conjunto de datos una de las clases es representada por un número reducido de casos. En estos casos los clasificadores tienden a ignorar los elementos de la clase minoritaria.

Esto sucede porque la mayor parte de los clasificadores están diseñados para reducir el error de clasificación del clasificador. Suele concurrir el hecho de que, la clase en cuestión sea de especial interés, de modo que clasificar erróneamente casos pertenecientes a dicha clase supone un alto coste.

Esta problemática o reto para el aprendizaje no es concreto de una disciplina, sino que se da en diversos dominios prácticos. A continuación se citan unos cuantos casos:

- *Fraudulent credit card transactions* (Chan, P.K. and Stolfo, S.J., 1998)
- *Learning word pronunciation* (Van den Bosch, A. et al, 1997)
- *Prediction of pre-term births* (Grzymala-Busse, J.W. et al., 2000)
- *Prediction of telecommunications equipment failures* (GrzymalaBusse, J.W. et al., 2000)
- *Detection oil spills from satellite images* (Kubat, M. Et al., 1998)

Como se ha anunciado antes, en la sección referida a los criterios de bondad de un clasificador del capítulo 2, en este tipo de problemas no es adecuado expresar el desempeño del clasificador en función de la tasa de error. Los criterios más adecuados para expresar la bondad o calidad de clasificador en este contexto, son la media geométrica (GM) y las curvas ROC. Esto es así porque son métricas independientes en la distribución entre clases.

Existen principalmente dos enfoques diferenciados para abordar este problema. El primero consiste en aplicar técnicas de muestreo para balancear las clases y el segundo se basa en realizar modificaciones en el algoritmo de clasificación [CJK04].

De la ilustración 24 se desprende, de una manera gráfica, el problema del desbalanceo de datos: nos encontramos muchos ejemplos de la clase negativa, mientras que, el número de ejemplos de la clase positiva es más reducido y se encuentran dispersos.

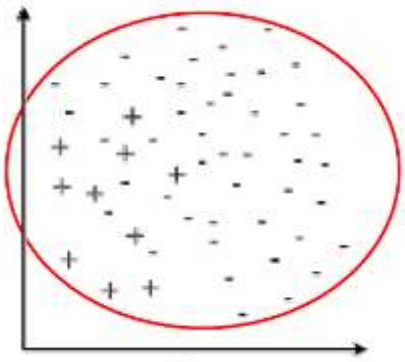


Ilustración 24 Ejemplo gráfico de una distinción no balanceada de datos

4.2 Técnicas empleadas para solucionarlo

Basándonos en la distinción realizada por la gran mayoría de la comunidad investigadora, entre ellos Chawla N.V, Japkowicz N, Kolcz A, existen dos metodologías o maneras de afrontar el desbalanceo de datos.

Realizar un remuestreo con el objetivo de equilibrar la muestra de casos y modificar el algoritmo de clasificación.

4.2.1 Técnicas de remuestreo

La mayor parte de los esfuerzos de la comunidad investigadora han ido orientados a desarrollar técnicas de remuestreo, que produzcan un balanceo de los datos. Existen numerosos artículos que realizan un estudio sobre las distintas técnicas. Uno de los más destacados es [BPM04].

Otro aporte destacado en este ámbito es el realizado por Weiss, quien analizo en profundidad el efecto del cambio de clases en los árboles de clasificación [WP03].

Dentro de las técnicas de muestreo, distinguimos las siguientes opciones:

- Oversampling: Se basan en muestrear la clase minoritaria, aumentan artificialmente el tamaño de la muestra y esto conlleva una mayor carga computacional.

- Undersampling: Se basan en submuestrear la clase mayoritaria, a diferencia del método de oversampling, esto puede eliminar datos potencialmente útiles.

- Genéticos o algoritmos evolutivos: Se basan en el uso de distintas funciones *fitness* que toman en consideración la naturaleza del problema para equilibrar las clases.

4.2.1.1 Métodos de Oversampling

Su objetivo es mantener ejemplos influyentes y balancear el conjunto de entrenamiento. Esta técnica de sencilla aplicación aumenta el número de casos de la muestra minoritaria. Se engloba dentro de las metodologías no heurísticas. Véase ilustración 25.

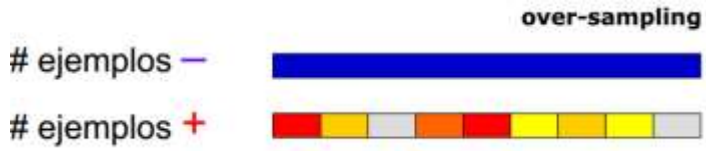


Ilustración 25 Representación gráfica de la técnica de Oversampling

Random Oversampling

El más sencillo y simple dentro de esta familia. Consigue equilibrar la distribución de clases a través de la replicación aleatoria de los casos positivos.

Como contrapartida puede producir el fenómeno *overfiting*, ya que hace copias exactas de las instancias de la clase minoritaria. A pesar del gasto computacional que ocasiona se muestra como una técnica buena en el caso de que dispongamos de pocos casos en la clase minoritaria [BPM04].

Para evitar esta consecuencia surge:

Synthetic Minority Over-Sampling Technique (SMOTE)

Genera nuevas instancias de la clase minoritaria interpolando los valores de las instancias minoritarias más cercanas a una dada [CBHK02].

La generación se realiza extrapolando nuevas instancias en lugar de duplicarlas

```
1  Algorithm: SMOTE ( d is Dataset, N is integer, k is integer ) return (dOut is
   Dataset)
   Data: N is the proportion of over-sampling
         k is the number of neighbors considered to create new instances
   Result: dOut is the new re-sampled data set
2  var
3  |  numNewNeigh, numMin, i is integer
4  |  currEx, newEx, selectedNeigh is Example
5  |  att is Attribute
6  |  dOut, neighbors is Dataset
7  end
8  numMin := number of instances of the minority class in dOut
9  i := 0
10 dout := d
11 while i < numMin do
12 |  currEx := get the ith example of the minority class
13 |  neighbors := get the k nearest neighbors of the minority class closer to currEx
14 |  numNewNeigh := N
15 |  while numNewNeigh > 0 do
16 |  |  selectedNeigh := randomly get a neighbor from neighbors
   |  |  /* create a new example of the minority class
17 |  |  forall attribute att do
18 |  |  |  newEx[att] := currEx[att] + (currEx[att] - selectedNeigh[att])· rand(0,1)
19 |  |  end
20 |  |  numNewNeigh := numNewNeigh - 1
21 |  |  dout := addExample(dout, newEx)
22 |  end
23 |  i := i + 1
24 end
25 return dout
```

(Orriols-Puig, 08)

Ilustración 26 Algoritmo SMOTE en Pseudocódigo

Adaptive Synthetic Minority Oversampling Method (ASMO)

Intenta evitar la expansión de la clase minoritaria en zonas de la clase mayoritaria. Para ello tiene en cuenta la distancia a la clase mayoritaria cuando un ejemplo de la clase minoritaria está rodeado por ejemplos de la clase mayoritaria, en otro caso se utiliza SMOTE.

4.1.2.2 Undersampling

Pretenden borrar instancias ruidosas en el límite, reducir el conjunto de datos y balancear los datos. Este proceso se aprecia en la ilustración 27. A pesar de su simplicidad ha demostrado ser una de las técnicas más efectivas.

Ilustración 27 Representación gráfica del undersampling

Random Undersampling

El método más sencillo de todos, realiza la selección de casos a eliminar de la clase mayoritaria mediante la selección aleatoria de los mismos. Es el método que emplearemos para generar las submuestras para el CTC.

Sin embargo, existe el riesgo de descartar elementos de la muestra potencialmente importantes para el proceso de clasificación.

Debido a esto se han desarrollado métodos con más inteligencia, capaces de realizar una selección inteligente sobre los elementos del conjunto de datos de la clase mayoritaria.

Kubat y Matwin proponen eliminar los casos redundantes y que se encuentran en los bordes (ruido), del conjunto de casos de la clase minoritaria. Métodos para esta tarea más destacados:

Selección de casos en los bordes mediante Tomek links, algoritmos de Wilson, Condensed nearest neighbourt.

La idea en la que se basan estos algoritmos, es suponer que las instancias próximas entre sí tienen mayor probabilidad de pertenecer a la misma clase.

Para clasificar un nuevo elemento se realiza un cálculo de la distancia entre los casos ya existentes y el nuevo y se asocia a la clase más cercana. Como contrapartida principal tiene al alto coste computacional.

Tomek links

El método Tomek Links permite limpiar los datos, elimina los datos ruidosos y los denominados *Borderline*. Esta basados en el conocido paradigma clasificatorio Knn. Se eliminan ejemplos de ambas clases. Véase ilustración 28.

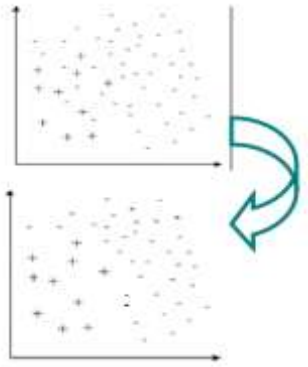


Ilustración 28 Aplicación Tomelinks para eliminar ruido y datos borderlaine

Condensed Nearest Neighbor Rule (CNN)

La idea detrás de este método consiste en eliminar los ejemplos de la clase mayoritaria y que están alejados de la frontera decisión, ya que este tipo de ejemplos pueden ser considerados menos importantes para aprendizaje.

Puede ser interesante mezclar técnicas de oversampling con técnicas de undersampling, ya que SMOTE puede producir *overgeneralization*, dicho de otra manera, introducir elementos de la clase minoritaria en la clase mayoritaria. Para evitar esto se pueden usar Tomek Links.

Como último ejemplo se muestra en la ilustración 29 la combinación aplicada de las técnicas predominantes para luchar contra el desbalanceo de datos.

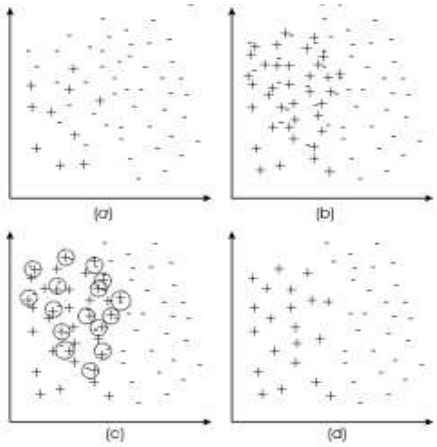


Ilustración 29 Balanceo de datos: a) conjunto de datos original b) oversampling c) Tomelinks d) eliminación de ruido y bordes

4.2.1.3 Algoritmos evolutivos para remuestreo

Los Algoritmos Genéticos son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización.

La selección de prototipos puede ser considerado como un problema de búsqueda, en el que se pueden aplicar algoritmos evolutivos. Estos algoritmos son métodos estocásticos de búsqueda que imitan la evolución biológica natural.

Se basan en la representación de una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor o puntuación, relacionado con la bondad de dicha solución. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para reproducirse.

Este cruce producirá nuevos individuos, descendientes de los anteriores, los cuales comparten algunas de las características de sus padres. Así vamos generando una nueva población de posibles soluciones reemplazando la anterior y conteniendo una mayor proporción de buenas características

Fernández y Herrera propone el uso de algoritmos evolutivos para sub-muestreo en conjunto de datos desequilibrados, denominándolo *Evolutionary Under-Sampling* (EUS) [GCFH06].

Para logar esto destacan dos cuestiones importantes, la especificación de la representación de las soluciones y la definición de la función *fitness*, se trata de una función evaluadora de la calidad de un individuo como solución a nuestro problema.

- Representación: Mediante una representación binaria, un cromosoma se compone de *n* genes, con dos posibles estados: un 1, representa que esa

instancia es añadida en la submuestra, un 0 representa que no pertenece a la submuestra.

- Función *fitness*: Se define como una función de aptitud que combina dos valores, el ratio de clasificación y el porcentaje de reducción de las instancias.

4.2.2 Adaptación de los algoritmos

Con este enfoque lo que se busca es cambiar los algoritmos clásicos de clasificación, con el fin de mejorar su precisión en bases de datos desbalanceadas.

Otra propuesta ampliamente estudiada son los sistemas difusos o las máquinas de vectores soporte.

4.2.3 Técnicas de remuestreo y adaptación

Un ejemplo de esto sería el CTC, *Consolidated Tree Construction*. Realiza una combinación de las dos variantes expuestas anteriormente, por un lado emplea técnicas de remuestreo para generar las submuestras y por otro realiza una adaptación del algoritmo C4.5.

Capítulo 5

Material, Desarrollo y Métodos

En el capítulo quinto se encuentra el desarrollo del proyecto como tal, una vez alcanzada la base teórica en los capítulos anteriores, se enmarcan los esfuerzos llevados a cabo para adaptar la plataforma del grupo ALDAPA a los requerimientos necesarios. Comienza con un análisis de los formatos de datos implicados UCI y KEEL, la familiarización con las herramientas que se emplearán, la interiorización de la estructura de las diferentes secciones que conforman GureKS para abarcar posteriormente la programación de funciones que hacen posible esta adaptación. Finalmente se describe el proceso empleado para la utilización del resto de herramientas que posibilitan el estudio

5.1 Introducción

El objetivo principal que pretende este proyecto, no es otro que, realizar un estudio o análisis del comportamiento del algoritmo CTC desarrollado en mayor medida por Txus Pérez -con la colaboración de los diversos componentes que han formado el grupo de investigación ALDAPA a lo largo de estos años - en el ámbito de la problemática del desbalanceo de datos.

Las bases de datos empleadas para dicho estudio han sido proporcionadas gracias al proyecto KEEL, sobre las cuales disponemos de un extenso artículo comparativo, en él, se propone el uso de algoritmos genéticos para luchar contra el desbalanceo de datos, a la vez que muestra los resultados de los algoritmos clásicos del *machine learning*. De esta manera, podemos comparar el desempeño del CTC en este contexto con 22 algoritmos y en un conjunto de 33 bases de datos.

El estudio [FGLBH10] es relevante, pues posee un gran factor de impacto y ha sido publicado en la prestigiosa revista IEEE *Transactions on Evolutionary Computation,* que posee un factor de impacto 3.341.

Para llevar a cabo este análisis del comportamiento del CTC hay que abordar los objetivos derivados del principal. El punto de partida ha sido alcanzar la base teórica necesaria que se podría resumir en los siguientes puntos:

- Situarnos en la disciplina de la minería de datos y el fundamento del KDD, entendiendo en qué se basa el proceso de adquisición de conocimiento a partir de bases de datos, las etapas que lo componen, sus principales características, las aplicaciones más comunes y las diversas técnicas.

- El paradigma del aprendizaje automático, aclarando en qué consiste el proceso de aprendizaje, las distintas variantes del mismo, centrándonos en el aprendizaje supervisado. Conocer cómo se evalúa el poder de un clasificador y los principales métodos de validación sobre los mismos.

- La compresión del paradigma de los árboles de clasificación, conociendo los principales algoritmos, y el proceso de inducción, base de todos ellos en la construcción de árboles.

- Adquirir una primera toma de contacto de la problemática del *class imbalance*, conociendo los esfuerzos realizados por la comunidad científica, para abordar este problema y las técnicas que se han desarrollado a lo largo de estos años.

- Interiorizar el proceso de creación del árbol consolidado, analizando el algoritmo y el proceso de consolidación paso a paso.

Una vez superados los objetivos que, podríamos calificar como pre-objetivos, estamos preparados para abordar el estudio.

El primer paso en esta nueva etapa es tener claro cuáles son las fases por las que debe pasar un estudio de esta envergadura, es decir, aclarar conceptos sobre cómo realizar este estudio, que metodología experimental se empleará, cuáles son las distintas etapas por las que pasa la experimentación, qué herramientas software necesitamos, cómo funcionan, qué ampliaciones hay que hacer en las mismas, en qué lenguaje de programación trabajaremos, y en definitiva tener claro el proceso desde el inicio con el conjunto de muestras, hasta que obtenemos los resultados de clasificación y una vez obtenidos estos cómo los tratamos.

Lo primero que debe de quedar claro es de dónde partimos y a dónde queremos llegar: De una manera resumida, podemos afirmar que partimos de un conjunto de muestras de entrenamiento, y queremos obtener la tasa de acierto o error que comete el clasificador a estudio, cuando clasifica nuevas instancias de la base de datos.

Pero pasar de una muestra al resultado final es un proceso que transcurre por distintas fases. La primera, es obtener un conjunto de submuestras a partir de la muestra, esto en el caso del CTC. Se logra usando las plataformas desarrolladas para este cometido por el grupo de investigación ALDAPA, en concreto GureKS.

Ha sido necesario adaptar la plataforma de experimentación que emplearemos para llevar a cabo el estudio y así poder ampliar su funcionalidad y dotarlo de capacidad para leer bases de datos en el formato *.dat*, que es propio del proyecto KEEL [AFLDGSH11].

Esta plataforma empleada por el grupo para los diversos estudios y proyectos relacionados con los árboles de clasificación, ya sea el C4.5 de Quinlan [Qui93], CHAID o el propio CTC [Per06], consta de distintas aplicaciones con funciones diferenciadas.

Por un lado nos encontramos la aplicación GureKS, que es la encargada de generar las submuestras que se emplearán en la construcción de los árboles y que se detalla en la sección 5.8.1 GureKS. Nos da la posibilidad de crear submuestras a partir de una muestra con la distribución concreta que deseemos y aplicar cualquiera de las principales técnicas de remuestreo.

Por otra parte, para construir los árboles de clasificación a partir de las submuestras obtenidas emplearemos la aplicación Haritza a la que también se dedica una sección más adelante en este capítulo (5.8.2 Haritza).

Estas dos aplicaciones están desarrolladas en la plataforma Windows, usando el lenguaje de programación Visual C++.

Una vez construidos los árboles de clasificación y testeados, se les aplica un tercer programa desarrollado en Java por el grupo ALDAPA. Kurbak es el encargado de tratar las salidas del Haritza, que más adelante se detallan y generar un fichero con el resultado

de la clasificación mostrando las principales medidas de bondad de un algoritmo. Se tiene un fichero por cada uno de los árboles de clasificación construidos.

Obtenidos los ficheros *.aue* por cada base de datos, emplearemos la potencia que ofrece un programa como Office en combinación con el lenguaje VBA para calcular las medias de los 6 parámetros de bondad de una manera automatizada y trabajaremos los resultados mediante gráficas, rankings y test estadísticos.

Una vez resumida de una manera burda el proceso que realizaremos para este estudio, que no pretende ser más que una primera aproximación, se muestra en la próxima página un esquema grafico del proceso completo; no se describe el mismo, pues el objetivo que persigue es introducirnos en el procedimiento para profundizar más detenidamente en futuras secciones de este capítulo.

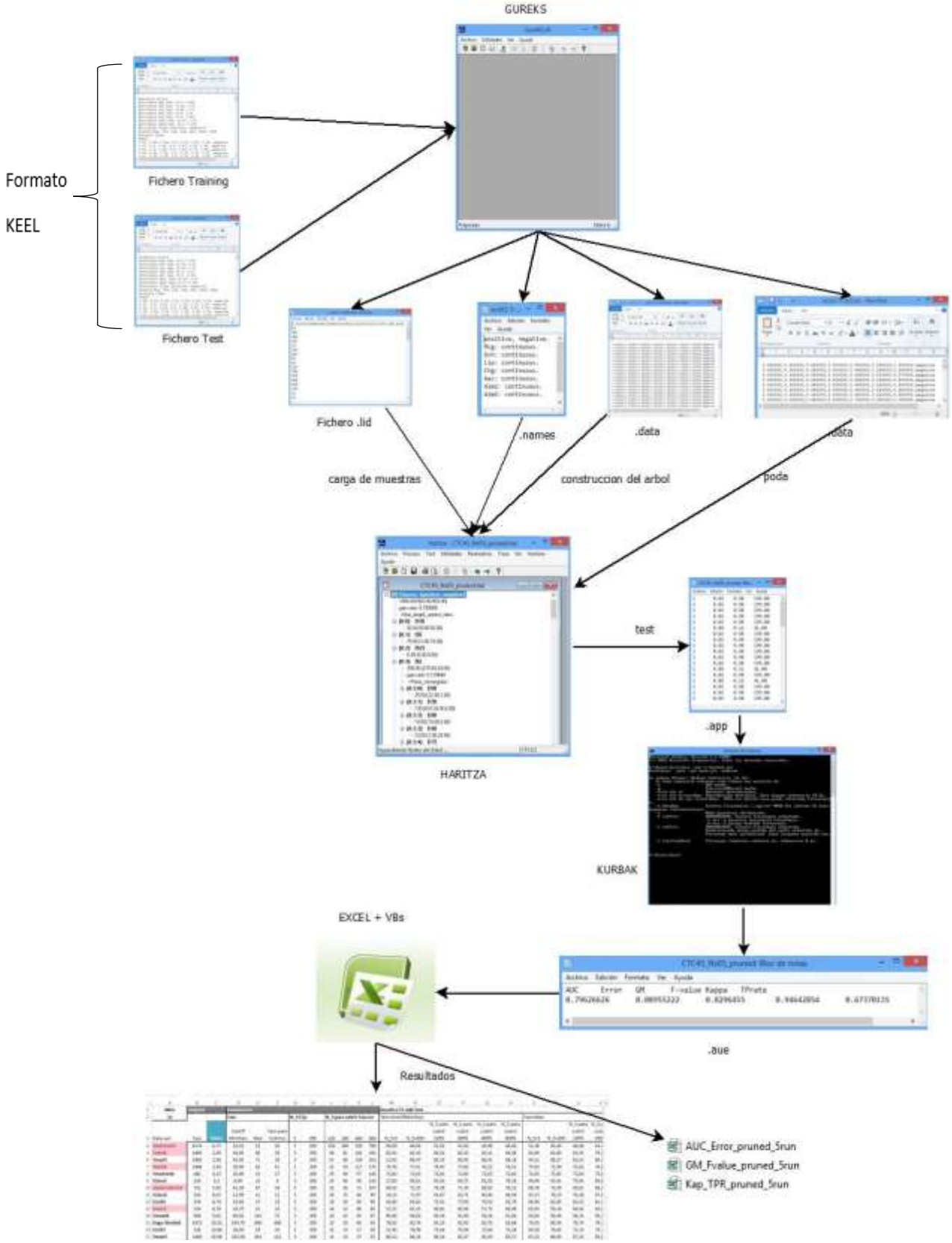


Ilustración 30 Proceso completo de un estudio para un conjunto de bases de datos

5.2 Algoritmos comparados

Se pretende comparar el desempeño del CTC frente a los 16 algoritmos denominados como GBML *(Genetic Based Machine Learning)* para la construcción de conjuntos de reglas propuestos en *genetic based machine learning for rule induction* y otros 6 algoritmos clásicos no evolutivos. Para este fin, se disponen de los mismos conjuntos de entrenamiento y testeo basados en un 5x5 *fold cross validation* empleados por los autores en las 33 bases de datos.

Para ello se sigue la misma metodología experimental desarrollada en el trabajo original, incluyendo una comparación jerárquica a través del test de Shaffer.

5.2.1 Algoritmos basados en genéticos

Los autores especifican el estado del arte en el aprendizaje automático a través de algoritmos basados en genéticos para la inducción de reglas.

Los algoritmos de inducción de reglas tienen por objeto descubrir una descripción para el concepto de destino, mediante reglas explícitas formuladas en términos de pruebas para ciertos valores de los atributos.

El conjunto de reglas resultante debe ser capaz de reconocer correctamente instancias del concepto objetivo y discriminarlos de objetos que no pertenecen a ella. El uso de conjuntos de reglas como representación del conocimiento hace a estos algoritmos muy competitivos en términos de interpretabilidad, puesto que las reglas pueden ser leídas fácilmente por personas expertos.

Se propone una taxonomía sobre este tipo de métodos que define un marco general en el que cada algoritmo puede ser encuadrado.

La metodología empleada está basada en una comparación jerárquica; primeramente se identifica los que muestran un mejor desempeño dentro de cada familia y son elegidos como representantes de la misma.

En cuadro inferior se encuentra un resumen de la taxonomía propuesta para los 16 algoritmos.

1. Chromosome = Rule:
(a) Michigan: XCS and UCS
(b) IRL (Iterative Rule Learning): SIA and HIDER
(c) GCCL (Genetic Cooperative Competitive Learning): CORE, OCEC and COGIN
2. Chromosome = Rule Set:
(a) Pittsburgh: GIL, Pitts-GIRLA, DMEL, GASSIT, OIGA and ILGA

3. Chromosome = Decision Tree or Tree Rule
(a) HEDT (Hybrid Evolutionary Decision Tree): DT-GA, Oblique-DT and TARGET

5.2.2 Algoritmos clásicos de clasificación

En el mismo trabajo se comparan los algoritmos expuestos anteriormente con los algoritmos no evolutivos que clásicamente han obtenido buenos resultados.

- **CN2**
- **CART**
- **AQ**
- **C4.5**
- **C4.5 Rules**
- **Ripper**

5.3 Condiciones del estudio de KEEL

Se han considerado treinta y tres conjuntos de datos de UCI con diferentes relaciones entre las clases minoritaria y mayoritaria. Se encuentra una descripción de cada base de datos en la sección 5.4 Bases de datos del estudio.

En los experimentos desarrollados en el estudio, se ha adoptado 5-veces la validación cruzada como modelo, es decir, se ha dividido el conjunto de datos al azar en 5 rodajas, cada uno contiene el 20% de los patrones del conjunto de datos. De las 5 rodajas o *folds* se han utilizado cuatro pliegues para el entrenamiento y uno para la prueba. Para cada una de las cinco particiones, se han ejecutado 5 ensayos de los algoritmos.

Por cada conjunto de datos, por lo tanto, se considerará los resultados promedio de 25 ejecuciones.

5.4 Bases de datos del estudio

Descripción de las bases de datos empleadas

A continuación se presenta una breve descripción sobre las bases de datos empleadas en los 2 experimentos de los capítulos 6 y 7 llevados a cabo con el fin de analizar el funcionamiento del algoritmo de construcción de árboles consolidados en la problemática del *class imbalance*.

Como ya se ha comentado en las secciones referentes a las experimentaciones, por cada una de las BD se han construido 25 árboles consolidados.

Todas las bases de datos que forman este estudio han sido proporcionadas y pertenecen al proyecto KEEL y se pueden encontrar para su descarga en:

http://sci2s.ugr.es/keel/study.php?cod=13

Estas bases de datos han sido obtenidas del repositorio UCI, pero algunas han sido modificadas. La razón que subyace es convertir las bases de datos multiclase en biclase para obtener dos clases desbalanceadas. Esto lo consiguen mediante la unión de dos o más clases.

Abalone

Esta base de datos contiene la predicción de la edad del abulón a partir de mediciones físicas. La edad de las orejas de mar se determina mediante la reducción de la cáscara a través del cono, la tinción, y contando el número de anillos a través de un microscopio.

BD	abalone	Clases originales	
		{15, 7, 9, 10, 8, 20, 16, 19, 14, 11, 12, 18, 13, 5, 4, 6, 21, 17, 22, 1, 3, 26, 23, 29, 2, 27, 25, 24}	
	abalone 9-18	Versión desequilibrada del conjunto de datos abalone con la siguiente distribución	
		Clase minoritaria	**Clase mayoritaria**
		{18}	{9}
	abalone 19	Versión desequilibrada del conjunto de datos abalone con la siguiente distribución	
		Clase minoritaria	**Clase mayoritaria**
		{19}	^{19}

Ecoli

En esta base de datos se predice el sitio de localización de las proteínas mediante el empleo de algunas medidas sobre la célula (citoplasma, membrana interna, perisplasma, la membrana externa, lipoproteína de la membrana externa, membrana interna, lipoproteína de la membrana interna).

BD	ecoli	Clases originales	
		{cp,im,imS,imL,imU,om,omL,pp}	
	ecoli_0_vs_1	Versión desequilibrada del conjunto de datos ecoli con la siguiente distribución	
		Clase minoritaria	**Clase mayoritaria**
		{im}	{cp}
	ecoli1	Versión desequilibrada del conjunto de datos ecoli con la siguiente distribución	
		Clase minoritaria	**Clase mayoritaria**
		{im}	{cp,imS,imL,imU,om,omL,pp}
	ecoli2	Versión desequilibrada del conjunto de datos ecoli con la siguiente distribución	
		Clase minoritaria	**Clase mayoritaria**
		{pp}	{cp,im,imS,imL,imU,om,omL}
	ecoli3	Versión desequilibrada del conjunto de datos ecoli con la siguiente distribución	
		Clase minoritaria	**Clase mayoritaria**
		{imU}	{cp,im,imS,imL,om,omL,pp}
	ecoli3	Versión desequilibrada del conjunto de datos ecoli con la siguiente distribución	
		Clase minoritaria	**Clase mayoritaria**
		{om}	{cp,im,imS,imL,imU,omL,pp}

Glass

Esta base de datos contiene información del servicio científico forense de EE.UU acerca de 6 tipos de vidrio que se pueden encontrar en la escena del crimen, se define en términos de su contenido de óxido (es decir, Na, Fe, K, etc.).

BD	glass	Clases originales
		{0, 1, 2, 3 , 4 , 5, 6}
	glass0	Versión desequilibrada del conjunto de datos glass con la siguiente distribución **Clase minoritaria**: {0} — **Clase mayoritaria**: {1,2,3,4,5,6}
	glass1	Versión desequilibrada del conjunto de datos glass con la siguiente distribución **Clase minoritaria**: {1} — **Clase mayoritaria**: {0,2,3,4,5,6}
	glass0-1-2-3_VS_4-5-6	Versión desequilibrada del conjunto de datos glass con la siguiente distribución **Clase minoritaria**: {0,1,2,3} — **Clase mayoritaria**: {4,5,6}
	glass2	Versión desequilibrada del conjunto de datos glass con la siguiente distribución **Clase minoritaria**: {2} — **Clase mayoritaria**: {0,1,3,4,5,6}
	glass4	Versión desequilibrada del conjunto de datos glass con la siguiente distribución **Clase minoritaria**: {4} — **Clase mayoritaria**: {0,1,2,3,5,6}
	glass5	Versión desequilibrada del conjunto de datos glass con la siguiente distribución **Clase minoritaria**: {5} — **Clase mayoritaria**: {0,1,2,3,4,6}
	glass6	Versión desequilibrada del conjunto de datos glass con la siguiente distribución **Clase minoritaria**: {6} — **Clase mayoritaria**: {0,1,2,3,4,5}

Haberman

Esta base de datos contiene casos de un estudio que se realizó entre 1958 y 1970 en el Hospital de la Universidad de Chicago Billings en la supervivencia de los pacientes que se habían sometido a cirugía para el cáncer de mama.

BD	haberman	Clases originales
		{positive, negative}

Iris

Esta base de datos es quizás la mejor que se encuentra en la literatura de reconocimiento de patrones. El conjunto de datos contiene 3 clases de 50 casos cada uno, donde cada clase se refiere a un tipo de planta iris. Una clase es linealmente separable de las otras 2, estas últimas no son linealmente separables una de otra.

BD		
	**iris**	**Clases originales** {Iris-setosa, Iris-versicolor, Iris-virginica}
	iris0	Versión desequilibrada del conjunto de datos iris con la siguiente distribución **Clase minoritaria** {iris-setosa} **Clase mayoritaria** {iris-virginica}

NewThyroid

Esta base de datos contiene información sobre la enfermedad de la tiroide. La tarea es detectar si un paciente dado es normal (1) o sufre de hipertiroidismo (2) o hipotiroidismo (3).

BD		
	**thyroid**	**Clases originales** {1,2,3}
	new-thyroid1	Versión desequilibrada del conjunto de datos thyroid con la siguiente distribución **Clase minoritaria** {2} **Clase mayoritaria** {1,3}
	new-thyroid2	Versión desequilibrada del conjunto de datos thyroid con la siguiente distribución **Clase minoritaria** {3} **Clase mayoritaria** {1,2}

Pageblocks

Esta base de datos contiene bloques de la disposición de página de un documento que ha sido detectada por un proceso de segmentación. La tarea consiste en determinar el tipo de bloque: Texto (1), la línea horizontal (2), Gráfico (3), línea vertical (4) o imagen (5).

BD		
	**page-blocks**	**Clases originales** {1, 2, 3, 4, 5}
	page-blocks0	Versión desequilibrada del conjunto de datos page-blocks con la siguiente distribución **Clase minoritaria** {1} **Clase mayoritaria** {2,3,4,5}

Pima

Esta base de datos contiene información de los pacientes de sexo femenino de al menos 21 años de edad y de herencia indígena.

BD	pima	Clases originales
		{tested_negative,tested_positive}
	Pima	Versión desequilibrada del conjunto de datos pima con la siguiente distribución Clase minoritaria: {positive} Clase mayoritaria: {negative}

Segment

Esta base de datos contiene instancias extraídas aleatoriamente de una base de datos de 7 imágenes al aire libre. Las imágenes eran segmentadas para crear una clasificación para cada píxel. Cada instancia codifica una región de 3x3. La tarea consiste en determinar el tipo de superficie de cada región.

BD	segment	Clases originales
		{1, 2, 3, 4, 5, 6, 7}
	segment0	Versión desequilibrada del conjunto de datos segment con la siguiente distribución Clase minoritaria: {1} Clase mayoritaria: {2,3,4,5,6,7}

Vehicle

El propósito de esta base de datos es clasificar una silueta dada como uno de los cuatro tipos de vehículo, utilizando un conjunto de características extraídas de la silueta. El vehículo puede ser visto desde uno de los muchos ángulos diferentes.

BD	vehicle	Clases originales
		{van, saab, bus, opel}
	vehicle0	Versión desequilibrada del conjunto de datos vehicle con la siguiente distribución Clase minoritaria: {van} Clase mayoritaria: {saab,bus,opel}
	vehicle1	Versión desequilibrada del conjunto de datos vehicle con la siguiente distribución Clase minoritaria: {saab} Clase mayoritaria: {van,bus,opel}
	vehicle2	Versión desequilibrada del conjunto de datos ecoli con la siguiente distribución Clase minoritaria: {bus} Clase mayoritaria: {saab,van,opel}
	vehicle3	Versión desequilibrada del conjunto de datos ecoli con la siguiente distribución Clase minoritaria: {opel} Clase mayoritaria: {van,saab,bus}

Vowel

Esta base de datos contiene información acerca del reconocimiento de las once vocales del Inglés Británico. Esta versión es una mezcla de las dos bases de datos originales preestablecidas en el repositorio UCI.

BD		
	**vowel**	**Clases originales** {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
	vowel0	Versión desequilibrada del conjunto de datos vowel con la siguiente distribución **Clase minoritaria**: {0} **Clase mayoritaria**: { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Wisconsim

Esta base de datos contiene casos de un estudio que se llevó a cabo en la Universidad de Hospitales de Wisconsin, acerca de los pacientes que se sometieron a cirugía para el cáncer de mama. La tarea consiste en determinar si el tumor detectado es benigno (2) o maligno (4).

BD		
	**wisconsin**	**Clases originales** {2,4}

Yeast

Esta base de datos contiene información sobre un conjunto de células de levadura. La tarea consiste en determinar el sitio de localización de cada célula.

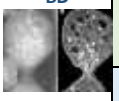
BD		
	**yeast**	**Clases originales** {MIT, NUC, CYT, ME1, ME2, ME3, EXC, VAC, POX, ERL}
	yeast1	Versión desequilibrada del conjunto de datos yeast con la siguiente distribución **Clase minoritaria**: {NUC} **Clase mayoritaria**: {MIT,CYT, ME1, ME2, ME3, EXC, VAC, POX, ERL}
	yeast-2_vs_8	Versión desequilibrada del conjunto de datos yeast con la siguiente distribución **Clase minoritaria**: {CYT} **Clase mayoritaria**: {ME2}
	yeast3	Versión desequilibrada del conjunto de datos yeast con la siguiente distribución **Clase minoritaria**: {ME3} **Clase mayoritaria**: {MIT, NUC, CYT, ME1, ME2, EXC, VAC, POX, ERL}
	yeast4	Versión desequilibrada del conjunto de datos yeast con la siguiente distribución **Clase minoritaria**: {ME2} **Clase mayoritaria**: {MIT, NUC, CYT, ME1, EXC, VAC, POX, ERL}
	yeast5	Versión desequilibrada del conjunto de datos yeast con la siguiente distribución **Clase minoritaria**: {ME1} **Clase mayoritaria**: {MIT, NUC, CYT, ME2, EXC, VAC, POX, ERL}
	yeast6	Versión desequilibrada del conjunto de datos yeast con la siguiente distribución **Clase minoritaria**: {EXC} **Clase mayoritaria**: {MIT, NUC, CYT, ME1, ME2, EXC, VAC, POX, ERL}

Tabla resumen de características

En la tabla inferior se muestran las principales características de las 33 bases de datos empleadas. Para cada base de datos se indica el número de instancias, número de atributos, nombre de las dos clases y el tanto por ciento de casos que pertenecen a la clase minoritaria.

Base de Datos	Numero instancias	Numero atributos	Clase(min,max)	% Clase Minoritaria
glass1	214	9	(build-win-non float-proc; remainder)	35,51
ecoli-0_vs_1	220	7	(im; cp)	35
wisconsin	683	9	(malignant; benign)	35
pima	768	8	(tested-positive; tested-negative)	34,84
iris0	150	4	(Iris-Setosa; remainder)	33,33
glass0	214	9	(build-win-float-proc; remainder)	32,71
yeast1	1484	8	(nuc; remainder)	28,91
vehicle1	846	18	(Saab; remainder)	28,37
vehicle2	846	18	(Bus; remainder)	28,37
vehicle3	846	18	(Opel; remainder)	28,37
haberman	306	3	(Die; Survive)	27,42
glass-0-1-2-3_vs_4-5-6	214	9	(non-window glass; remainder)	23,83
vehicle0	846	18	(Van; remainder)	23,64
ecoli1	336	7	(im; remainder)	22,92
new-thyroid2	215	5	(hypo; remainder)	16,89
new-thyroid1	215	5	(hyper; remainder)	16,28
ecoli2	336	7	(pp; remainder)	15,48
segment0	2308	19	(brickface; remainder)	14,26
glass6	214	9	(headlamps; remainder)	13,55
yeast3	1484	8	(me3; remainder)	10,98
ecoli3	336	7	(imU; remainder)	10,88
page-blocks0	5472	10	(remainder; text)	10,23
vowel0	988	13	(hid; remainder)	9,01
glass2	214	9	(Ve-win-float-proc; remainder)	8,78
ecoli4	336	7	(om; remainder)	6,74
glass4	214	9	(containers; remainder)	6,07
abalone9-18	731	8	(18; 9)	5,65
glass5	214	9	(tableware; remainder)	4,2
yeast-2_vs_8	482	8	(pox; cyt)	4,15
yeast4	1484	8	(me2; remainder)	3,43
yeast5	1484	8	(me1; remainder)	2,96
yeast6	1484	8	(exc; remainder)	2,49
abalone19	4174	8	(19; remainder)	0,77

5.5 Bases de datos KEEL

La aplicación o herramienta empleada por el grupo de investigación para crear las submuestras a partir de una muestra de entrenamiento, necesita una modificación en la programación de la misma con el objetivo de que sea capaz de leer o entender las bases de datos del formato KEEL. Esto es así porque el GureKS sólo admite hasta este momento bases de datos en formato de la UCI [AN07] y en el formato *.for* que es el propio de la aplicación.

Por lo tanto, el primer paso antes de ponernos manos a la obra en el análisis y primera toma de contacto con el código fuente, es estudiar cómo funciona el formato de datos empleado en las bases de datos KEEL y comprender los principios en los que se fundamenta.

Pero antes de esto conviene aclarar en qué consiste KEEL.

Ilustración 31 Interfaz principal de la herramienta KEEL

KEEL [AFLDGSH11] es una herramienta Java de código abierto para evaluar los algoritmos evolutivos de la minería de datos incluyendo regresión, clasificación y clustering. Contiene una gran colección de algoritmos clásicos para la extracción de conocimiento, técnicas de pre-procesado (selección de conjunto de entrenamiento, selección de características, discretización, métodos de imputación de valores perdidos, etc…), Inteligencia Computacional basado en algoritmos de aprendizaje, incluyendo el evolutivo de reglas basados en diferentes enfoques (Pittsburgh , Michigan e IRL, ...), y los modelos híbridos, tales como los sistemas difusos genéticos, redes neuronales evolutivas, etc. Nos permite realizar un análisis completo de cualquier modelo de aprendizaje en comparación con los existentes, incorporando un módulo de prueba estadística para la comparación.

El formato de las bases de datos con las que realizaremos el estudio es el propio del proyecto KEEL, si bien tiene similitudes con otro formato de datos ampliamente extendido como es el *Arff*. *Arff* es el formato de bases de datos de la extendida aplicación Weka (*Waikato Environment for Knowledge Analysis - Entorno para Análisis del Conocimiento de la Universidad de Waikato*), plataforma de software libre, la cual se centra en el aprendizaje automático y en la minería de datos.

El formato de los ficheros de datos para los algoritmos implementados en KEEL es una extensión del formato de datos presentado en WEKA, pero permitiendo una mayor flexibilidad al usuario.

La extensión del formato de los datos que define KEEL es *.dat*. Los ficheros *.dat* son archivos ASCII que describen una lista o conjunto de instancias a través de una serie de atributos que varían en número y tipo dependiendo del archivo.

El archivo siempre consta de dos partes: la cabecera y los datos o matriz de datos.

La sección de la cabecera: En esta sección se define:

- Nombre de la relación: El nombre de la base de datos. Forma parte de la primera línea del fichero y es obligatorio.

```
@Relation <nombre>
```

El formato que debe seguir el <nombre> ha de ser de tipo cadena de caracteres.

- Declaración de atributos: La lista de atributos y el tipo que se pasan a describir a continuación.

```
@Attribute <nombre> <tipo>
```

Como sucede con el nombre de la relación, el nombre del atributo también debe de ser del tipo cadena de caracteres. Definir el @attribute nombre y tipo es obligatorio.

Los tipos de atributos definidos en KEEL son los siguientes:

- Integer [min, max]
- Real [min, max]

El símbolo de separación de números decimales es un punto en lugar de una coma.

En el caso de atributos de tipo integer y real, no es obligatorio indicar el intervalo de valores, si vienen lo hacen separados por coma.

- Lista de valores de la forma {<valor0>, <valor1>, …, <valorN>}

Consta de un conjunto de valores posibles separados por comas (caracteres o cadenas), que puede tomar el atributo.

- Declaración de entradas y salidas: Sólo si es necesario en el problema, en el caso de nuestro estudio sí que lo es.

```
@inputs < nombre1 >, < nombre2 >, ... , < nombreX >
@outputs < nombre1 >, < nombre2, ... , < nombreY >
```

Mediante @inputs indicamos qué atributos forman parte de la base de datos, si un atributo no consta en la línea inputs es que es ignorado de la misma.

Mediante @outputs indicamos los atributos de salida. En problemas como éste de clasificación es necesario, ya que mediante outputs indicamos la variable clase.

La sección de datos: Los datos se encuentran agrupados en líneas, formando lo que podría calificarse como una matriz de datos, cada una de las líneas forma una instancia de la base de datos. Los datos se encuentran separados por comas. Como sucede con el nombre y la definición de atributos es una sección obligatoria y sigue el orden en el que se definieron los atributos.

```
@Data
```

Si necesitamos indicar un valor nulo en los datos emplearemos < null >. Por último destacar que los identificadores de los atributos pueden contener espacios y que su tamaño máximo es de 12 caracteres.

Estos ficheros se almacenarán, por defecto, con la extensión *.dat*. Un ejemplo de este tipo de ficheros puede ser el siguiente:

```
@relacion paint

@attribute colour {yellow, white, black}

@attribute amount integer [1, 10]

@attribute density real [0.1, 2.5]

@inputs colour, amount

@outputs density

@data

yellow, 4, 1.2

black, 1, 2.1

yellow, 2, 0.8

white, 8, 0.9
```

5.6 Bases de datos del formato UCI

El formato de datos de la UCI también conocido como formato de datos C4.5, es una modificación del programa famoso y ampliamente utilizado de Ross Quinlan C4.5.

La característica principal -que define este formato de bases de datos-, es que se diferencia la parte de la definición de tipos o cabecera y por otro lado, separado en otro fichero, encontramos los datos.

Diferenciamos entonces, dos ficheros, uno con extensión *.names* y el referente a los datos se encuentran en el fichero con extensión *.data*.

El archivo *.names* contiene una serie de entradas que describen las clases, atributos y valores del conjunto de datos, cada línea del fichero se termina con un punto, pero éste se puede omitir si es el último carácter de una línea.

El fichero *.names* contiene la cabecera: En esta sección se define:

- Nombre de las clases: El primer registro del archivo contiene los nombres de las clases, separados por comas y termina con un punto

- Declaración de atributos: Se definen los atributos en el orden en el que aparecen en el fichero de datos, siguiendo el siguiente formato:

```
<at-name>: <at-type>.
```

El nombre del atributo es un identificador seguido de dos puntos y el tipo del atributo. Cada nombre se compone de una cadena de caracteres sin comas, comillas o dos puntos, puede contener un punto pero éste debe ser seguido de un espacio en blanco.

El tipo de atributo debe ser:

- continuous: Si el atributo tiene valores continuos.
- discrete <n>: la palabra "discreto" seguida de un número entero que indica cuántos valores puede tomar el atributo.
- <list of identifiers>: Atributo discreto con los valores. Los identificadores deben estar separados por comas.
- ignore: Indica que este atributo debe ser ignorado.
- | (barra vertical): Significa que el resto de la línea debe ser considerada como un comentario.

Un ejemplo de este formato para definir la cabecera podría ser el siguiente.

```
| Firstly the name of classes

good, bad.

|Then the attributes
```

```
dur: continuous.
wage1: continuous.
wage2: continuous.
wage3: continuous.
cola: tc, none, tcf.
hours: continuous.
pension: empl contr, ret allw, none.
stby_pay: continuous.
shift_diff: continuous.
educ_allw: yes, no.
holidays: continuous.
vacation: average, generous, below average.
lngtrm_disabil: yes, no.
dntl_ins: half, none, full.
bereavement: yes, no.
empl_hplan: half, full, none.
```

El fichero *.data* se encuentran los datos:

El archivo contiene una línea por instancia de la base de datos. Cada una contiene los valores de los atributos ordenados de acuerdo a lo definido en el fichero *.names*. Los valores que toman los atributos para cada instancia se encuentran separados por comas.

Como sucede en el formato de bases de datos KEEL, la ausencia del valor de un atributo se indica mediante el carácter '?'.

A continuación se encuentra el fichero *.data* asociado al *.names* anterior.

```
2,5.0,4.0,?,none,37,?,?,5,no,11,below average,yes,full,yes,full,good
3,2.0,2.5,?,?,35,none,?,?,?,10,average,?,?,yes,full,bad
3,4.5,4.5,5.0,none,40,?,?,?,no,11,average,?,half,?,?,good
3,3.0,2.0,2.5,tc,40,none,?,5,no,10,below average,yes,half,yes,full,bad
```

5.7 Adaptación de la herramienta GureKS al formato KEEL

Como se ha introducido al comienzo de este capítulo, se hace necesario realizar unas modificaciones a nivel de programación en la herramienta GureKS, con el fin de dotarla de capacidad para leer bases de datos en formato KEEL, y poder generar los ficheros de salida necesarios en las próximas fases de esta experimentación.

Las acciones que hay que llevar a cabo podrían resumirse en 2 puntos:

1-Lectura de la base de datos KEEL y almacenamiento en la estructura de datos.

- Lectura del formato de datos
- Lectura de la matriz de datos
- Creación de la muestra

2-Generación de ficheros *.names* y *.data.*

Lenguaje de programación empleado

El lenguaje C++ es una extensión del lenguaje C diseñado a mediados de los años 1980 por Bjarne Stroustrup para soportar el paradigma de la programación orientada a objetos.

El punto de partida de este paradigma se fundamenta en la idea de que el mundo está formado por objetos. Entendiendo por objetos a la entidad significativa existente en el entorno. El entorno se concibe como el conjunto de relaciones entre los distintos objetos. Otro concepto importante en este paradigma es la clase, que puede verse como un conjunto de objetos distintos del mismo tipo. Estos tienen un estado, características o propiedades que lo componen en un momento dado y unos métodos asociados que son las operaciones realizables por el objeto.

En la actualidad, el C++ es un lenguaje versátil, potente y general. Ofrece la ventaja de permitir el desarrollo de aplicaciones modulares con código reutilizable.

En cuanto al número de ficheros que maneja este lenguaje son principalmente dos, los ficheros de cabecera con extensión *.h* y *.hpp* y los ficheros fuentes, de extensión *.cpp*.

En los ficheros de cabecera se encuentran las declaraciones de constantes, variables y funciones de las que consta el módulo, así como llamadas a otros archivos de encabezado necesarios, mientras que en los ficheros fuente se implementa el código para las funciones del módulo declaradas en los archivos de cabecera.

El elemento fundamental de las aplicaciones Windows son las ventanas o interfaz gráfica. El API de Windows dispone de todas las funciones necesarias para la gestión de las funcionalidades de las ventanas. Para gestionar la creación y uso de la interfaz gráfica emplearemos la librería MFC de Microsoft.

¿Qué es MFC?

La librería MFC es una librería de clases, que encapsula el comportamiento del sistema de ventanas, descargando de trabajo al programador y permitiéndole centrarse en la funcionalidad específica de su programa.

Las clases en la biblioteca MFC están escritas en el lenguaje de programación C++. Hay clases MFC de biblioteca para todos los elementos de la interfaz gráfica de usuario (ventanas, marcos, menús, barras de herramientas, barras de estado, etc.), para la

creación de interfaces de base de datos, para controlar los eventos, como mensajes de otras aplicaciones, para el manejo del teclado y entrada del ratón, y para crear ActiveX control.

En el anexo 3 se encuentra una pequeña introducción a Microsoft Visual C++ 6.0 con la que se programará.

5.7.1 Lectura de la BD KEEL y almacenamiento en la estructura de datos

Una vez aclarados los aspectos fundamentales del desarrollo de aplicaciones en C++, es hora de comenzar el trabajo de implementación. Detallado el formato de bases de datos KEEL y el empleado para la salida UCI; el siguiente paso es conocer la estructura de datos y la forma en la que está programado el GureKS, para deducir o descubrir cuáles son las funciones o procedimientos que requieren una modificación.

La complejidad inherente que se encuentra uno al comienzo, es que se trata de una aplicación que se ha ido actualizando o ampliando año tras año por diferentes personas y este hecho acompañado de la escasa documentación supone una primera barrera.

La ayuda proporcionada por Txus en forma de reuniones ha sido vital en este punto del proyecto.

Ilustración 32 Conjunto de clases que forman la aplicacion GureKS

Además la envergadura o tamaño de la aplicación es considerable con lo que acostumbra uno a encontrarse en la carrera.

Se pretende modificar el programa para que sea capaz de generar submuestras como hasta ahora (con extensión *.lid*), pero partiendo de bases de datos en formato KEEL y además realice una "traducción" de las mismas al formato UCI.

Para ello, es primordial estudiar las clases y detectar cuáles están implicadas en la tarea que queremos concluir con éxito.

Como se deprende de la ilustración 32 el proyecto GureKS está compuesto de 76 clases, entre las que destacan:

- Las cabeceras en las que se definen la representación de los datos que están involucrados, es decir, atributos*: _attribute_value*.
- Las clases empleadas para almacenar las variables que componen las muestras, tanto las que se emplearán en la construcción de submuestras mediante *CVable*

como las que realmente contiene el fichero de datos *CVableF* (sin variables ignoradas).

- Las relacionadas con la gestión de mensajes o diálogos: *CAboutDlg*
- Las clases implicadas en la preparación de la herramienta: *CArbolApp*
- Las propias para la carga de muestras *CModuloCargarMuestra*, generación de submuestras y testeo del clasificador *CModuloTest* desde scripts
- Las implicadas en la construcción de muestras: *CMuestra,CMuestraUCI*
- Las relacionadas con la lectura y ejecución de scripts *CLeeScript*
- Las ya implementadas para manejo de formatos *CFicLeeFor, CFormatoDatosUci*
- Las clases necesarias para la integración del lexer y parser, que posibilitan el uso de la gramática reconocedora: *mylexer, myparser*.

Tras un análisis y unas cuantas reuniones con Txus, se deduce que, la mayoría de las clases implicadas en este proyecto no deben ser modificadas, pues son independientes del cometido que se pretende abordar, que no es otro que "añadir" un nuevo formato al funcionamiento de GureKS.

Como ya se ha expuesto, se empleará el GureKS únicamente para la creación de submuestras a un determinado porcentaje de distribución de clases, con una técnica de remuestreo concreta, dejando el cometido de creación de árboles y las operaciones relacionadas con éstos en manos de Haritza. Las técnicas de remuestreo y los métodos que posibilitan la creación de submuestras a un determinado porcentaje de clases se deben dejar sin alterar, pues, en nada tiene qué ver el formato de los datos en el funcionamiento de estos métodos.

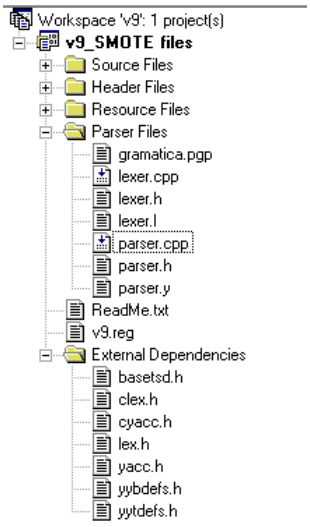


Ilustración 33 Workspace del proyecto GureKS

Además, las clases relacionadas con la construcción, gestión y operaciones propias de los árboles de clasificación, como la poda o la visualización del árbol no nos interesan para nada en este momento.

Lo mismo sucede con las clases implicadas en la lectura y ejecución de scripts, o configuración de la herramienta y los parámetros propios, que se continuarán usando como siempre.

Por otra parte, las clases que especifican el formato de las variables o atributos no deben de ser modificadas en absoluto, ya que únicamente se pretende añadir un nuevo formato a la aplicación. Aunque conocer el funcionamiento de las mismas se antoja necesario para poder leer los atributos y tipo de los mismos.

GureKS dispone de dos clases para la gestión de atributos o variables y conviene diferenciar entre ellas, CVable y CVableF. CVable es la clase encargada de crear objetos de tipo Vable para almacenar datos del atributo leído de la muestra, la estructura está formada por distintos campos, los más importantes se detallan a continuación:

```
public:
//       Formato de la Variable
// [Var] ----------------------
   CString Nom;    // Nombre de la Variable
   int Long;       // N bytes a leer en el fichero
   BOOL VD;        // Variable Dependiente
   BOOL Ign;       // Ignorar la Variable en el analisis
   BOOL Ord;       // Orden
   long Min,Max;   // Valores Mmo y Mḥmo
  // float MinR,MaxR;
   CWordArray Vec_Cat; // Vector Valores Distintos = Categorias
   CStringArray Vec_CatEtiq; // Vector de Etiquetas de las Categorias
```

Variable booleana Vd que indica si el atributo leído sea de tipo clase.

Ign indica si la variable debe ser ignorada del análisis.

Para atributos que contienen etiquetas o enumerados se almacenan en los vectores Vec_CatEtiq y en Vec_Cat el número de categorías.

El tipo de datos CVableF es más sencillo.

```
public:
//       Formato de la Variable
// [Var] ----------------------
   CString Nom;    // Nombre de la Variable
   int Long;       // N bytes a leer en el fichero (si el formato es
fijo)
   BOOL Ign;       // Ignorar la Variable en el analisis
```

Aunque puedan parecer iguales, se distinguen en la utilización que se les da, en CVableF están todas las variables que se encuentran en el fichero de formato, mientras que en CVable sólo se crean las variables que se emplearán para la generación de submuestras y que no han sido ignoradas. GureKS diferencia entre todas las variables y las que realmente se emplearán. El objetivo es que, aunque haya algunas que se ignorarán, es necesario conocer su formato (mínimo) para poderlas leer y distinguirlas de las que serán usadas.

Aclarada la primera estructura empleada para guardar los atributos que vayamos leyendo, los siguientes esfuerzos se enmarcan en encontrar en qué clases gestiona GureKS la lectura del formato de datos y los datos propiamente dichos.

El trabajo de programación se desarrolla únicamente en dos clases, CarbolDoc y CMuestra.

CarbolDoc contiene la definición de una gran cantidad de estructuras, destacan:

- La estructura para gestionar el árbol consolidado
- El vector de submuestras para la consolidación del árbol
- Vector formato de variable de datos

```
/ ------------------------------------------------------------------------
```

```
//  Vector Formato Variables Datos (Fichero '.for')
// -------------------------------------------------------------------------------
    int Vables_Ign;
        // SIN_IGNORAR  // Todos los datos del fichero son almacenados en Tab_Datos
        // IGNORANDO    // S□se almacenan los datos asociados a las vables NO ignoradas
    CString Nom_Fich_For; // Nombre del Fichero de Formato
    CObArray Vec_VablesF; // Vector con todas las Vables. del fichero de Formatos
    int T_Var;            // Num. Total de Variables
    CObArray Vec_Vables;  // Vector con todas las Vables. NO Ignoradas
    int N_Var;            // Num. de Variables No Ignoradas
    BYTE n_Fic_Fmt;       // Formato de los Campos a leer: (0) LIBRE | (1) FIJO
    int Ident_VD;         // Identificador de la Vable. Dependiente
    int Ident_VD_Tot_Var; // Idem. pero respecto al vector Vec_VablesF
    int NumCatVD;         // Nmero de Categorde la VD

    //          - 0 ,si la vable es discreta (tipo T Tab Datos) y estar□uardada en
Tab_Datos
    //          - 1 ,si la vable es continua (tipo T_Tab_DatosCont) y estar□uardada en
m_Tab_DatosCont
    // ***** Esto podrser generalizable con un array de Tablas de Datos de distintos
tipo base!!!!!
    CMiArrayGenerico<BYTE> m_vLocalizadorTabDatos;
    // Vecor que indica cual es el ice de cada vable en su tabla de datos
correspondiente
    CMiArrayGenerico<BYTE> m_vLocalizadorVableTabDatos;
    int T_VarNoCont;    // Num. Total de Variables NO CONTINUAS (Discretas e Ignoradas)
    int T_VarCont;      // Num. Total de Variables CONTINUAS
    CArbolC45 *m_pArbolC45;
```

- Vector de muestras de test
- Las estructuras necesarias para la impresión del árbol en fichero,

y los métodos que operan sobre los mismos.

CMuestra contiene las estructuras y métodos necesarios para la creación y gestión de muestras que contendrán los datos, mediante esta clase se crean las muestras a partir de los ficheros de datos y se almacenan para su posterior uso en la estructura que hace uso de la clase CMiMatrizGenerica.

```
CMiMatrizGenerica<T_Tab_Datos> Tab_Datos
```

5.7.1.1 Lectura del formato de datos

Analizando detalladamente la programación de GureKS, se aprecia que en estos momentos la aplicación gestiona dos tipos de formato:

- El propio de la aplicación e involucrado en la creación de los ficheros *.lid* o ficheros de casos de identificadores denominado *.for*
- El formato de datos UCI.

Es de vital importancia estudiar qué operaciones se realizan para seguir las mismas decisiones de implementación y poder añadir el tercer formato, el *.data* de KEEL.

La lectura del formato que siguen los datos, y la lectura de los mismos se realiza en dos funciones diferenciadas, y se encuentran en las dos clases anteriormente expuestas.

A pesar de que en el formato de datos KEEL, los ficheros de datos y formato están integrados en un único fichero y se podría realizar el proceso en la misma función,

se ha decidido realizarla en dos fases -como se ha venido desarrollando hasta ahora- para no variar la metodología empleada con los otros dos formatos.

La función encargada de la lectura del formato en CArbolDoc es:

```
        int CArbolDoc::Leer_Fic_Formato()

       /* Lee el fichero de formato de los datos y configuracion de vables.
(*.FOR
```

La función comienza con la inicialización y declaración de variables necesarias para almacenar el formato: Contadores para las variables

```
int  cont_var,Tot_Var; // Contador de Variables No Ignoradas y Totales
int i_vableNoCont,i_vableCont;
```

Buffer para almacenar las líneas del fichero que se van leyendo

```
        char lin[L_LIN]/*,lin_old[L_LIN]*/; // Buffer de Lectura (Linea fichero)
```

Variables para almacenar atributos

```
        CVable *p_Vable,*p_Vable_VD;
        CVableF *p_VableF;
```

Al abrir el fichero formato se mira la extensión que éste posee, aquí se realiza la diferencia entre formato GureKS, UCI O KEEL. Si la extensión leída es *.dat* activamos una variable booleana indicando que el formato de datos es el de KEEL, impidiendo que ejecute el código correspondiente a los otros dos formatos.

Entramos en un if que inicializa las variables necesarias, y se empieza a leer el contenido del fichero:

- En la primera línea se tiene que encontrar el token *@relation* con el nombre de la relación y se continúa leyendo el fichero. En caso de no ser así se muestra mensaje de error y aborta el proceso.

- Seguidamente viene la serie de atributos que forman la muestra, también obligatorio. Mientras se siga leyendo el token *@atributte* se continua en un bucle en el que se comienza a procesar la definición de atributos.

Se almacena el nombre del atributo y en función del tipo del mismo se diferencia el proceso a seguir, podemos encontrar atributos enteros, reales o enumerados-discretos, cada uno con sus particularidades.

  - En caso de encontrar un atributo entero o real, avanza la lectura del contenido del buffer y almacena los límites que definen el conjunto que vienen entre [].

  - Finalmente si lo que encuentra es enumerado o discreto, en el vector Vec_Cat se almacena el índice de la etiqueta que será el valor que se guardará en la tabla de datos asociada a la etiqueta leída y en Vec_Cat_Etiq las etiquetas

propiamente dichas, reservando la posición cero del mismo para indicar valores *missing* (como los otros dos formatos).

Existe una estructura destacada cuyo cometido es indicar el tipo de datos de la variable, su funcionamiento es el siguiente:

```
CMiArrayGenerico<BYTE> m_vLocalizadorTabDatos; - 0 ,si la vable es discreta (tipo
T Tab Datos) y estar guardada en Tab Datos
    //          - 1 ,si la vable es continua (tipo T_Tab_DatosCont) y
estar guardada en m Tab DatosCont

CMiArrayGenerico<BYTE> m_vLocalizadorVableTabDatos; // Vecor que indica cual es
el ice de cada vable en su tabla de datos correspondiente
```

Después de la lectura de cada atributo, se almacena el objeto de tipo CVable, y CVableF empleado para colectar la información del atributo, en un vector creado para este efecto, definido como variable global y de nombre Vec_Vables. También se actualizan todos los contadores, encargados de contar el número de atributos continuos, los no continuos, número total de variables….

En estos momentos todavía no se sabe si existen variables ignoradas o cual es la variable clase, para este cometido, el formato posee los token *@inputs* y *@outputs.*

- Se almacenan en un vector temporal los atributos que vienen precedidos del token *@inputs*. Al finalizar se recorre éste comparándolo con el que contiene las variables ya creadas (Vec_Vables). Si coinciden no se ignora. Si al recorrer entero el vector temporal no hemos encontrado la variable debe ser ignorada, actualizando los campos correspondientes y eliminándola del vector.

- Con *@outputs* se indica la variable clase, se lee el nombre y se busca en el vector de variables la que posee el mismo nombre para modificar el atributo VD que indica que es la variable clase. En el caso de encontrar más de una variable después del token @outputs se muestra un mensaje de error. El motivo es que en este tipo de problemas de clasificación la variable clase sólo puede ser una.

En este momento, se ha leído la parte del fichero correspondiente al formato y ya se han rellenado las estructuras que definen y gestionan el tipo de datos de GureKS.

5.7.1.2 Lectura de la matriz de datos

Para leer los datos, GureKS hace uso de una función que se encuentra definida y programada en CMuestra. Almacena los datos en dos estructuras, una para los datos continuos, otra para los datos enteros.

Pero para almacenar estos datos antes es necesario hacer uso del constructor de la clase para crear un objeto de tipo CMuestra con los parámetros adecuados. Para esta función nos apoyamos en el método:

```
CMiMatrizGenerica<T_Tab_Datos> Tab_DatosCMuestra *
CMuestra::Construir_Muestra(CString nom_fic_dat,CArbolDoc *pDoc,int
fil_x_col)
```

Esta función necesita ser ampliada y es la encargada de crear la muestra; se encuentra en la misma clase y consta de la siguiente definición:

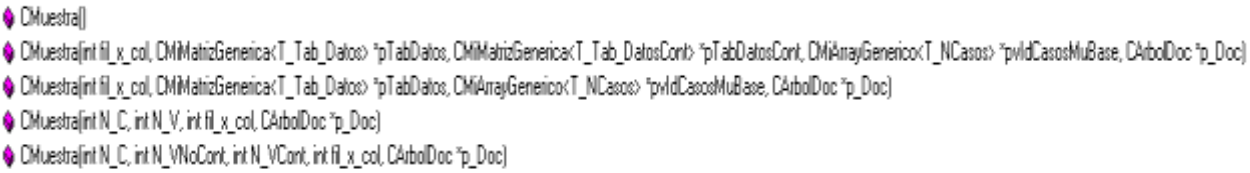
La función comienza obteniendo el número de casos que componen el fichero de datos, para ello procede de la siguiente manera: Lee el contenido del fichero línea por línea a través de un bucle, incrementando un contador hasta llegar al final del fichero desde el momento en el que encuentra el token *@data*.

Al final de la función se diferencia el formato que sigue la muestra que se va a crear, en caso de ser el tipo KEEL se emplea el código:

```
     if(pDoc->m_FormatoKEEL){
        if(pDoc->Vables_Ign==SIN_IGNORAR)
            p_mu = new CMuestra(num_casos,pDoc->T_VarNoCont,pDoc-
>T_VarCont,C_CASOSxVABLES,pDoc);
        else
            p_mu = new CMuestra(num_casos,pDoc->T_VarNoCont,pDoc-
>T_VarCont,fil_x_col,pDoc);
        return p_mu;
    }

    else{
```

Esta función es una de los 4 posibles constructores de la propia clase para crear un objeto de tipo muestra:



En la que emplearemos nosotros, se indica el número de casos que componen la muestra, número de atributos continuos, atributos no continuos, tipo de matriz que se desea generar (acceso a memoria por casos (C_CASOSxVABLE) o por variables (C_VABLESxCASOS)), y el objeto de tipo CarbolDoc.

Una vez establecidos estos parámetros básicos para la creación de la muestra, la aplicación ya está preparada para realizar la lectura de los datos.

La función encargada en GureKS de leer el fichero de los datos y almacenarlos en la estructura es:

```
int CMuestra::Leer_Fic_Datos(CString nom_fich, BOOL Modificar_Vec_Cat)
```

Lee los datos del fichero según el formato y los guarda en la Tabla de Datos (Tab_Datos). Establece, a su vez, la estructura (m_vMatIdentCasoVD) con los vectores de identificadores de casos de cada una de las categorías de la VD.

Para no complicar demasiado la función y diferenciar el proceso de lectura de los datos en KEEL se define una nueva función:

```
int CMuestra::Leer_Fic_Datos_KEEL(CString nom_fich, BOOL Modificar_Vec_Cat)
```

Como punto de partida se establecen las estructuras para almacenar los datos leídos del fichero.

```
// Traspasar los datos a la Tabla de Datos
    // ======================================

    CMiArrayGenericoBase<T_Tab_Datos> *p_fila;
    CMiArrayGenericoBase<T_Tab_DatosCont> *p_filaCont;
    CMiArrayGenericoBase<T_Tab_Datos> *p_fila1;//Para el .names
    CMiArrayGenericoBase<T_Tab_DatosCont> *p_filaCont1;
```

Como los datos en KEEL se encuentran en el mismo fichero que el formato, una vez abierto éste, es necesario, moverse por el mismo hasta encontrar el token *@data* a través de un bucle. Una vez alcanzado este punto comienza la lectura de datos.

Por cada una de las líneas que lo componen, y hasta que no llegue hasta el último caso indicado por N_Casos, se extraen tantos valores como número de atributos tiene el fichero. Cada vez que se copia un valor se almacena en la estructura correspondiente:

GureKS distingue entre las estructuras empleadas para guardar los datos dependiendo de la naturaleza de los mismos.

Estructura `CMiArrayGenericoBase<T_Tab_Datos> *p_fila;` para datos enteros o discretos.

Estructura CMiArrayGenericoBase<T_Tab_DatosCont> *p_filaCont; para almacenar los datos reales.

Para saber de qué tipo es el valor que esperamos leer, se debe consultar el valor del vector m_vLocalizadorTabDatos[var] para la posición que tiene ese atributo concreto en el fichero.

Al finalizar el bucle se encuentran los datos almacenados en esas estructuras y se cierra el fichero de datos.

5.7.2 Generación de ficheros *.names* y *. data*

En este momento, la aplicación ya está preparada para leer datos en formato KEEL y ser capaz de generar las submuestras, pero para generar los árboles a partir de las

muestras se hace necesario obtener los datos y formato en dos ficheros por separado, respetando el formato UCI.

Para la generación, poda y test de árboles en Haritza y como bien se describirá en este mismo capítulo, es necesario que los datos se encuentren en formato *.data* y el formato que describen los mismos posea extensión *.names.*

Simplificando el proceso hay que traducir el fichero *.dat* en *.data* para los datos y en *.names* para el formato.

Aprovechando que los datos se encuentran almacenados en las estructuras anteriormente expuestas, únicamente es necesario recorrer estas dos estructuras respetando el orden en que se encuentran los atributos, es decir, por cada atributo que compone la muestra, se mira el tipo de datos que contiene mediante el vector vLocalizadorTabDatos[var], y en función del tipo de atributo se "vuelca" el contenido bien de la estructura de datos entero y discreto, o bien, de la de los reales a un fichero nuevo creado con extensión *.data*.

Una vez finalizado el proceso se cierra el fichero creado y se crea uno nuevo que tendrá como objetivo, definir el formato que sigue la muestra en formato *.names*.

Como sucede en el caso de la generación del fichero *.data*, podemos aprovechar el trabajo realizado hasta ahora y en vez de releer el fichero original, obtener la información a partir de las estructuras que se ha definido en la página 75.

Como define el formato de datos UCI la primera línea de un fichero *.names* la componen las etiquetas de la clase.

Se accede directamente a la variable que se encuentra en la posición indicada por pDoc->Ident_VD en el vector de variables y obtenemos las etiquetas accediendo al vector de etiquetas. Posteriormente se escriben en el fichero.

El siguiente paso es escribir los atributos y el tipo de los mismos. La metodología empleada es la misma; se accede a través del vector de variables a cada una de las variables que lo componen menos la última, que ya ha sido tratada como variable dependiente o clase, se mira el tipo de cada uno de ellas y se distinguen tres variantes, también en necesario comprobar si se trata de una variable ignorada o no, ya que UCI lo indica en el fichero *.names*. Esto se realiza consultando el valor del campo booleano Ign, que poseen los objetos de tipo CVableF. Junto al nombre del atributo se indica el tipo:

- En el caso de ser una variable de tipo entero, se indica mediante la palabra *Discrete 0.* De esta manera se interpreta que el número de valores es desconocido.
- Para el caso de variables enumeradas, se escriben las etiquetas o posibles valores enumerados separados por coma y la última acabada en punto.
- Para el tercer caso, variables reales se indica mediante *continuous.*

Una vez recorrido el vector de variables hasta la penúltima posición del mismo se da por concluida la generación del *.names* y se cierra el fichero.

La adaptación de la plataforma al formato de datos KEEL ha llegado a su fin en este momento.

5.8 Proceso técnico y software empleado

Una vez expuestos y detallados los esfuerzos realizados para adaptar la plataforma GureKS al formato de bases de datos KEEL, en esta sección se describen las distintas herramientas o aplicaciones que se emplearán para llevar a cabo la experimentación, su funcionamiento y el proceso empleado para:

- Construir las submuestras a partir de la muestra de entrenamiento mediante GureKS.
- La carga de submuestras, generación del árbol de clasificación, poda y testeo mediante Haritza.
- La generación por cada una de las bases de datos del fichero *.aue* con las medidas de bondad del clasificador mediante Kurbak.
- Cálculo de medias para las bases de datos del estudio mediante Excel y Visual Basic Editor.

5.8.1 GureKS

Como se ha descrito en el capítulo referente al algoritmo CTC, éste basa su fundamento en dos partes diferenciadas, la creación de submuestras y por otro lado, la construcción del árbol a partir de éstas.

Para la creación de submuestras a partir de la muestra de entrenamiento se hará uso de la herramienta GureKS.

Introducción y fundamentos

Ya adaptada dicha herramienta para la lectura de bases de datos KEEL, es el momento de generar las submuestras necesarias para la futura creación del árbol consolidado.

GureKS es una herramienta diseñada principalmente para generar submuestras a partir de una muestra dada con la distribución de clases concreta que se desee y mediante la técnica de remuestreo que se considere más adecuada en la experimentación que se quiera llevar a cabo.

En estos momentos ofrece la posibilidad de realizar un remuestreo oversampling, undersampling, y dentro del remuestreo oversampling ofrece técnicas modernas y

efectivas como son los métodos SMOTE, de los cuales ya se ha hablado anteriormente y de los que encontramos un estudio en [AMP10].

La apariencia o interfaz gráfica de la herramienta es sencilla y liviana y se encuentra una ilustración de la misma justo debajo de estas líneas.

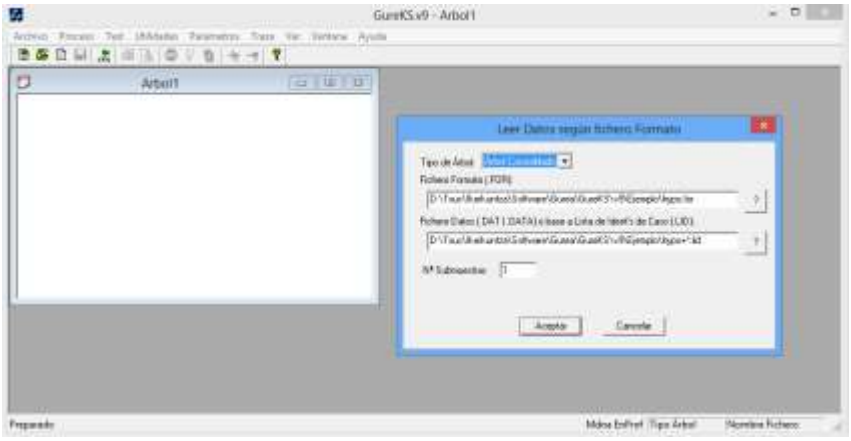


Ilustración 34 Menú principal de la herramienta GureKS

Como se desprende de esta ilustración, la herramienta consta de un menú sencillo y una seria de botones o accesos directos que facilitan la utilización.

Cabe destacar que como sucede con la aplicación Haritza, encontramos dos posibles formas de ejecución:

- Una que se puede calificar como manual, en la que es indispensable la interacción entre el usuario y aplicación.
- La otra que podríamos denominar como la ejecución mediante scripts y ficheros *.bat* (ejecutables MS-DOS) que posibilita la ejecución automática de la experimentación. Esta es la que emplearemos.

Funcionamiento de la aplicación

Para realizar una experimentación de manera manual, es necesario hacer uso de los menús que ofrece esta aplicación para cargar las muestras y configurar el resto de parámetros involucrados en la generación de submuestras.

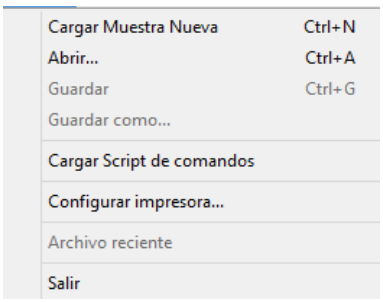


Ilustración 35 Opciones del menú archivo de la aplicación GureKS

El menú archivo es empleado para (véase ilustración 35):

- Cargar las muestras: esta acción también se puede realizar pulsando en el primer botón o icono que se encuentra justo debajo de archivo y que tiene apariencia de un árbol en una hoja blanca.

- Cargar scripts de comandos: estos scripts son una combinación de comandos que permiten realizar la experimentación de una manera autónoma gracias a la gramática generadora.
- Configurar impresora: cuadro de diálogo común que encontramos en cualquier aplicación que haga uso de impresora.

El menú utilidades es el núcleo de esta aplicación posibilita (Véase ilustración 36):

- La generación de submuestas a partir de una muestra.
- La generación de fichero con identificadores de caso.

Ilustración 36 Opciones para la generación de submuestras

El resto de menús son el menú Help y menú Ver, que permite configurar la apariencia visual de la aplicación.

El motivo por el que se decide usar la aplicación mediante scripts y ficheros ejecutables *.bat* generados a través del lenguaje VisualBasicScripting, es que realizar este proceso manualmente puede resultar tedioso además de una pérdida de tiempo, ya que si por cada base de datos construimos 25 conjuntos de submuestras como es el caso de este estudio y además esto se realiza por cada una de las 33 bases de datos, el hecho de tener que hacerlo manualmente resultaría "infernal".

Automatización del proceso

Con los scripts de comandos, y los ficheros ejecutables MS-DOS, conseguimos realizar una experimentación completa de una manera desatendida y solo tenemos que esperar los resultados mientras podemos emplear ese tiempo para seguir preparando el resto de experimentaciones.

La ejecución automática o desatendida se consigue gracias a los ficheros *.bat* y los scripts generados mediante la suite ofimática Excel y su potente herramienta de programación Visual Basic Scripting, detallada en la sección 5.8.4 Visual Basic Scripting y Excel, encontramos un ejemplo de estos dos ficheros a continuación.

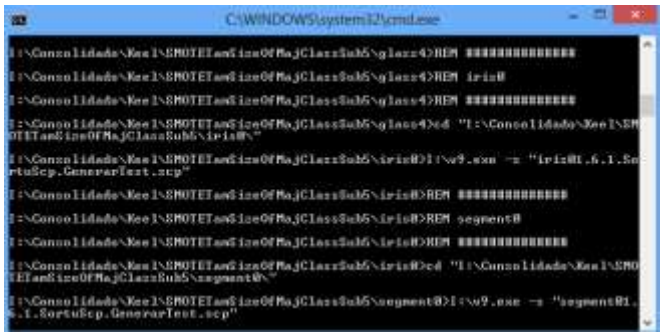


Ilustración 37 Fichero ejecutable .bat en MS-DOS

Se observa que mediante el comando *cd* dos, se sitúa en la carpeta de la base datos y una vez nos encontramos ahí, mediante el comando *I:\v9.exe -s "ecoli 0_vs_11.6.1.SortuScp. GenerarTest.scp"* se le pasa como parámetro el script en que se encentra los comandos que especifican la experimentación.

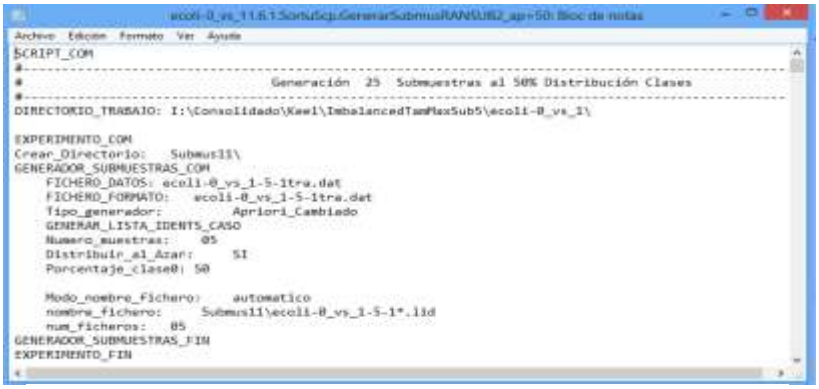


Ilustración 38 Script para generación de 25 submuestras de la bd ecoli-0_vs-1

En el script de la ilustración 38 se encuentran los comandos necesarios para la creación de 5 submuestras a partir de la primera muestra de entrenamiento de la base de datos ecoli-0_vs-1. El experimento se detalla entre los token EXPERIMENTO_COM y EXPERIMENTO_FIN y los parámetros que definen el tipo de submuestras que se obtendrán vienen entre los token GENERADOR_SUBMUESTRAS_COM y GENERADOR_SUBMUESTRAS_FIN.

Entre estos encontramos especificado el FICHERO_DATOS, FICHERO FORMATO (el mismo), el número de submuestras que se desean obtener, con qué distribución y con qué tipo de reemuestreo, además de indicar que se quiere generar el fichero de identificadores de caso (*.lid*).

Proceso llevado a cabo

En esta primera fase del estudio se emplea el GureKS con dos cometidos que se aprecian gráficamente en las ilustraciones 39 y 40 y que se detallan a continuación:

1. La creación del conjunto de submuestras a partir de una muestra de entrenamiento y asociado a éstas la generación del fichero *.data* y *.names* de la muestra leída.

2. La creación del fichero *.data* asociado a las muestras de test, el *.names* no se genera ya que es común entre las muestras de entrenamiento y las de test.

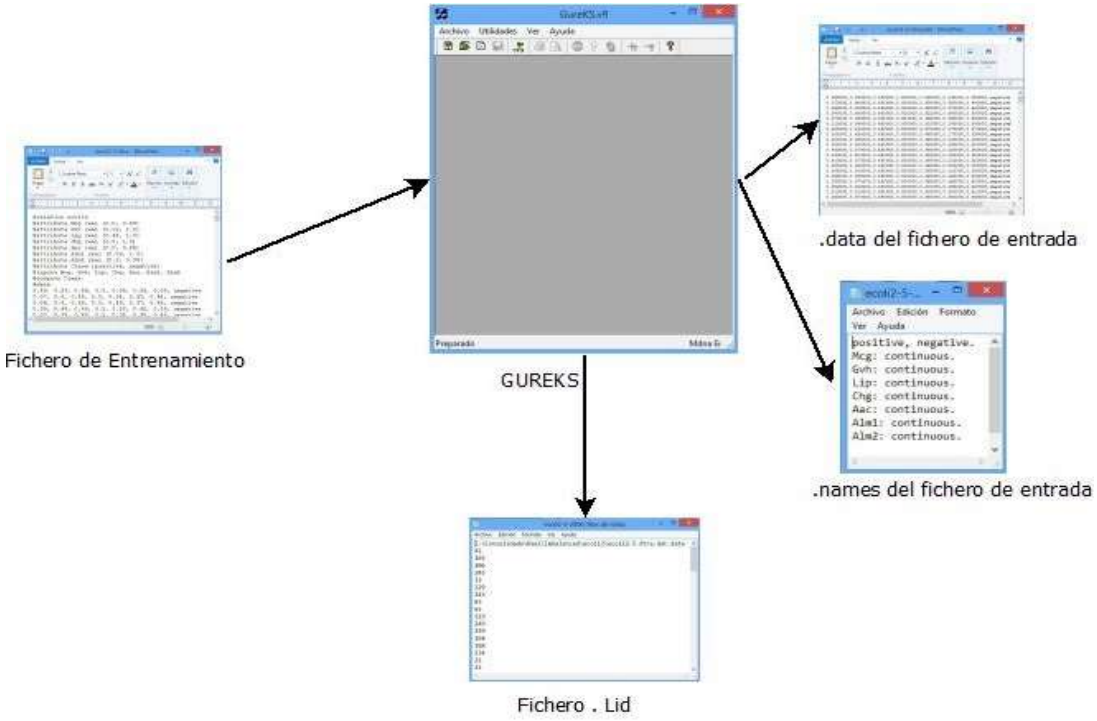


Ilustración 39 Proceso de creación de submuestras de entrenamiento en el GureKS

En la ilustración superior se detalla el proceso de creación de submuestras, se le pasa como entrada una muestra de entrenamiento en formato KEEL, la aplicación genera como salida tres ficheros distintos:

1. Fichero *.lid* o fichero con la lista de identificadores de casos: Es la manera que emplea GureKS para agrupar los casos de las muestras de entrenamiento en submuestras. Estos ficheros son archivos de texto con extensión *.lid* en el que la primera línea siempre debe ser la ruta completa en donde se encuentra la muestra de entrenamiento a partir de la cual se ha generado la submuestra. La segunda línea especifica el número de casos que componen la submuestra y el resto de líneas del fichero es la posición que toma la instancia de la base de datos seleccionada en la muestra de entrenamiento. De esta manera, se ahorra espacio en vez de generar submuestras con los casos concretos.

Ilustración 40 Fichero .lid para la base de datos abalone9-18

A la izquierda se encuentra un fichero *.lid*, creado a partir de una muestra de la base de datos abalone9-18, en la que la submuestra la componen 66

casos o instancias de la muestra entrenamiento.

2. Fichero *.data:* GureKS crea a partir de la muestra en formato KEEL, un fichero con los datos contenidos en la muestra de entrenamiento pero siguiendo el formato de datos de la UCI expuesto en la primera parte de este capítulo. Traduce la muestra de formato KEEL a formato UCI.

3. Fichero *.names:* Como sucede con el fichero *.data*, crea a partir de la muestra de entrenamiento de entrada la cabecera de la muestra siguiendo el formato de datos UCI, con lo que obtendremos a la salida un fichero *.names* indicando las clases que componen la muestra, los atributos y tipo de estos.

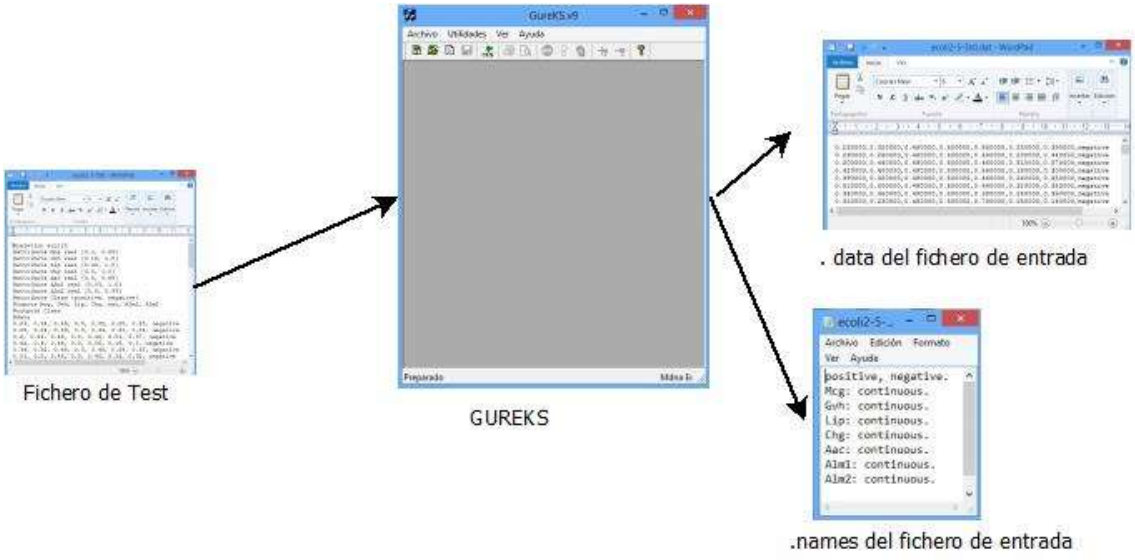


Ilustración 41 Proceso de creación para las muestras de test en GureKS

En esta última ilustración que encontramos en esta sección se detalla el proceso que se sigue con las muestras de test.

En este caso, a la salida del GureKS sólo obtenemos un fichero, pero esto no es del todo cierto, ya que realmente el proceso empleado es similar al de la creación de submuestras de entrenamiento, con la diferencia de que al finalizar el proceso, se eliminan de forma automática (mediante comandos MS-DOS) las submuestras generadas (los ficheros *.lid*) y el fichero *.names* asociado.

Como ejemplo para la base de datos ecoli al final del script se añaden estos dos comandos con este cometido:

Comando_DOS: "del ecoli-0_vs_1-5-1000.lid"

Comando_DOS: "del ecoli-0_vs_1-5-1tst.dat.names"

La razón o motivo para ello es que, el objetivo que se persigue con las muestras de test es solo obtener el fichero *.data* asociada al *.dat* leído y que se empleará en la fase de testeo del árbol en Haritza.

5.8.2 Haritza

Una vez generadas las submuestras a partir de las muestras de entrenamiento y ya con los bases de datos en formato UCI se dispone de todo lo necesario para la creación, poda y testeo de los árboles de clasificación consolidados.

Introducción y fundamentos

Esta tarea es trabajo de la herramienta Haritza. Podríamos calificarla como la encargada de construir los árboles de clasificación, en estos momentos la herramienta está dotada de capacidad para generar árboles de clasificación de tipo:

- CHAID.
- CTCHAID.
- C4.5.
- Los propios de este estudio, el CTC o C4.5 consolidado, CTCC4.5.
- Conjuntos de reglas PART.

Estos árboles pueden ser construidos a partir de ficheros de formato de tipo *.for* o *.names* y con ficheros de datos *.data* o *.lid.*

Los cometidos de esta herramienta podrían enumerarse en 5 fases:

- Carga de las submuestras con las que se crearán los árboles.
- Generación del árbol de clasificación a partir de dichas submuestras.
- Posible poda del árbol con los datos de la muestra de entrenamiento.
- Testeo del árbol con las muestras de test.
- Generación del fichero de resultados *.app* (probabilidad a posteriori).

Como sucede en el caso del GureKS, Haritza también posee una interfaz sencilla, intuitiva y liviana, siendo similares si bien ofrece distintas utilidades.

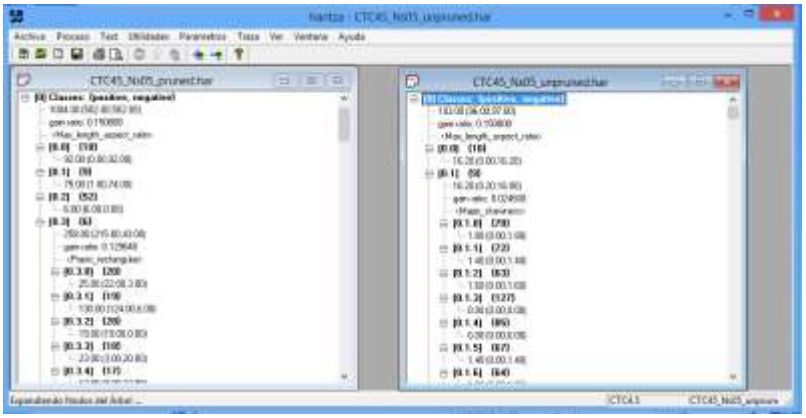


Ilustración 42 Apariencia de la aplicación Haritza

Funcionamiento de la aplicación

A primera vista se aprecia el menú que, como ocurre con el GureKS, es de vital importancia conocerlo si se desea realizar la experimentación de manera manual. También se disponen de accesos directos a las opciones que ofrece el menú. Como elemento distintivo entre las interfaces de las dos herramientas empleadas destacan dos nuevos botones.

Mediante los botones o accesos directos que se encuentran a la izquierda, se expanden o contraen los árboles de clasificación.

Ilustración 43 Botones para expandir nodo y contraer nodo

Mediante el menú archivo, como se desprende de la ilustración 44, se puede:

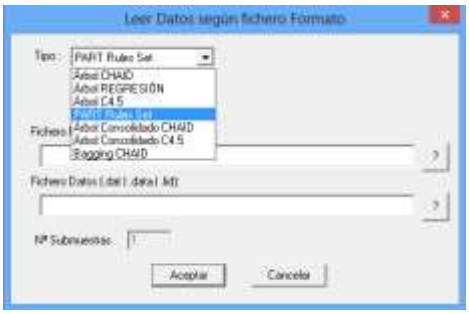


Ilustración 44 Ventana del menú archivo, cargar muestra

- Cargar la muestra. Se debe seleccionar el tipo de árbol que se desea crear, el fichero formato de la muestra, de tipo *.for* o *.names* y el fichero que contiene la muestra que puede ser de tipo *lid* y *data*.

- Abrir, para cargar en la herramienta árboles ya generados, son ficheros con extensión *.har*

- Cargar script de comandos*:* Estos scripts son una combinación de comandos que permiten realizar las operaciones de manera autónoma.

- Configurar impresora

El menú Ver y el Help son iguales que GureKS.

El menú utilidades ofrece la posibilidad de desalojar muestras de memoria. Esto es interesante ya que permite acelerar la experimentación guardando en memoria las muestras que va leyendo y evitando tener que releerlas de nuevo. En bases de datos muy voluminosas el tiempo que se ahorra es destacable.

Automatización del proceso

Procediendo de manera análoga al GureKS, a la hora de llevar a cabo experimentación se ha decidido realizarla de la forma más automatizada posible por los mismos motivos.

Los archivos *.bat* necesarios se han creado de una manera similar a la empleada hasta ahora. En la ilustración 45 se ve el fichero *.bat* resultante.

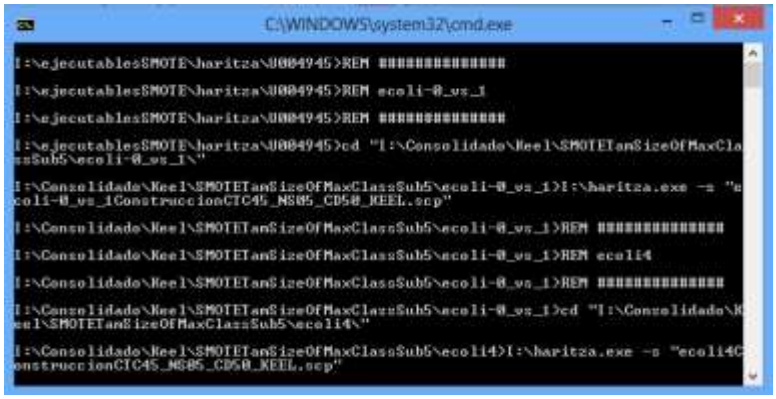


Ilustración 45 Fichero ejecutable en MS-DOS para lanzar experimentos en Haritza

Una vez situados en la carpeta de la base de datos, mediante el comando MS-DOS cd, con: I:\haritza.exe -s "ecoli-0_vs_1ConstruccionCTC45_NS05_CD50_KEEL.scp" se lanza el experimento, y como sucede en la generación de submuestras se le pasa como argumento el script que contiene los comandos para crear y operar con los árboles.

Ilustración 46 Script para creación del árbol CTC, poda y test

En la ilustración 46 se ven los comandos empleados para la creación de un árbol de clasificación consolidado de la base de datos ecoli3. El experimento se detalla entre los token EXPERIMENTO_COM y EXPERIMENTO_FIN. La carga de las submuestras se realiza entre los TOKEN CARGAR_MUESTRA_COM y CARGAR_MUESTRA_FIN, seguidamente se encuentra el token CONSTRUIR_ARBOL_CONSOLIDADO_C45_COM, se indica mediante él el tipo de árbol que se construirá. Para podar el árbol hay que indicar el nombre del fichero de datos y el nombre que se le dará al árbol podado (*.har)*, ambos parámetros se encuentran entre los token PODAR_ARBOL_COM y PODAR_ARBOL_FIN. De manera similar se procede indicar el fichero de testeo y el de creación del resultado.

Proceso llevado a cabo

El proceso que se emplea para realizar la parte de la experimentación correspondiente a Haritza se detalla de manera visual en la siguiente ilustración.

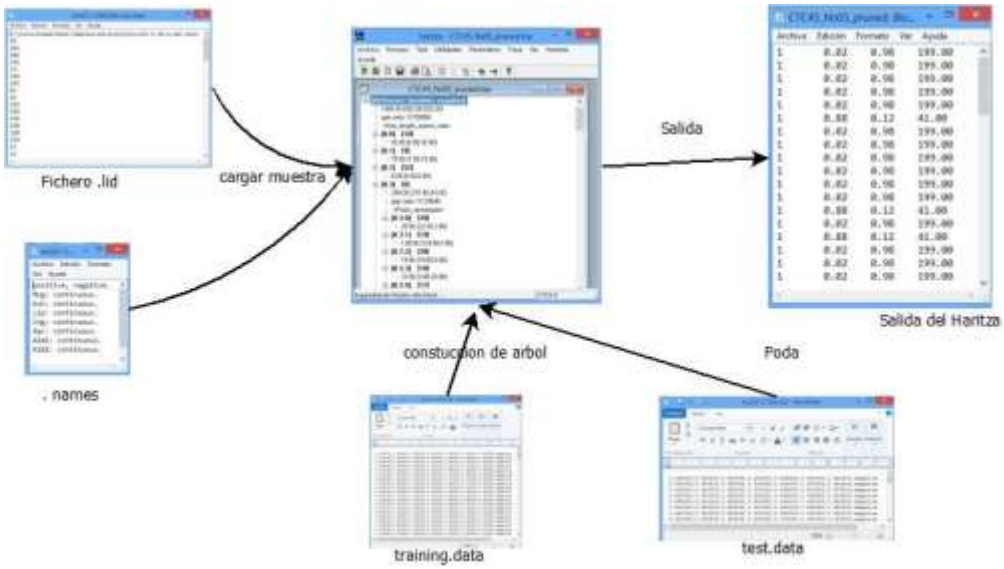


Ilustración 47 Proceso completo de la creación, poda y test de árboles en Haritza

El proceso de creación del árbol consolidado comienza indicando cuál es el fichero *.lid* en el que se encuentran los identificadores de casos y el fichero que define el formato, de esta manera Haritza carga las submuestas que se emplearán en la creación del árbol.

Una vez creado el árbol de clasificación emplea la muestra de entrenamiento para realizar la poda del mismo y guarda el resultado en formato de datos *.har*, véase ilustración 48.

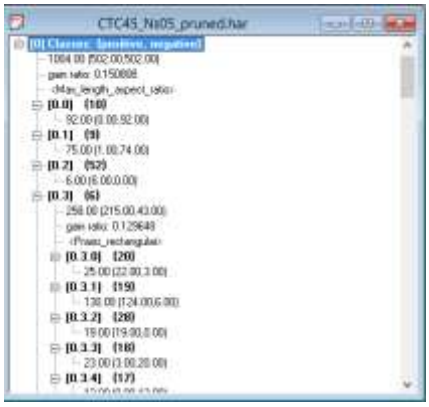


Ilustración 48 Fichero .har expandido

Como resultado de todo este proceso Haritza genera un fichero de salida con extensión *.app* del que se encuentra un ejemplo en la ilustración 49. Muestra los resultados obtenidos en la clasificación de las instancias pertenecientes a la muestra de test correspondiente.

Los datos se encuentran en 4 columnas para problemas biclase. En caso de encontrarnos con problemas multiclase, se dispondrían de (2 + número de clases) columnas.

Cada línea del fichero representa a uno de los casos de la muestra de test, por lo tanto tendremos tantas líneas como casos compongan el fichero de test.

- La primera columna es la clase verdadera a la que pertenece el caso.

- Las dos siguientes, columna 2 y 3 hacen referencia a las dos clases posibles, y muestran la probabilidad a posteriori de clasificar el caso como perteneciente a la primera clase, para la segunda columna o a la segunda clase en el caso de la tercera columna. Se define la probabilidad a posteriori como la probabilidad que asigna el clasificador al caso de pertenecer a esa clase en la que está.

- La última columna es el peso de la hoja del árbol en la que ha sido encuadrado el caso, puede aportar información extra a la clasificación.

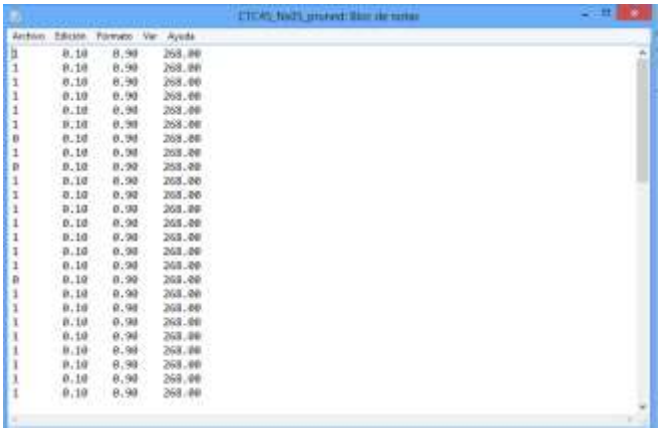


Ilustración 49 Fichero .app resultante del testeo del árbol

5.8.3 Kurbak

Al finalizar el proceso en Haritza para cada una de las muestras de test, se dispone de un fichero *.app* con los resultados de la clasificación de los casos que componen el conjunto de test, es necesario pues tratar estos resultados y obtener las principales medidas de bondad de un clasificador.

Introducción y fundamentos

Kurbak es la tercera herramienta que se empleará en esta experimentación. Está desarrollada en Java por el grupo ALDAPA. A diferencia de las dos aplicaciones vistas hasta ahora ésta carece de interfaz gráfico y su ejecución se realiza únicamente mediante MS-DOS.

A simple vista puede parecer una aplicación poco atractiva, pero es realmente útil pues ofrece distintos modos de funcionamiento reuniendo las utilidades que se pretende que ofrezca un programa de este tipo.

El objetivo principal es el de calcular las medidas de bondad resultantes de la clasificación en un árbol. El funcionamiento de esta aplicación es sencillo.

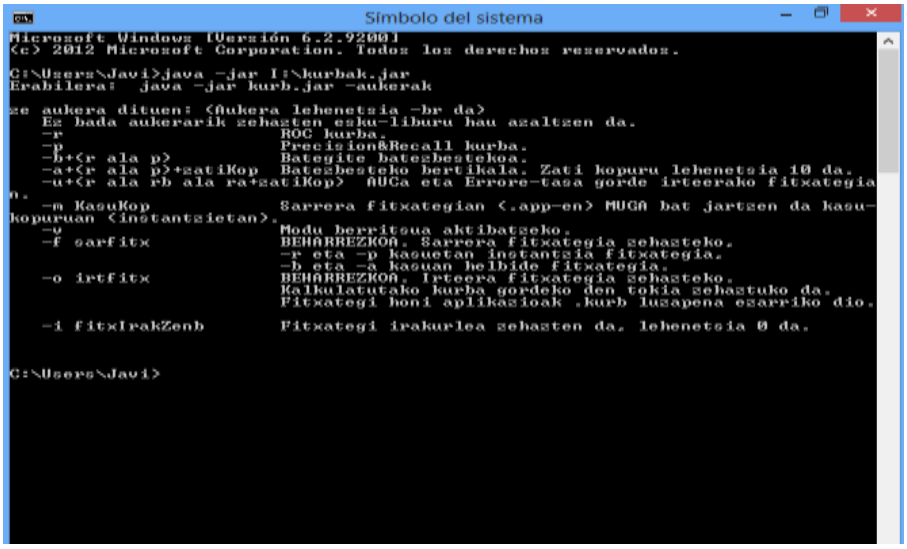


Ilustración 50 Apariencia de la aplicación Kurbak

Funcionamiento

En la ilustración anterior se detallan los modos de funcionamiento que soporta. En función del argumento que se pase en la llamada de la aplicación la salida obtenida es distinta:

- -r para sacar la curva roc.
- -p para obtener la curva Precison&Recall.

- -b y (r o p) para obtener la media de un conjunto de *.app's*
- -a y (r o rb o ra y numTrozos) para obtener la media pero mediante la vertical.
- -n para indicar el número de casos máximos a tratar del fichero *.app*.
- -u + (r o rb p ra+numTrozos) para obtener la curva AUC y la tasa de error el fichero de salida.

En la propia llamada se indica el fichero de entrada y el fichero de salida. Indicar estos dos argumentos es obligatorio.

- –f ficheroEntrada
- –o ficheroSalida
- -i ficheroLectorNum

El fichero de entrada que se indica como argumento tras el parámetro –f debe ser un fichero *.txt* como el que se encuentra en la ilustración 51. En él se debe indicar la ruta completa en la que se encuentra el archivo *.app* del que queremos calcular las distintas medidas de bondad

Este fichero se genera mediante VisualBasicScripting. Además del fichero de entrada, en la zona inferior de la ilustración se encuentra el fichero *.app* al que hace referencia, los ficheros *.lid* que forman las submuestras y los dos archivos generados en la fase de ejecución de Haritza que son el árbol consolidado sin podar y el podado, ambos poseen extensión *.har.*

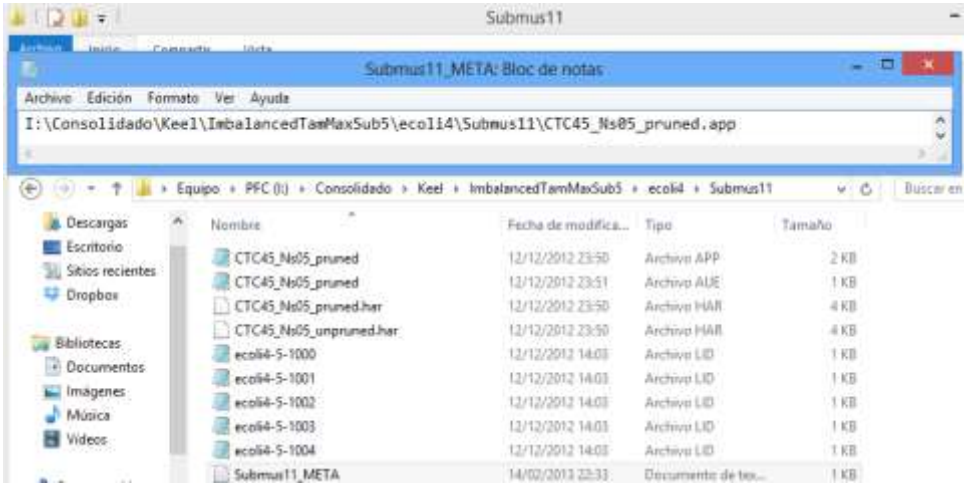


Ilustración 51 Ejemplo de fichero meta con la ruta del app

Kurbak genera un fichero de salida como resultado del proceso, por tanto, es necesario indicar el nombre después del parámetro –o.

El formato del fichero de salida tiene extensión *.aue*, y muestra los resultados de la clasificación obtenidos a partir de los ficheros *.app*. Está organizado en 6 columnas, una por cada una de las medidas de bondad expuestas en el capítulo 2. Se encuentra un ejemplo de este fichero en la ilustración inferior.

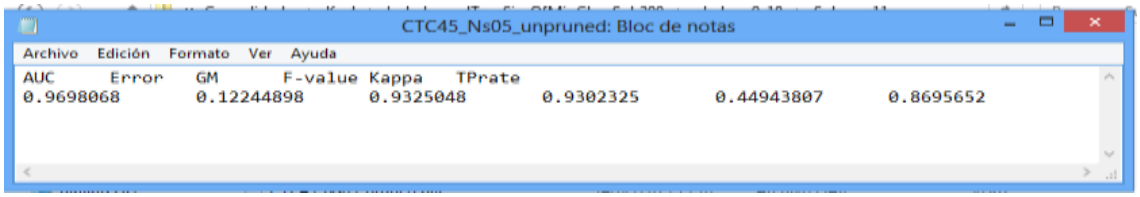


Ilustración 52 Fichero .aue de salida del Kurbak

Automatización del proceso

Para automatizar el proceso es necesario crear dos tipos de ficheros distintos. Se genera en cada uno de los conjuntos de submuestras un fichero de texto en el que se indica la ruta en la que se encuentra el fichero *.app*. El segundo tipo de ficheros son los ya conocidos *.bat*, que están formados por los comandos MS-DOS que posibilitan ejecutar la aplicación mediante línea de comandos.

El fichero ejecutable estaría formado por sentencias de la forma:

```
REM ##############

REM glass2

REM ##############

REM A priori  C45 AUC Pruned

java -jar I:\kurbak.jar -rbu -f I:\Consolidado\Keel\ImbalancedTamMaxSub5\glass2\Submus11\Submus11_METASINPODA.txt -o
I:\Consolidado\Keel\ImbalancedTamMaxSub5\glass2\Submus11\CTC45_Ns05_unpruned

java -jar I:\kurbak.jar -rbu -f I:\Consolidado\Keel\ImbalancedTamMaxSub5\glass2\Submus12\Submus12_METASINPODA.txt -o
I:\Consolidado\Keel\ImbalancedTamMaxSub5\glass2\Submus12\CTC45_Ns05_unpruned
```

El proceso completo se describe en la ilustración 53. La aplicación Kurbak recibe como entrada el fichero meta en el que se especifica la localización del fichero *.app* y produce como salida el fichero *.aue* con los resultados.

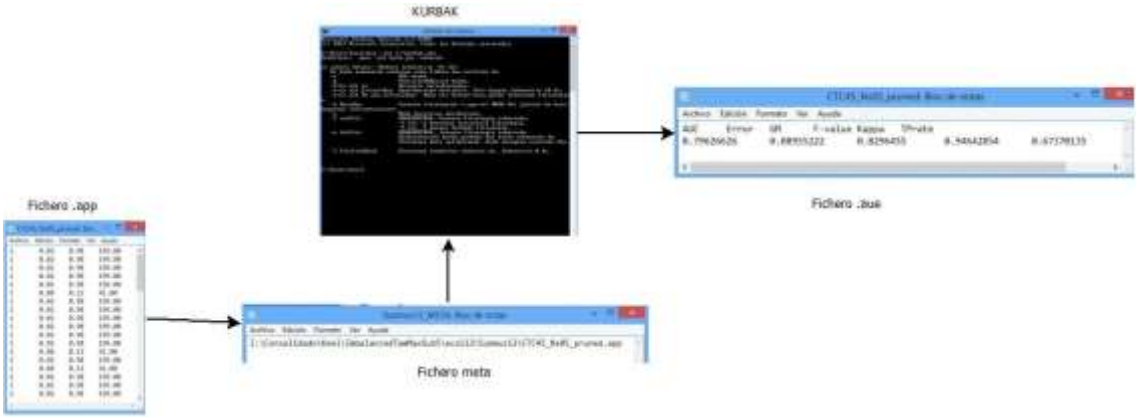


Ilustración 53 Proceso completo del Kurbak

5.8.4 Visual Basic Scripting y Excel

La última herramienta empleada en esta experimentación es la aplicación por excelencia en cuanto a las hojas de cálculo se refiere: Excel y lenguaje Visual Basic Scripting.

El lenguaje Visual Basic Scripting o VBScript deriva del lenguaje Visual Basic creado por Microsoft, al que se podría calificar como el primer lenguaje que permite desarrollar aplicaciones en la plataforma Windows de una manera más sencilla e intuitiva, simplificando el proceso mediante un entorno gráfico.

Visual Basic Script es un lenguaje de scripts, como su nombre hace referencia. Nos da la posibilidad de automatizar procesos y en combinación con el Excel lo convierte en una herramienta poderosa. Cabe destacar que se basa en el modelo de programación orientado a objetos.

Preparación de la herramienta.

En las últimas versiones de Office es necesario activar la casilla de programador, que por defecto viene oculta. Para ello clicamos en:

1. Archivo, opciones, personalizar cuenta de opciones y finalmente, en el cuadro del lado derecho tenemos un listado, solo hay que dar clic en la opción de desarrollador.

En la ilustración 54 se aprecia este menú de opciones, con la casilla programador clicada.

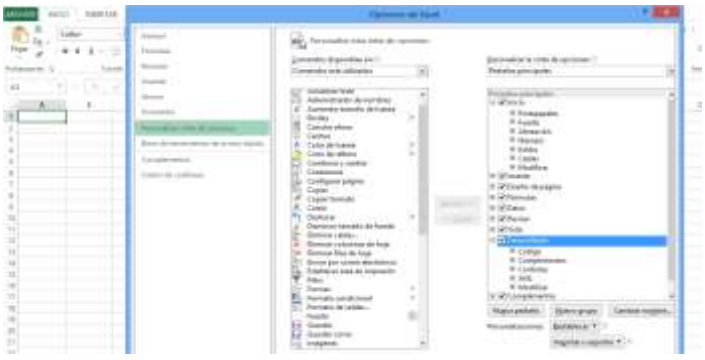


Ilustración 54 Menú de configuración de las opciones en Excel 2013

En estos momentos ya está preparado Excel para poder programar macros desde cero. Para acceder a la zona de programación, dentro de la pestaña desarrollador se encuentra un acceso directo como el de la ilustración 55.

Ilustración 55 Accceso rápido al entrono de trabajo VBScripting

Proceso de automatización

La metodología o procedimiento empleado para la automatización de las tareas mediante la herramienta que se acaba de exponer es similar para las 4 variantes en las que se emplea (GureKS, Haritza, Kurbak y las medias).

En la hoja Excel, véase la ilustración 56, se detallan las bases de datos que emplearemos en el estudio, en este caso se encuentran en la primera columna de la hoja a partir de la línea 15. Mediante un bucle se leen todas las bases de datos especificadas, esto es común para todas las herramientas. El código que posibilita esta tarea es el siguiente:

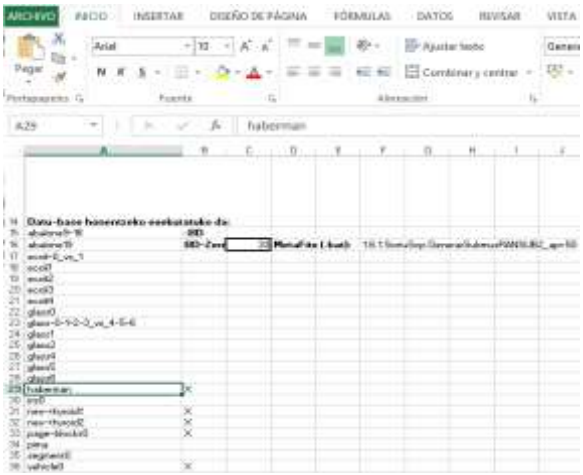


Ilustración 56 Fichero Excel con las bases de datos del estudio

```
    For i = 0 To BDKop - 1
        Datubasea = Cells(15 + i, 1).Value ' las BD
        unitatea = Cells(15 + i, 24).Value ' ordenagailua
Select Case unitatea
        Case "Yo"
        unitatea2 = "I:"
        Case "U004945"
        unitatea2 = "I:" '"G:"
        Case "U004946"
        unitatea2 = "I:" '"H:"
        Case "U004947"
        unitatea2 = "I:" ' "J:"
        Case "U004948"
        unitatea2 = "I:" '"K:"
```

Además se distribuyen las bases de datos entre las máquinas que participarán en la experimentación, 4 en concreto.

A continuación se encuentra la distribución concreta de las bases de datos entre las máquinas del grupo ALDAPA.

Para el caso de GureKS y Haritza se generan dos tipos de ficheros distintos:

- Los scripts que emplearemos para la creación de submuestras y la generación de los árboles.

  Para el GureKS del código Visual Basic Scripting empleado, destaca el encargado de crear el script, el resto del código es trivial y no posee ningún interés algorítmico:

  Para crear el script que genera las submuestras a partir de las muestras de training y los ficheros *.dat* y *.names*:

```
                    Print #1, "SCRIPT_COM"
                Print #1, "DIRECTORIO_TRABAJO: " + "I:" + PathBaseDB
                Print #1, "EXPERIMENTO_COM"
                Print #1, "Crear_Directorio:    Submus" + CStr(r + 1) + CStr(f + 1) + "\"
                Print #1, "GENERADOR_SUBMUESTRAS_COM"
                Print #1, "      FICHERO_DATOS: " + DBSin + "-5-" + CStr(f + 1) + "tra.dat"
                Print #1, "      FICHERO_FORMATO:    " + DBSin + "-5-" + CStr(f + 1) + "tra.dat"
                Print #1, "      Tipo_generador:            Apriori_Cambiado"
                Print #1, "      GENERAR_LISTA_IDENTS_CASO"
                Print #1, "      Numero_muestras:      05"
                Print #1, "      Distribuir_al_Azar:       SI"
                Print #1, "      Porcentaje_clase0: " + stApriori(c)
                Print #1, ""
                Print #1, "      Modo_nombre_fichero:      automatico"
                Print #1, "      nombre_fichero:       Submus" + CStr(r + 1) + CStr(f + 1) + "\" + DBSin + "-5-" +
CStr(f + 1) + "*.lid"
                Print #1, "      num_ficheros:     05"
```

```
                Print #1, "GENERADOR_SUBMUESTRAS_FIN"
                Print #1, "EXPERIMENTO_FIN"
                Print #1, ""
            Next c
        Next f
    Next r
    Print #1, "SCRIPT_FIN"
```

Para crear el script que genera el fichero *.dat* a partir de las muestras de test se emplea el siguiente código, destaca que al finalizar se borran los ficheros *.lid* creado y el *.names*, ya que no son necesarios en absoluto en las muestras de test, sólo se necesita el *.data*:

```
                Print #1, "EXPERIMENTO_COM"
                Print #1, "GENERADOR_SUBMUESTRAS_COM"
                Print #1, "      FICHERO_DATOS: " + DBSin + "-5-" + CStr(f + 1) + "tst.dat"
                Print #1, "      FICHERO_FORMATO:    " + DBSin + "-5-" + CStr(f + 1) + "tst.dat"
                Print #1, "      Tipo_generador:              Apriori_Cambiado"
                Print #1, "      GENERAR_LISTA_IDENTS_CASO"
                Print #1, "      Numero_muestras:      01"
                Print #1, "      Distribuir_al_Azar:       SI"
                Print #1, "      Porcentaje_clase0: " + stApriori(c)
                Print #1, "      Modo_nombre_fichero:      automatico"
                Print #1, "      nombre_fichero:       " + DBSin + "-5-" + CStr(f + 1) + "*.lid"
                Print #1, "      num_ficheros:    01"
                Print #1, "GENERADOR_SUBMUESTRAS_FIN"
                Print #1, "Comando_DOS: " + Chr(34) + "del " + DBSin + "-5-" + CStr(f + 1) + "000.lid" +
Chr(34)
                Print #1, "Comando_DOS: " + Chr(34) + "del " + DBSin + "-5-" + CStr(f + 1) + "tst.dat.names" +
Chr(34)
                Print #1, "EXPERIMENTO_FIN"
                Print #1, ""
            Next c
             Next f
    Print #1, "SCRIPT_FIN"
```

Para Haritza del código Visual Basic Scripting empleado destaca el encargado de crear el script, mediante el cual se cargarán muestras, se creará el árbol, se podará, testeará y se obtendrán los resultados a posteriori:

```
            Print #1, "DIRECTORIO_TRABAJO: " + "I:" + PathBaseDB
            Print #1, "EXPERIMENTO_COM"
            Print #1, "CARGAR_MUESTRA_COM"
            Print #1, "Tipo_Arbol:       CTC4.5"
            Print #1, "      Num_Ficheros:         5"
            Print #1, "      Fichero_Datos:        " + "Submus" + CStr(r + 1) + CStr(f + 1) + "\" + DBSin + "-5-" +
CStr(f + 1) + "*.lid"
            Print #1, "      Fichero_Formato:      " + DBSin + "-5-" + CStr(f + 1) + "tra.dat.names"
            Print #1, "CARGAR_MUESTRA_FIN"
            Print #1, "CONSTRUIR_ARBOL_CONSOLIDADO_C45_COM"
            Print #1, "      GUARDAR " + "Submus" + CStr(r + 1) + CStr(f + 1) + "\" +
"CTC45_Ns05_unpruned.har"
            Print #1, "CONSTRUIR_ARBOL_CONSOLIDADO_C45_FIN"
            Print #1, "PODAR_ARBOL_COM"
            Print #1, "      Nombre_Fichero:       " + "Submus" + CStr(r + 1) + CStr(f + 1) + "\" +
"CTC45_Ns05_pruned.har"
            Print #1, "      Fichero_Datos:        " + DBSin + "-5-" + CStr(f + 1) + "tra.dat.data"
```

```
            Print #1, "PODAR_ARBOL_FIN"
            Print #1, "TESTEAR_ARBOL_COM"
            Print #1, "      Modo_nombre_Fichero:      unico"
            Print #1, "      nombre_fichero:       " + DBSin + "-5-" + CStr(f + 1) + "tst.dat.data"
            Print #1, "      num_ficheros:         1"
            Print #1, "TESTEAR_ARBOL_FIN"
            Print #1, "RESULTADOS_COM"
            Print #1, "      Nombre_fichero:       " + "Submus" + CStr(r + 1) + CStr(f + 1) + "\" +
"CTC45_Ns05_pruned.app"
            Print #1, "      Tipo_resultado:       Probabilidad_aposteriori"
            Print #1, "RESULTADOS_FIN"
            Print #1, "EXPERIMENTO_FIN"
            Print #1, ""
        Next r
        Print #1, "DESALOJA_MEMORIA_MUESTRAS"
    Next f
    Print #1, "SCRIPT_FIN"
```

- Los ficheros *.bat* ejecutables que contienen las instrucciones para la ejecución de las plataformas en la línea de comandos.
  Para generar el fichero *.bat* que se ejecutará en la máquina U004945.

```
    FitxMeta_Izena = Cells(16, 6).Value 'MetaFitx (.bat)
    FitxMeta_Izena = "I:\ejecutablesTamMaxSub5\gureks\U004945\" + "ImbalancedTamMaxSub5" + FitxMeta_Izena +
".bat"
```

Las instrucciones que formarán el fichero se generan de la siguiente manera, para cada una de las bases de datos del estudio:

```
Case "U004945"
            Print #2, "REM ###############"
        Print #2, "REM " + Databasea
        Print #2, "REM ###############"
        Print #2, ""
        Print #2, "cd " + Chr(34) + "I:" + GehituBestePorzentaia(PathBaseDB) + Chr(34)
        FitxMeta_Izena = Cells(16, 6).Value
        FitxMeta_Izena = FitxMeta_Izena + ".scp"
        Print #2, "I:\v9.exe -s " + Chr(34) + Databasea + FitxMeta_Izena + Chr(34)
        FitxMeta_Izena = unitatea2 + PathBaseDB + Databasea + FitxMeta_Izena
```

En el caso del Kurbak también generaremos dos ficheros pero, en este caso no es un script sino un meta fichero de texto, en el que se indica la ruta en la que se encuentra el archivo *.app* a partir del cual se calcularán las medidas de bondad.

```
Sub MetafitxategiakAPPSortu(DB As String)
Dim PathModI As String, FitxApp As String, PathModI2 As String
Dim r As Integer, f As Integer, ap As Integer, i As Integer, m As Integer
        PathBaseDBI = "I:\Consolidado\Keel\ImbalancedTamMaxSub5\" + DB + "\"
    For r = 0 To 4
        For f = 0 To 4
                    ' C45 submu
                    PathModI = PathBaseDBI + "Submus" + CStr(r + 1) + CStr(f + 1)
                    'PathModI = PathBaseDBI
                    MetaFitx = PathModI + "\Submus" + CStr(r + 1) + CStr(f + 1) + "_META.txt"

                    Open MetaFitx For Output As #6
                        FitxApp = PathModI + "\CTC45_Ns05_pruned.app"
                    Print #6, FitxApp
```

```
                    Close #6
            Next f
        Next r
End Sub
```

Como sucede con GureKS y Haritza aquí también se genera un fichero *.bat*, que es el encargado de ejecutar desde la línea de comando el programa Kurbak con los parámetros correspondientes. Para generar las instrucciones que lo forman se emplea el siguiente código, que se ejecuta por cada una de las bases de datos.

```
Print #2, "REM ##############"
Print #2, "REM " + Datubasea
Print #2, "REM ##############"
Print #2, "REM A priori  C45 AUC Pruned"
    For r = 0 To 4
      For f = 0 To 4
          FitxDB = PathBaseDBI + "Submus" + CStr(r + 1) + CStr(f + 1)
              Print #2, "java -jar I:\kurbak.jar -rbu -f " + FitxDB + "\Submus" + CStr(r + 1) + CStr(f + 1) +
"_META.txt" + " -o " + FitxDB + "\CTC45_Ns05_pruned"
            Next f
        Next r
```

Por último se emplea código Visual Basic Scripting para generar automáticamente las medias y tratar los datos. En cada una de las bases de datos del estudio se generan 3 archivos Excel, en los que se agrupan de dos en dos las 6 medidas de bondad para cada árbol generado y la media de estos 25 valores.

```
Sub RellenarResumen(DB As String, i As Integer)
Dim pos As Integer
pos = i + 15
Workbooks.Open Filename:="I:\Consolidado\Keel\ImbalancedTamMaxSub5\" + DB + "\AUC_Error_pruned_5run.xls"
Application.Workbooks(1).Worksheets(2).Range("B" + CStr(pos)).Value = ActiveSheet.Range("E4")
Application.Workbooks(1).Worksheets(2).Range("C" + CStr(pos)).Value = ActiveSheet.Range("F4")
Application.Workbooks(1).Worksheets(2).Range("E" + CStr(pos)).Value = ActiveSheet.Range("E22")
Application.Workbooks(1).Worksheets(2).Range("F" + CStr(pos)).Value = ActiveSheet.Range("F22")
ActiveWorkbook.Close
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Workbooks.Open Filename:="I:\Consolidado\Keel\ImbalancedTamMaxSub5\" + DB + "\GM_Fvalue_pruned_5run.xls"
Application.Workbooks(1).Worksheets(2).Range("H" + CStr(pos)).Value = ActiveSheet.Range("E4")
Application.Workbooks(1).Worksheets(2).Range("I" + CStr(pos)).Value = ActiveSheet.Range("F4")
Application.Workbooks(1).Worksheets(2).Range("K" + CStr(pos)).Value = ActiveSheet.Range("E22")
Application.Workbooks(1).Worksheets(2).Range("L" + CStr(pos)).Value = ActiveSheet.Range("F22")
ActiveWorkbook.Close
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Workbooks.Open Filename:="I:\Consolidado\Keel\ImbalancedTamMaxSub5\" + DB + "\Kap_TPR_pruned_5run.xls"
Application.Workbooks(1).Worksheets(2).Range("N" + CStr(pos)).Value = ActiveSheet.Range("E4")
Application.Workbooks(1).Worksheets(2).Range("O" + CStr(pos)).Value = ActiveSheet.Range("F4")
Application.Workbooks(1).Worksheets(2).Range("Q" + CStr(pos)).Value = ActiveSheet.Range("E22")
Application.Workbooks(1).Worksheets(2).Range("R" + CStr(pos)).Value = ActiveSheet.Range("F22")
ActiveWorkbook.Close
End Sub
```

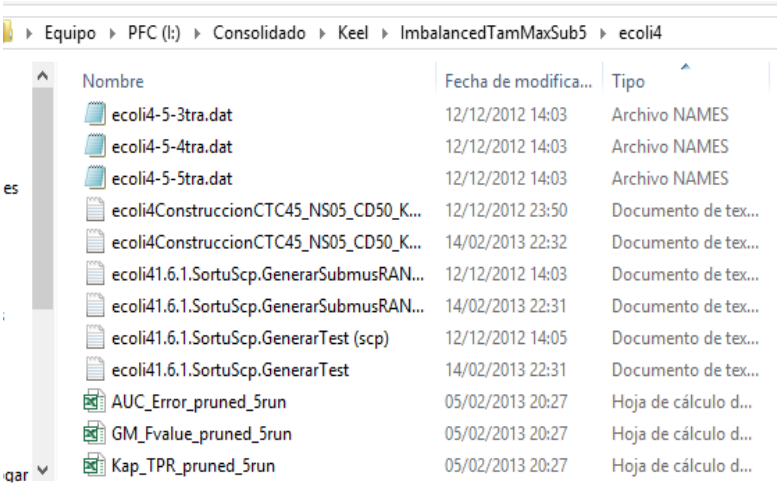


Ilustración 57 Ficheros Excel generados con las medidas de bondad

Se muestra a continuación el código para generar el archivo con los resultados del AUC y el error, de la misma manera se procede para crear el archivo que reúne los resultados del GM-FValue y TPR-Kappa:

```
        Sub RellenarResumenExcel5RunAUCError(DB As String, stUn_Pruned As String)
Dim Helb_ap_etik As String 'Helburuko aprioriaren etiketa <= Klase minoritarioaren posizioaren araberakoa

Application.DisplayAlerts = False
    FitxTmp = "5.0.0.Tmp(Plantilla).xls"
    Workbooks.Open Filename:="I:\ejecutablesTamMaxSub5\medias aue\" + FitxTmp
    FitxIzenaPl = "I:\ejecutablesTamMaxSub5\medias aue\5.0.2Irizpide_unpruned_5run(Plantilla).xls
    ' Excel Resultado 5 run
    Workbooks.Open Filename:=FitxIzenaPl
    FitxIzena1 = "AUC_Error_" + stUn_Pruned + "_5run.xls"
    ActiveWorkbook.SaveAs Filename:=PathBaseDBI + FitxIzena1
    Sheets("Laburpena").Select
    Cells(1, 7).Value = DB
    Cells(3, 4).Value = "%C=" + CStr(stApriori(k))
    Cells(21, 4).Value = "%C=" + CStr(stApriori(k))
    Cells(2, 1) = "AUC"
    Cells(19, 1) = "Error"
```

A partir de estos ficheros y también mediante código, se genera un único archivo Excel que contiene la media de las 25 ejecuciones para todas las medidas de bondad y todas las bases de datos.

```
        For met = 0 To 4
        For r = 0 To 3
            Pathres = "Submus" + CStr(met + 1) + CStr(r + 1) + "\"
                TestuFitx = "CTC45_Ns05_pruned" + ".aue"
                TestuFitxIzenOsoa = PathBaseDBI + Pathres + TestuFitx
                Workbooks.OpenText Filename:=TestuFitxIzenOsoa, Tab:=True, DecimalSeparator:=".",
ThousandsSeparator:=","
                Range("A1:F2").Select
                Selection.Copy
                Windows(FitxTmp).Activate
                Cells(1, 1).Select
                Selection.PasteSpecial Paste:=xlValues
                ' AUC
                Windows(FitxTmp).Activate
                Range("A21").Select
                Selection.Copy
                Windows(FitxIzena1).Activate
```

```
Sheets("kb" + CStr(1)).Select
Cells(met + 2, r + 3).Select
Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, SkipBlanks:= _
    False, Transpose:=False
    ' Err
Windows(FitxTmp).Activate
Range("B21").Select
Selection.Copy
Windows(FitxIzena1).Activate
Sheets("kb" + CStr(1)).Select
Cells(met + 13, r + 3).Select
Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, SkipBlanks:= _
    False, Transpose:=False
```

El último paso es tratar los datos. Se programa mediante Visual Basic la creación de los test de significancia expuestos en el capítulo 2.

También el código es el encargado de generar los rankings que se emplearán para la comparación directa, así como funciones capaces de resaltar datos concretos que puedan resultar interesantes.

Para finalizar esta sección, a modo resumen en la ilustración 58 se encuentran los distintos tipos de archivos generados mediante código Visual Basic Scripting correspondiente. (Scripts para GureKS y Haritza, *.bat* para GureKS, Haritza y Kurbak, ficheros meta de Kurbak, medias y tratamiento de datos

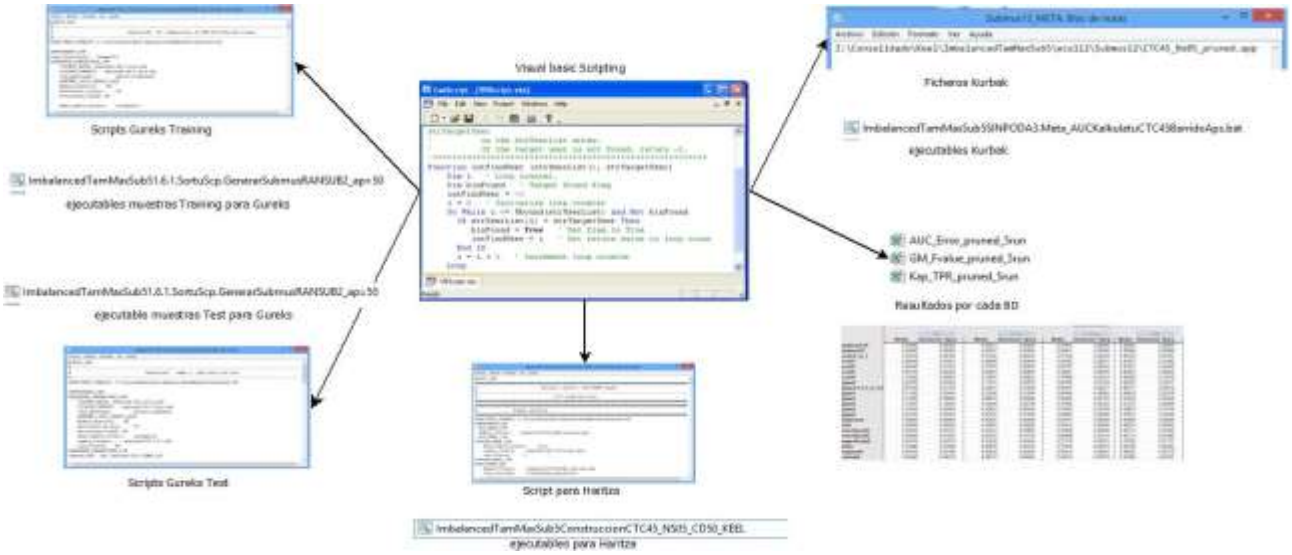


Ilustración 58 Proceso de creación de los scripts y ejecutables necesarios para la experimentación

5.9 Introducción a la experimentación realizada

Se han realizado 2 bloques de experimentos, los dos sobre el mismo conjunto bases de datos si bien en el último experimento, el conjunto de bases de datos ha sido preprocesado mediante técnicas SMOTE con el fin de balancear las clases. En el otro

bloque se realiza la experimentación con dos tamaños de submuestras distintos, submuestras al tamaño de la clase minoritaria y submuestras al tamaño máximo y una distribución al 50% en ambos casos.

La metodología experimental aplicada, es ampliamente conocida. Su principal característica es que resulta ser lo suficientemente rigurosa y extensa para no poder achacar los resultados obtenidos al azar.

En los siguientes dos capítulos de esta memoria se describen de manera detallada cada uno de los experimentos llevados a cabo, describiendo la metodología experimental utilizada, los distintos valores para los parámetros del algoritmo de construcción de árboles consolidados, y para finalizar, los resultados obtenidos.

Posteriormente se compararán los resultados obtenidos por el CTC con los publicados por Fernández et al en [FGLBH10] y se muestran las conclusiones que se desprenden de ellos.

Cabe destacar que entre los 22 algoritmos del estudio, presumiblemente el CTC debería obtener mejores o similares resultados que el C4.5, que podría calificarse como el competidor por excelencia del CTC ya que fue elegido como algoritmo base para la construcción del algoritmo de construcción de árboles consolidados. Pese a basarse en el C4.5 conviene destacar que existen varios aspectos en los que difieren ambos algoritmos.

Capítulo 6
Experimentación 1

Análisis del comportamiento del CTC en un contexto de [33 BDS] desbalanceadas y en comparación con 22 algoritmos de inducción de reglas [(16 evolutivos y 6 clásicos)]

En este capítulo se analiza el comportamiento del algoritmo de construcción de árboles consolidados, CTC, para las 33 bases de datos (imbalanced) descritas en el capítulo precedente, y una comparación con los resultados obtenidos por Fernández et al en [FGLBH10]

6.1 Introducción al experimento [1]

Para llevar a cabo esta experimentación, el análisis del comportamiento del CTC en un contexto de bases de datos desbalanceadas, se han empleado las 33 bases de datos en la modalidad *imbalance* descargadas del enlace http://sci2s.ugr.es/keel/index.php. Las bases de datos en cuestión son : abalone9-18, abalone19, ecoli-0_vs_1, ecoli1, ecoli2, ecoli3, ecoli4, glass0, glass-0-1-2-3_vs_4-5-6, glass1, glass2, glass4, glass5, glass6, haberman, iris0, new-thyroid1, new-thyroid2, page-blocks0, pima, segment0, vehicle0, vehicle1, vehicle2, vehicle3, vowel0, Wisconsin, yeast1, yeast-2_vs_8, yeast3, yeast4, yeast5, yeast6.

El número de instancias de los conjuntos de datos varían entre 150 y 5.472, el número de atributos entre 3 y 16, el porcentaje de la clase minoritaria entre 0,77 % y 33,51%. Esto supone 10 casos de la clase minoritaria en la base de datos más desbalanceada y 560 casos de la clase minoritaria en el conjunto de datos menos desbalanceado.

6.2 Metodología experimental

Respecto a la metodología empleada para la validación se han realizado 5 validaciones cruzadas de 5 subconjuntos o *folds*. Para cada subconjunto de la validación cruzada se ha aplicado la misma técnica de remuestreo generando un número de submuestras variable y con una distribución de clases al 50%. Esta distribución de clases empleada deriva de los trabajos del grupo de investigación que concluyen que con esta distribución se obtienen resultados cerca del óptimo.

El tamaño de las submuestras se ha establecido al número de elementos pertenecientes a la clase minoritaria del conjunto de entrenamiento, como realizaron Weiss y Provost en [WP03]. Esto hace que sea posible generar la distribución de cualquier clase para este tamaño sin replicar ningún elemento o caso.

Para un caso extremo como abalone19, el tamaño de las submuestras sería únicamente de 26 casos cuando el conjunto de entrenamiento tiene un tamaño de 3.339 casos (80% de todos los casos). Si se emplean únicamente 5 submuestras sólo se recogerán el 3,89% de los ejemplos de la muestra de entrenamiento, siempre y cuando todos sean diferentes.

Por este motivo se decide emplear además otro tamaño de submuestras que denominaremos TamMax, y también variar el número de submuestras.

Por un lado tenemos los dos valores posibles para el tamaño de las submuestras:

- El tamaño de la clase minoritaria del conjunto de entrenamiento.

- Por otro lado el tamaño máximo al que se pueden generar las submuestras manteniendo una distribución de clases del 50 %, sin reemplazo, es decir, eliminar el número de casos de la clase mayoritaria que sobran.

La nomenclatura empleada será TamSizeOfMinClass para referirse a las muestras de tamaño dado por la clase minoritaria y TamMax para las submuestras al tamaño máximo. Haremos referencia al número de submuestras mediante N_S.

Se han empleado dos cantidades de submuestras fijo, 5 y 200 submuestras y se ha variado el número de submuestras al número de muestras necesarias para "envolver" o garantizar el tamaño de la muestra de entrenamiento a un porcentaje determinado teniendo en cuenta el tamaño de las submuestras, es decir, tamaño de la clase minoritaria. Estos porcentajes son 120%, 200%, 400% y finalmente 600%.

En concreto para cada base de datos se han empleado los dos tamaños de muestras, y el número de submuestras indicado en la Tabla 3.

Teniendo en cuenta los 6 valores que indican el número de submuestras empleados y los dos tamaños posibles, para cada base de datos se recogen 12 resultados.

En la siguiente ilustración se aprecia el proceso llevado a cabo para generar las submuestras para cada uno de los tamaños de submuestras a partir de la muestra original.

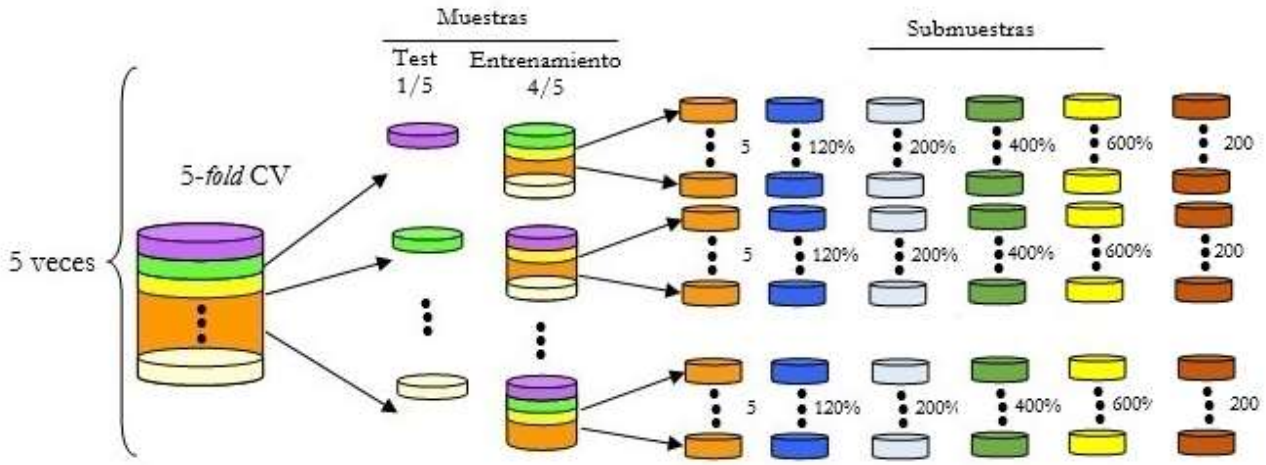


Ilustración 59 Proceso de la creación de submuestras para el primer experimento

Por cada base de datos, cada tipo de submuestra (TamMax y TamSizeOfMinClass) y cada una de las diferentes cantidades de submuestras (5, 200, 120%, 200%, 400% y al 600%) se han construido 25 árboles de clasificación consolidados (5 x 5 *cross validation*), lo que genera un total de 300 árboles CT.

Como disponemos de 33 bases de datos para el estudio al final se habrán generado 9900 árboles CT.

#BDs				N_S Fijo		N_S para cubrir Tam en p%			
33	Tamaños de las submuestras								
Data-set	sizeOfMinClass	Max	Tam para Submus	5	200	120	200	400	600
Abalone19	32,14	52,00	26,00	5	200	156	260	520	780
Yeast6	36,95	60,00	30,00	5	200	49	81	161	241
Yeast5	43,93	71,00	36,00	5	200	41	68	136	203
Yeast4	50,90	82,00	41,00	5	200	35	59	117	175
Yeast2vs8	20,00	33,00	17,00	5	200	29	49	97	145
Glass5	8,99	15,00	8,00	5	200	29	48	96	143
Abalone9vs18	41,30	67,00	34,00	5	200	22	36	71	107
Glass4	12,99	21,00	11,00	5	200	20	33	66	99
Ecoli4	22,65	37,00	19,00	5	200	18	30	60	90
Glass2	18,79	31,00	16,00	5	200	14	23	46	69
Vowel0	89,02	143,00	72,00	5	200	14	23	45	67
Page-blocks0	559,79	896,00	448,00	5	200	12	20	40	59
Ecoli3	36,56	59,00	30,00	5	200	12	19	37	56
Yeast3	162,94	261,00	131,00	5	200	11	19	37	55
Glass6	29,00	47,00	24,00	5	200	9	15	30	45
Segment0	329,12	527,00	264,00	5	200	9	15	29	43
Ecoli2	52,01	84,00	42,00	5	200	8	13	26	39
New-thyroid1	35,00	57,00	29,00	5	200	8	13	25	37
New-thyroid2	36,31	59,00	30,00	5	200	8	12	24	36
Ecoli1	77,01	124,00	62,00	5	200	6	9	18	27
Vehicle0	199,99	320,00	160,00	5	200	6	9	17	26
Glass0123vs456	51,00	82,00	41,00	5	200	6	9	17	26
Haberman	83,91	135,00	68,00	5	200	5	8	15	22
Vehicle1	240,01	385,00	193,00	5	200	5	8	15	22
Vehicle2	240,01	385,00	193,00	5	200	5	8	15	22
Vehicle3	240,01	385,00	193,00	5	200	5	8	15	22
Yeast1	429,02	687,00	344,00	5	200	5	7	14	21
Glass0	70,00	112,00	56,00	5	200	5	7	13	19
Iris0	50,00	80,00	40,00	5	200	5	7	13	19
Pima	267,57	429,00	215,00	5	200	5	6	12	18
Ecoli0vs1	77,00	124,00	62,00	5	200	5	6	12	18
Wisconsin	239,05	383,00	192,00	5	200	5	6	12	18
Glass1	75,99	122,00	61,00	5	200	5	6	12	17
Mean	119,97	192,58	96,61	**5,00**	**200,00**	**17,48**	**28,48**	**56,45**	**84,42**
Median	52,01	84,00	42,00	5,00	200,00	8,00	13,00	26,00	39,00

Tabla 3 Tamaño y número de submuestras empleadas para cada una de las bases de datos

[3]

En el anexo 1 se puede encontrar los resultados obtenidos para cada una de las 33 bases de datos que forman el estudio. Como se puede apreciar, en el primer anexo se ha calculado a parte de la media del valor GM empleado en esta experimentación, la media de las principales medidas de bondad AUC, Error, GM, F-Value, Kappa y TPrate.- obtenida como resultado de la clasificación de los conjuntos de test en los 25 árboles de clasificación creados-.

No obstante para realizar la comparación directa con los 22 algoritmos propuestos en el artículo de Fernández et al únicamente se ha empleado el GM, puesto que éste es el que se usó y el único disponible.

[3] Las siguientes bases de datos representadas en las tablas de este capítulo en rojo han sido incluidas en el estudio sin podar, debido a que en la versión podada los arboles únicamente estaban formados por el nodo raíz: Abalone19, Yeas6, Yeast4, Abalone9Vs18, Glass2, Haberman, Vehicle1 y Vehicle3.

Una vez finalizada la exposición de la metodología experimental empleada, se pasa a mostrar los resultados obtenidos en el artículo *Genetic based machine learning for rule induction* mediante la Tabla 4, que muestra los resultados obtenidos en función del GM y a su vez también a través de la Tabla 5, mostrando el ranking de los 11 mejores algoritmos.

La selección de éstos 11 mejores algoritmos sobre los 22 es resultado de la selección jerárquica del mejor de cada familia en el caso de los GBML y los 6 clásicos.

#BDs	Results GM Test										
33	GBML					Classical methods					
Data-set	UCS	SIA	OCEC	Gassist	Oblique-DT	CART	AQ	CN2	C45	C45-Rules	Ripper
Abalone19	0,00	0,00	61,72	0,00	7,53	0,00	56,64	0,00	0,00	0,00	23,20
Yeast6	75,53	51,47	54,24	0,00	62,80	68,03	55,87	0,00	56,60	73,27	79,99
Yeast5	78,14	77,71	78,04	72,67	80,82	71,45	68,12	41,33	87,51	84,89	87,39
Yeast4	59,96	26,21	58,31	17,16	55,56	10,84	38,39	21,21	41,97	55,08	63,82
Yeast2vs8	56,57	10,00	71,45	62,74	69,05	50,86	53,66	58,08	10,00	28,13	71,89
Glass5	33,97	67,79	88,34	13,97	93,65	93,90	71,86	0,00	88,04	88,04	73,47
Abalone9vs1	41,36	40,17	48,65	40,41	54,11	25,14	19,79	20,78	44,82	47,55	53,37
Glass4	36,12	95,41	69,72	62,51	62,70	76,37	74,23	58,55	55,37	58,22	78,89
Ecoli4	68,73	65,21	76,06	78,02	88,09	77,88	55,98	63,98	82,33	85,51	88,41
Glass2	9,61	19,34	65,99	0,00	32,05	25,08	65,72	0,00	60,08	60,08	36,66
Vowel0	96,68	100,00	62,38	93,60	95,59	84,68	60,06	48,83	96,83	96,83	95,73
Page-blocks	85,04	86,59	75,14	80,87	90,07	71,04	11,70	59,29	91,95	93,09	93,57
Ecoli3	46,29	61,69	63,29	67,11	67,15	68,12	45,38	56,04	67,38	69,56	80,69
Yeast3	89,46	81,71	90,55	85,25	82,05	80,82	81,17	18,56	85,10	87,23	91,81
Glass6	88,49	90,63	89,15	84,74	82,29	80,89	88,10	81,14	79,42	79,21	84,49
Segment0	98,88	99,44	93,37	98,00	98,41	98,60	92,83	63,88	98,24	98,39	98,82
Ecoli2	50,81	86,90	53,49	87,91	86,47	81,24	55,42	51,73	85,14	86,08	86,18
New-thyroid	89,55	91,87	83,68	96,21	95,65	94,62	86,56	85,66	90,84	91,78	92,88
New-thyroid	91,59	90,37	93,51	97,67	93,29	91,18	92,89	85,66	93,53	93,27	92,75
Ecoli1	58,09	73,85	58,72	86,22	81,55	82,91	42,12	55,93	85,38	85,70	91,52
Vehicle0	88,64	80,54	76,49	90,77	90,07	92,28	69,86	29,88	92,90	92,83	90,88
Glass0123vs	87,07	86,97	85,35	88,57	87,19	91,01	76,51	82,19	91,31	90,98	90,53
Haberman	47,87	52,29	45,26	45,26	59,77	25,44	13,85	23,20	35,63	35,06	55,54
Vehicle1	62,74	57,71	58,95	56,23	66,92	53,63	43,78	30,94	64,70	61,75	71,28
Vehicle2	92,63	85,09	88,41	94,75	94,48	93,52	89,08	49,64	95,57	93,64	95,33
Vehicle3	69,14	53,18	65,12	58,83	65,21	50,07	42,55	39,67	61,37	67,99	71,01
Yeast1	68,76	57,08	41,15	61,23	61,20	53,97	35,03	10,14	62,83	68,96	67,72
Glass0	30,93	77,20	68,40	81,03	74,53	74,60	63,11	21,62	81,36	78,30	76,80
Iris0	98,97	100,00	89,83	99,49	97,89	100,00	92,25	91,73	98,97	98,97	97,89
Pima	69,14	63,17	50,38	65,64	66,94	70,25	14,77	38,62	68,72	68,95	69,66
Ecoli0vs1	20,00	96,90	87,73	97,97	96,90	96,86	87,05	88,50	98,31	98,31	95,82
Wisconsin	96,22	96,61	95,38	95,23	93,21	92,23	85,36	94,84	94,50	94,81	96,38
Glass1	55,40	74,51	50,79	70,07	68,57	73,36	44,10	45,38	71,45	69,35	73,96
Mean	**64,92**	**69,62**	**70,88**	**67,58**	**75,81**	**69,72**	**59,81**	**45,97**	**73,28**	**75,21**	**79,34**
Median	68,76	77,20	69,72	78,02	81,55	76,37	60,06	48,83	82,33	84,89	84,49

Tabla 4 Resultado en función del GM para las 33 imbalanced datasets y los 11 algoritmos seleccionados del artículo

Basándonos en la media se aprecia que el algoritmo que mejores resultados obtiene es el Ripper, después el ObliqueDT y muy seguidamente el algoritmo C4.5Rules. El C4.5 ocupa la cuarta posición. De los 5 algoritmos que muestran un mejor desempeño en estas bases de datos desbalanceadas 3 pertenecen al grupo de los algoritmos clásicos y 2 al conjunto de algoritmos GBML.

	Results GM Test										
	GBML					Classical methods					
	UCS	SIA	OCEC	Gassist	Oblique-DT	CART	AQ	CN2	C45	C45-Rules	Ripper
Rankings	7	6	9	5	3	8	10	11	4	2	1

Tabla 5 Ranking de los 11 algoritmos seleccionados (5 GBML + 6 classical)

Los resultados de los 5 mejores algoritmos si nos basamos en el ranking como medida de rendimiento, son casi los mismos con la única diferencia del que ocupa la quinta posición, mientras que basándonos en la media el que ocupa esta posición es el OCEC. Si miramos el ranking el que consigue la quinta es el Gassist.

6.3 Resultados

Tal y como ya se ha enunciado unas cuantas veces, el objetivo principal de este trabajo es comparar el CTC con los algoritmos propuestos por Fernández et al en [FGLBH10] ante bases de datos muy desbalanceadas y de tamaño reducido en los que el aprendizaje resulta más difícil. Esta comparación se lleva a cabo únicamente desde el punto de vista de la capacidad de aprendizaje dejando al lado otros posibles puntos de vista como podrían ser la estabilidad y la complejidad de los clasificadores.

Para medir la capacidad de aprendizaje se ha empleado el valor de la media geométrica o GM.

Como se ha descrito en la sección metodología experimental, se ha realizado una extensa experimentación con este fin. La comparación se realiza frente a los 22 algoritmos (6 métodos clásicos y 12 GBML). Por lo tanto se han evaluado estos 23 algoritmos sobre las 33 bases de datos del contexto desbalanceado.

Como en el artículo de Fernández et al [FGLBH10] se ha empleado una metodología jerárquica en la comparación. Primero se analizan las 12 diferentes estrategias empleadas para construir los árboles consolidados, para después seleccionar entre ellas las que mejor se adapten al contexto.

En la gráfica que se encuentra en la ilustración 60 se aprecia el comportamiento que ofrece el algoritmo CTC en función del número de submuestras y también teniendo en cuenta el tamaño de las mismas. Esta gráfica permite abstraer el comportamiento global y no fijarnos en ninguna base de datos concreta, ni tener en cuenta otras medidas interesantes como podría ser el ranking medio de las bases de datos.

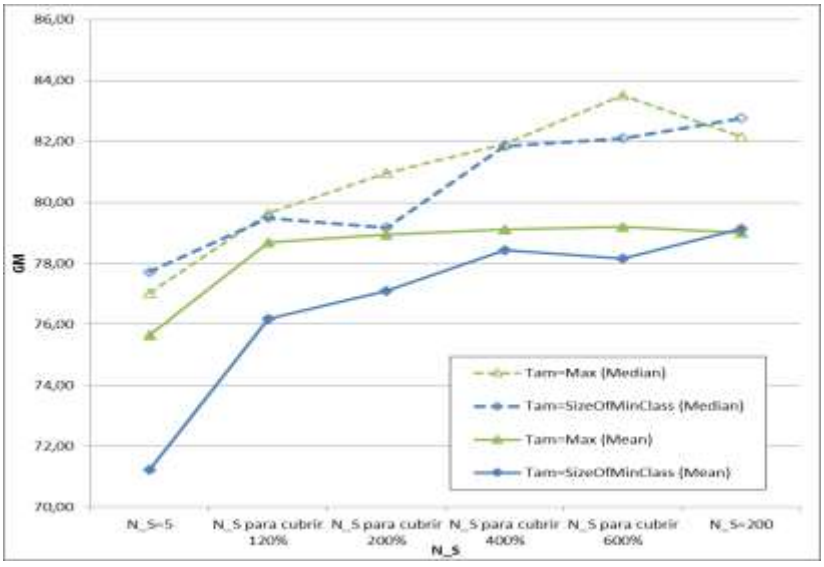


Ilustración 60 Valor del GM en función del número de submuestras y del tamaño de las mismas para el CTC

En esta ilustración se aprecia de manera visual la media y mediana de la tasa del GM obtenidas por las 33 bases de datos. Según puede observarse el tamaño de la submuestras está directamente relacionado con el desempeño del clasificador. Este factor tiene más importancia si además el número de submuestras empleado es reducido.

A medida que se amplía el número de submuestras los resultados obtenidos se equiparan y se aprecia que existe menos diferencia hasta llegar a 200 submuestras en el que se podría afirmar que obtienen el mismo resultado, tanto para la media, como para la mediana. La razón que radica en ello es que con un número de submuestras elevado el *coverage* o tanto por ciento de los elementos de la case minoritaria que se cubre con el conjunto de submuestras es del 100% o muy cercano, tanto para tamaños de submuestra al mínimo, como para tamaños al máximo.

No obstante queda claro que las submuestras creadas al tamaño máximo obtienen mejores resultados. De la gráfica también se desprende que la mejora más notable se encuentra al pasar de un número fijo de submuestas que hemos colocado en 5 al número de submuestras necesario para cubrir el 120%. A medida que aumentamos este factor se sigue obteniendo mejores resultados en media pero la diferencia no es tan notable, no podríamos calificarla como lineal.

Dif. Rel. respecto a N_S anterior							Dif. Rel. respecto a N_S anterior					
1	2	3	4	5	6		1	2	3	4	5	6
52,38	-15,16%	2,69%	-4,83%	-8,26%	-12,49%	Abalone19	58,99	-13,49%	-12,68%	18,10%	-16,44%	1,06%
82,09	-0,80%	-2,85%	3,56%	2,39%	-8,09%	Yeast6	83,93	-1,06%	0,60%	-0,26%	-1,45%	2,29%
43,31	92,88%	7,04%	-2,09%	0,84%	0,33%	Yeast5	13,42	535,38%	5,47%	-1,71%	-2,19%	1,96%
77,02	-1,56%	-1,36%	-5,92%	4,38%	-3,21%	Yeast4	79,76	-0,37%	-2,28%	-1,84%	0,39%	0,64%
72,83	0,00%	0,00%	0,00%	0,00%	0,00%	Yeast2vs8	72,83	0,00%	0,00%	0,00%	0,00%	0,00%
49,84	59,80%	12,62%	4,68%	0,62%	-1,12%	Glass5	27,60	57,79%	60,15%	17,35%	-6,44%	17,01%
68,78	1,51%	-1,90%	0,77%	-0,39%	5,59%	Abalone9vs18	69,62	0,94%	1,43%	-4,58%	3,08%	1,75%
47,27	61,36%	-12,15%	18,09%	-4,72%	3,58%	Glass4	16,15	166,69%	-0,82%	53,11%	2,36%	7,51%
66,49	25,16%	-2,12%	2,59%	-0,05%	-0,80%	Ecoli4	45,80	61,36%	4,30%	3,68%	3,59%	1,24%
65,93	-2,28%	-3,72%	-12,24%	3,10%	5,37%	Glass2	57,22	6,27%	-1,26%	-3,87%	4,74%	4,46%
93,83	0,53%	0,87%	-1,08%	-1,15%	1,51%	Vowel0	90,46	-1,21%	2,83%	2,69%	-2,72%	2,42%
75,03	4,95%	0,82%	0,97%	-0,21%	0,45%	Page-blocks0	78,82	5,55%	-0,33%	0,98%	-1,28%	0,13%
69,28	2,75%	3,64%	-2,51%	7,03%	-2,80%	Ecoli3	51,00	44,44%	-3,66%	2,37%	-0,37%	6,36%
87,22	0,13%	-1,88%	0,90%	-1,56%	1,04%	Yeast3	88,53	-2,25%	-1,24%	0,14%	-0,03%	0,70%
87,99	-1,53%	-1,85%	0,59%	-0,62%	-0,27%	Glass6	89,71	-1,87%	-0,74%	-1,48%	-1,06%	-2,30%
98,59	-0,09%	0,25%	-0,18%	0,33%	-0,11%	Segment0	98,07	0,02%	0,53%	-0,17%	0,33%	0,14%
87,07	-1,80%	2,46%	-0,35%	-0,72%	-0,98%	Ecoli2	83,04	1,97%	1,25%	-2,18%	2,91%	-1,38%
95,71	-0,16%	0,01%	0,65%	0,37%	-1,41%	New-thyroid1	89,22	5,37%	-0,06%	-1,19%	-0,07%	0,41%
94,72	-0,62%	0,49%	0,97%	-0,48%	1,34%	New-thyroid2	93,04	0,37%	0,03%	-1,16%	1,70%	-0,95%
87,57	-3,21%	0,92%	1,00%	0,60%	-0,16%	Ecoli1	85,69	0,16%	-2,73%	2,34%	-1,28%	-2,82%
91,98	-0,28%	0,57%	-0,12%	0,07%	0,34%	Vehicle0	90,84	0,30%	-4,01%	4,58%	-0,57%	1,95%
86,78	0,32%	-0,23%	-1,06%	0,40%	-1,20%	Glass0123vs456	85,35	1,90%	-1,22%	0,84%	1,61%	-3,08%
50,40	-2,39%	2,19%	-1,02%	-0,69%	0,65%	Haberman	49,46	-4,92%	0,53%	-1,56%	2,27%	-4,10%
55,71	-0,35%	0,73%	0,45%	0,08%	0,56%	Vehicle1	54,03	0,71%	0,87%	1,55%	0,19%	0,98%
78,21	-1,12%	4,70%	-2,21%	2,36%	1,38%	Vehicle2	77,71	-11,48%	13,31%	-0,35%	0,08%	4,38%
57,06	-0,66%	0,53%	0,72%	-0,82%	0,35%	Vehicle3	56,50	-1,97%	1,70%	0,02%	1,03%	0,00%
65,32	0,56%	-1,30%	0,55%	-0,06%	-0,92%	Yeast1	62,97	2,62%	-1,97%	1,65%	-1,14%	0,56%
80,37	-2,21%	1,61%	-1,62%	0,48%	1,37%	Glass0	75,97	4,64%	-0,40%	0,99%	-0,09%	-0,78%
98,97	0,00%	0,00%	0,00%	0,00%	0,00%	Iris0	98,77	-0,21%	-0,42%	0,63%	-0,21%	-0,42%
69,12	-2,42%	0,59%	-0,20%	-0,50%	-2,78%	Pima	68,09	0,24%	-0,24%	0,37%	-0,63%	1,13%
98,16	0,00%	0,00%	-0,07%	0,00%	0,15%	Ecoli0vs1	97,60	-0,79%	0,65%	0,50%	-0,21%	0,21%
91,82	0,12%	0,26%	-0,31%	0,05%	0,00%	Wisconsin	92,02	0,54%	1,26%	-1,09%	-0,36%	0,20%
69,62	0,78%	-2,88%	4,41%	-1,36%	1,08%	Glass1	68,13	2,56%	1,79%	-0,85%	-0,94%	2,29%
75,65	**6,49%**	**0,33%**	**0,16%**	**0,05%**	**-0,34%**	Mean	**71,22**	**26,07%**	**1,90%**	**2,71%**	**-0,40%**	**1,33%**
77,02	-0,09%	0,26%	0,00%	0,00%	0,00%	Median	77,71	0,37%	0,00%	0,14%	-0,09%	0,64%

Tabla 6 Diferencia relativa para submuestras al tamaño TamMax en la izquierda y TamSizeOfMinClass a la derecha

En la Tabla anterior se muestra, para cada una de las bases de datos del estudio, la diferencia relativa que existe entre los resultados obtenidos para un número de submuestras concreto y los resultados que se obtienen al aumentarlo. En concreto, la columna etiquetada con el uno representa los resultados obtenidos para una configuración del CTC con 5 submuestras. La columna dos muestra la ganancia relativa obtenida aumentar el número de submuestras al 120%, en la columna 3, se muestra la ganancia obtenida al pasar de 120% a 200%, en la cuarta la producida al pasar de 200% a 400%, la quinta de 400% 600% y para finalizar en la columna 6 la ganancia alcanzada al pasar de submuestras al 600% a un número fijo de 200 submuestras.

Como se ha apreciado en la gráfica, el mayor aumento se da para tamaño TamSizeOfMinClass y al pasar de 5 submuestras al número de submuestras dado por el 120%, alcanzando una mejora del 27%, mientras que para el otro tamaño de submuestras roza el 6.5% aunque la mediana se mantiene cercana al cero siempre.

La diferencia entre las dos variantes o configuraciones del CTC se encuentra en que la diferencia relativa entre el número de submuestras es mayor para submuestras TamSizeOfMinClass. Podría concluirse que es interesante aumentar el número de submuestras si son generadas con tamaño al mínimo de la clase minoritaria mientras que pasa a ser algo con menos impacto si las submuestas son generadas al tamaño máximo.

Results CTC GM Test
Tam=SizeOfMinClass

1	2	3	4	5	6	Max	N_S (Max)		Results CTC GM Test Tam=Max 1	2	3	4	5	6	Max	N_S (Max)
58,99	51,03	44,56	52,63	43,98	44,44	58,99	5	Abalone19	**52,38**	44,44	45,63	43,43	39,84	34,87	52,38	5
83,93	83,03	83,53	83,31	82,10	**83,98**	83,98	241	Yeast6	82,09	81,43	79,11	81,93	**83,89**	77,10	83,89	200
13,42	85,29	**89,95**	88,41	86,47	88,16	89,95	68	Yeast5	43,31	83,53	**89,40**	87,54	88,27	88,56	89,40	68
79,76	79,47	77,65	76,22	76,52	77,01	79,76	5	Yeast4	**77,02**	75,82	74,80	70,37	73,45	71,09	77,02	5
72,83	**72,83**	**72,83**	**72,83**	**72,83**	**72,83**	72,83	5	Yeast2vs8	**72,83**	**72,83**	**72,83**	**72,83**	**72,83**	**72,83**	72,83	5
27,60	43,56	69,75	81,85	76,58	**89,61**	89,61	200	Glass5	49,84	79,64	89,69	93,89	**94,47**	93,41	94,47	143
69,62	70,28	71,29	68,02	70,12	**71,35**	71,35	200	Abalone9vs18	68,78	69,81	68,49	69,02	68,75	**72,59**	72,59	200
16,15	43,07	42,71	65,40	66,94	**71,97**	71,97	200	Glass4	47,27	76,28	67,01	**79,13**	75,40	78,10	79,13	66
45,80	73,91	77,09	79,92	82,79	**83,82**	83,82	200	Ecoli4	66,49	83,22	81,45	**83,56**	83,52	82,85	83,56	60
57,22	60,81	60,04	57,72	60,46	**63,15**	63,15	200	Glass2	**65,93**	64,42	62,03	54,44	56,13	59,14	65,93	5
90,46	89,36	91,89	**94,36**	91,80	94,02	94,36	45	Vowel0	93,83	94,33	**95,14**	94,12	93,04	94,44	95,14	23
78,82	83,19	82,92	**83,73**	82,66	82,76	83,73	40	Page-blocks0	75,03	78,74	79,38	80,15	79,98	**80,34**	80,34	200
51,00	73,66	70,96	72,64	72,38	**76,98**	76,98	200	Ecoli3	69,28	71,19	73,78	71,92	**76,98**	74,83	76,98	56
88,53	86,54	85,47	85,59	85,57	86,16	88,53	5	Yeast3	87,22	**87,33**	85,69	86,46	85,11	86,00	87,33	11
89,71	88,04	87,38	86,08	85,18	83,21	89,71	5	Glass6	**87,99**	86,65	85,04	85,54	85,02	84,78	87,99	5
98,07	98,08	98,61	98,44	98,77	**98,91**	98,91	200	Segment0	98,59	98,50	98,75	98,57	**98,89**	98,78	98,89	43
83,04	84,67	85,73	83,86	**86,30**	85,11	86,30	39	Ecoli2	87,07	85,50	**87,60**	87,29	86,67	85,82	87,60	13
89,22	**94,01**	93,95	92,83	92,76	93,15	94,01	8	New-thyroid1	95,71	95,57	95,58	96,20	**96,55**	95,19	96,55	37
93,04	93,38	93,41	92,32	**93,89**	93,00	93,89	36	New-thyroid2	94,72	94,13	94,60	95,52	95,06	**96,33**	96,33	200
85,69	**85,82**	83,48	85,43	84,33	81,96	85,82	6	Ecoli1	**87,57**	84,76	85,54	86,40	86,92	86,78	87,57	5
90,84	91,11	87,46	91,47	90,95	**92,72**	92,72	200	Vehicle0	91,98	91,72	92,24	92,13	92,20	**92,51**	92,51	200
85,35	86,97	85,91	86,63	**88,03**	85,32	88,03	26	Glass0123vs456	86,78	**87,05**	86,85	85,94	86,28	85,24	87,05	6
49,46	47,03	47,28	46,54	47,59	45,64	49,46	5	Haberman	**50,40**	49,20	50,28	49,77	49,43	49,75	50,40	5
54,03	54,41	54,89	55,74	55,84	**56,39**	56,39	200	Vehicle1	55,71	55,51	55,91	56,17	56,22	**56,53**	56,53	200
77,71	68,79	77,95	77,67	77,74	**81,14**	81,14	200	Vehicle2	78,21	77,33	80,96	79,17	81,04	**82,16**	82,16	200
56,50	55,38	56,32	56,33	**56,91**	56,91	56,91	22	Vehicle3	57,06	56,69	56,99	**57,40**	56,93	57,13	57,40	15
62,97	**64,62**	63,34	64,39	63,65	64,01	64,62	5	Yeast1	65,32	**65,69**	64,84	65,20	65,16	64,56	65,69	5
75,97	79,49	79,18	**79,96**	79,89	79,27	79,96	13	Glass0	**80,37**	78,59	79,86	78,57	78,95	80,03	80,37	5
98,77	98,56	98,15	**98,77**	98,56	98,15	98,77	5	Iris0	**98,97**	**98,97**	**98,97**	**98,97**	**98,97**	**98,97**	98,97	5
68,09	68,25	68,09	68,34	67,91	**68,68**	68,68	200	Pima	**69,12**	67,45	67,85	67,71	67,38	65,51	69,12	5
97,60	96,83	97,46	**97,95**	97,74	**97,95**	97,95	12	Ecoli0vs1	98,16	98,16	98,16	98,09	98,09	**98,24**	98,24	200
92,02	92,52	**93,69**	92,67	92,33	92,51	93,69	6	Wisconsin	91,82	91,93	**92,17**	91,89	91,93	91,93	92,17	6
68,13	69,87	71,13	70,52	69,86	**71,46**	71,46	200	Glass1	69,62	70,16	68,15	**71,15**	70,18	70,94	71,15	12
71,22	**76,18**	**77,09**	**78,44**	**78,16**	**79,14**	**80,53**	**90,97**	Mean	**75,65**	**78,68**	**78,93**	**79,10**	**79,20**	**79,01**	**80,84**	**67,09**
77,71	79,49	79,18	81,85	82,10	82,76	83,82	39,00	Median	77,02	79,64	80,96	81,93	83,52	82,16	83,56	15,00

Tabla 7 Número de submuestras óptimo para cada base datos, a la izquierda TamSizeOfMinClass y a la derecha para TamMax

Esta conclusión se refrenda en la Tabla 7. En las columnas del uno al seis se muestra el valor GM para las distintas configuraciones del CTC, 5, al 120%, 200%, 400%, 600% y 200 submuestras respectivamente. La última recoge el número de submuestras necesarios para obtener el valor óptimo del GM.

Si atendemos al número de submuestras que en cada caso han sido necesarias para obtener los mejores resultados en el caso de TamSizeOfMinClass, el número de submuestras aumenta frente al TamSizeOfMinClass, la explicación de esto vuelve a basarse en el *coverage* de las muestras.

Con el objetivo de detectar si existen diferencias significativas entre las distintas variantes en las que se ha probado el CTC, el primer paso es aplicar el test de Friedman, que es el encargado de detectar si existen diferencias o no.

El siguiente paso es aplicar un test post hoc más poderoso para descubrir sobre qué par de algoritmos aparecen las diferencias significativas. Como se ha expuesto en el capítulo 2, se aplicarán los test de Nemenyi y Bonferroni-Dunn. Se muestran los resultados de ambos test para los 6 números de submuestras empleados en los dos tamaños de submuestras en los diagramas CD de las ilustraciones 61 para TamSizeOfMinClass y 62 para TamMax. La línea de la parte inferior del diagrama indica la máxima diferencia que puede haber entre los rangos medios de dos clasificadores según el test de Nemenyi para que no haya entre ellos diferencias significativas. Sin entrar en gran detalle este valor crítico se calcula a través de una fórmula y unas tablas.

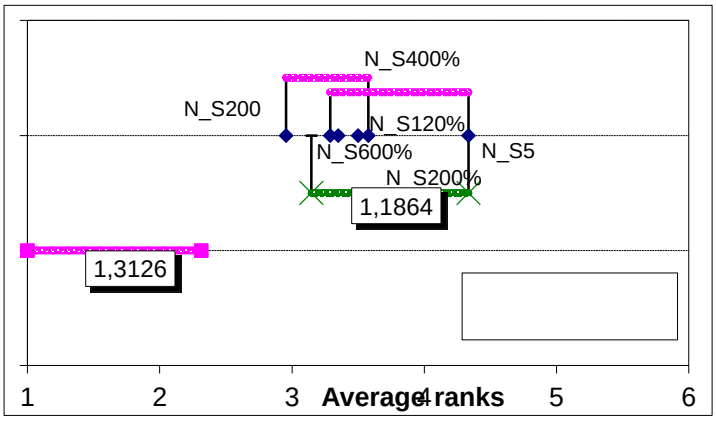


Ilustración 61 Diagrama CD para los test de Nemenyi y Bonferroni-Dunn para el algoritmo CTC y tamaño de submuestras TamSizeOfMinClass

N_S=200	**2,95**
N_S 600%	**3,29**
N_S 400%	**3,35**
N_S 120%	**3,50**
N_S 200%	**3,58**
N_S=5	**4,33**

En base a las líneas mostradas para el test de Nemenyi en el caso de submuestras al tamaño TamSizeOfMinClass se aprecia que existen diferencias significativas entre N_S=200 y N_S=5. En el mismo diagrama puede observarse el resultado del test de Bonferroni-Dunn para el que se ha escogido el algoritmo CTC en su versión de 5 submuestras como algoritmo de control. Los resultados son cualitativamente iguales, por lo tanto derivan en las mismas conclusiones. La línea que une los algoritmos en verde es la línea de Bonferroni-Dunn cuyo valor crítico es de 1,1864.

Para el caso del algoritmo CTC y el conjunto del número de submuestras distinto empleado para tamaños TamMax mediante el test de Nemenyi, cuyo valor crítico sigue

siendo de 1,3126, se observa que no existen diferencias significativas para ninguna de las configuraciones. Lo mismo sucede con el test de Bonferroni-Dunn, (como en el caso anterior se elige al CTC con 5 submuestras como algoritmo de control para concluir que no existen diferencias significativas). Además, junto a la línea, se muestra el valor de la distancia crítica para este test, que es de 1,1864.

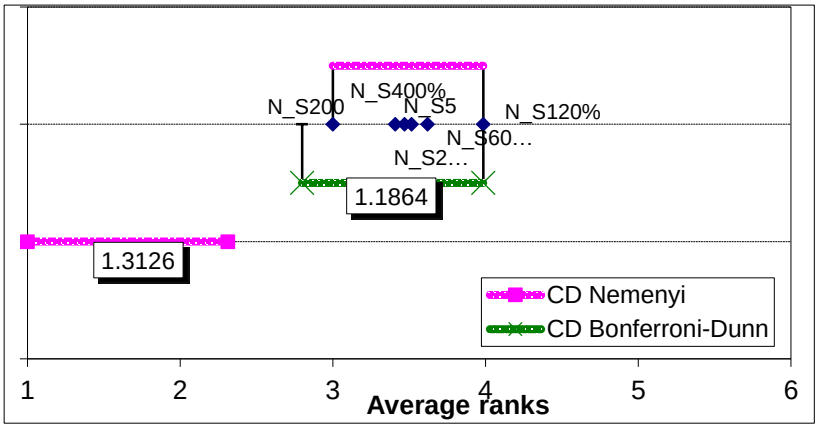


Ilustración 62 Diagrama CD para los test de Nemenyi y Bonferroni-Dunn para el algoritmo CTC y tamaño de submuestras Tammax

N_S=200	**3,00**
N_S 400%	**3,41**
N_S 200%	**3,47**
N_S=5	**3,52**
N_S 600%	**3,62**
N_S 120%	**3,98**

Entrando en más detalle, si se tienen en cuenta los resultados para cada base de datos, véase Tabla 8, se constata que la versión del CTC que mejores resultados obtiene es con 200 submuestras y tamaño de submuestras TamMax.

#BDs	Results CTC GM Test											
33	Tam=SizeOfMinClass						Tam=Max					
Data-set	N_S=5	N_S=200	N_S para cubrir 120%	N_S para cubrir 200%	N_S para cubrir 400%	N_S para cubrir 600%	N_S=5	N_S=200	N_S para cubrir 120%	N_S para cubrir 200%	N_S para cubrir 400%	N_S para cubrir 600%
Abalone19	58,99	44,56	51,03	52,63	43,98	44,44	52,38	45,63	44,44	43,43	39,84	34,87
Yeast6	83,93	82,10	83,03	83,53	83,31	83,98	82,09	83,89	81,43	79,11	81,93	77,10
Yeast5	13,42	86,47	85,29	89,95	88,41	88,16	43,31	88,27	83,53	89,40	87,54	88,56
Yeast4	79,76	77,01	79,47	77,65	76,22	76,52	77,02	71,09	75,82	74,80	70,37	73,45
Yeast2vs8	72,83	72,83	72,83	72,83	72,83	72,83	72,83	72,83	72,83	72,83	72,83	72,83
Glass5	27,60	89,61	43,56	69,75	81,85	76,58	49,84	93,41	79,64	89,69	93,89	94,47
Abalone9vs18	69,62	71,35	70,28	71,29	68,02	70,12	68,78	72,59	69,81	68,49	69,02	68,75
Glass4	16,15	71,97	43,07	42,71	65,40	66,94	47,27	78,10	76,28	67,01	79,13	75,40
Ecoli4	45,80	83,82	73,91	77,09	79,92	82,79	66,49	82,85	83,22	81,45	83,56	83,52
Glass2	57,22	63,15	60,81	60,04	57,72	60,46	65,93	59,14	64,42	62,03	54,44	56,13
Vowel0	90,46	94,02	89,36	91,89	94,36	91,80	93,83	94,44	94,33	95,14	94,12	93,04
Page-blocks0	78,82	82,76	83,19	82,92	83,73	82,66	75,03	80,34	78,74	79,38	80,15	79,98
Ecoli3	51,00	76,98	73,66	70,96	72,64	72,38	69,28	74,83	71,19	73,78	71,92	76,98
Yeast3	88,53	86,16	86,54	85,47	85,59	85,57	87,22	86,00	87,33	85,69	86,46	85,11
Glass6	89,71	83,21	88,04	87,38	86,08	85,18	87,99	84,78	86,65	85,04	85,54	85,02
Segment0	98,07	98,91	98,08	98,61	98,44	98,77	98,59	98,78	98,50	98,75	98,57	98,89
Ecoli2	83,04	85,11	84,67	85,73	83,86	86,30	87,07	85,82	85,50	87,60	87,29	86,67
New-thyroid1	89,22	93,15	94,01	93,95	92,83	92,76	95,71	95,19	95,57	95,58	96,20	96,55
New-thyroid2	93,04	93,00	93,38	93,41	92,32	93,89	94,72	96,33	94,13	94,60	95,52	95,06
Ecoli1	85,69	81,96	85,82	83,48	85,43	84,33	87,57	86,78	84,76	85,54	86,40	86,92
Vehicle0	90,84	92,72	91,11	87,46	91,47	90,95	91,98	92,51	91,72	92,24	92,13	92,20
Glass0123vs45	85,35	85,32	86,97	85,91	86,63	88,03	86,78	85,24	87,05	86,85	85,94	86,28
Haberman	49,46	45,64	47,03	47,28	46,54	47,59	50,40	49,75	49,20	50,28	49,77	49,43
Vehicle1	54,03	56,39	54,41	54,89	55,74	55,84	55,71	56,53	55,51	55,91	56,17	56,22
Vehicle2	77,71	81,14	68,79	77,95	77,67	77,74	78,21	82,16	77,33	80,96	79,17	81,04
Vehicle3	56,50	56,91	55,38	56,32	56,33	56,91	57,06	57,13	56,69	56,99	57,40	56,93
Yeast1	62,97	64,01	64,62	63,34	64,39	63,65	65,32	64,56	65,69	64,84	65,20	65,16
Glass0	75,97	79,27	79,49	79,18	79,96	79,89	80,37	80,03	78,59	79,86	78,57	78,95
Iris0	98,77	98,15	98,56	98,15	98,77	98,56	98,97	98,97	98,97	98,97	98,97	98,97
Pima	68,09	68,68	68,25	68,09	68,34	67,91	69,12	65,51	67,45	67,85	67,71	67,38
Ecoli0vs1	97,60	97,95	96,83	97,46	97,95	97,74	98,16	98,24	98,16	98,16	98,09	98,09
Wisconsin	92,02	92,51	92,52	93,69	92,67	92,33	91,82	91,93	91,93	92,17	91,89	91,93
Glass1	68,13	71,46	69,87	71,13	70,52	69,86	69,62	70,94	70,16	68,15	71,15	70,18
Mean	**71,22**	**79,04**	**76,18**	**77,34**	**78,18**	**78,29**	**75,65**	**79,53**	**78,68**	**78,87**	**79,00**	**78,85**
Median	77,71	82,10	79,49	79,18	81,85	82,66	77,02	82,85	79,64	80,96	81,93	81,04
Rankings	12	5	10	11	9	8	3	1	7	2	4	6

Tabla 8 Resultado en función de GM para los 33 imbalanced datasets y los 12 distintas configuraciones del algoritmo CTC

Para llegar a esta conclusión nos basamos en el ranking, cabe destacar que pese a ser la opción que mejores resultados obtiene, quizás sea interesante resaltar que la versión para número de submuestras dado por el valor necesario para cubrir el 200 % y tamaño TamMax podría resultar mejor, no desde el punto de vista de media, pero si desde el punto de vista de coste computacional. Ya que obtiene la segunda posición en ranking y el número medio de submuestras pasa a ser de 200 a 28,48 de media, con su correspondiente reducción en el coste computacional para construir los árboles. Estos datos se pueden consultar en la Tabla 3.

Como resultado de todo este análisis se han elegido las dos estrategias mencionadas en el párrafo anterior, como las dos mejores y más representativas.

Antes de comparar las actitudes de clasificación que poseen estas dos configuraciones del CTC en el ámbito del desbalanceo de clases frente a los 22 algoritmos, se muestran los resultados en una comparación entre el CTC y el C4.5. El motivo es que presumiblemente deberían tener un comportamiento similar y podría ser una buena forma de tener una idea real y concreta del desempeño del CTC.

#BDs	Results GM Test			Rankings	
33	Classical me	CTC Tam=Max		Classical me	CTC Tam=Max
Data-set	C45	N_S=200		C45	N_S=200
Abalone19	0,00	45,63		2	1
Yeast6	56,60	83,89		2	1
Yeast5	87,51	88,27		2	1
Yeast4	41,97	71,09		2	1
Yeast2vs8	10,00	72,83		2	1
Glass5	88,04	93,41		2	1
Abalone9vs18	44,82	72,59		2	1
Glass4	55,37	78,10		2	1
Ecoli4	82,33	82,85		2	1
Glass2	60,08	59,14		1	2
Vowel0	96,83	94,44		1	2
Page-blocks0	91,95	80,34		1	2
Ecoli3	67,38	74,83		2	1
Yeast3	85,10	86,00		2	1
Glass6	79,42	84,78		2	1
Segment0	98,24	98,78		2	1
Ecoli2	85,14	85,82		2	1
New-thyroid1	90,84	95,19		2	1
New-thyroid2	93,53	96,33		2	1
Ecoli1	85,38	86,78		2	1
Vehicle0	92,90	92,51		1	2
Glass0123vs456	91,31	85,24		1	2
Haberman	35,63	49,75		2	1
Vehicle1	64,70	56,53		1	2
Vehicle2	95,57	82,16		1	2
Vehicle3	61,37	57,13		1	2
Yeast1	62,83	64,56		2	1
Glass0	81,36	80,03		1	2
Iris0	98,97	98,97		1	2
Pima	68,72	65,51		1	2
Ecoli0vs1	98,31	98,24		1	2
Wisconsin	94,50	91,93		1	2
Glass1	71,45	70,94		1	2
Mean	73,28	79,53		1,58	1,42
Median	82,33	82,85		2,00	1,00

Tabla 9 Resultados medios y de ranking para el CTC con 200 submuestras y TamMax

En la Tabla 9 se encuentran los rankings de la comparación entre el CTC y el C4.5. De verde resalta el ganador para una base de datos en concreto, mostrándose los resultados para las 33 bases de datos que componen el estudio. Si bien se muestran los resultados únicamente del clasificador a estudio para una configuración de 200 submuestras y tamaño máximo, el algoritmo obtiene los mimos resultados en cuanto al ranking en las distintas configuraciones en las que se ha probado el CTC.

Se puede concluir viendo los resultados que en una comparación directa con C4.5 de las 20 bases de datos más desbalanceadas, por lo tanto las que suscitan mayor interés en este ámbito del *class imbalance*, y que tiene en común que el número de casos de la clase minoritaria es inferior al 17%, el CTC gana en 17 de ellas demostrando ser más adecuado para tratar con esta bases de datos.

Basándonos en el mismo análisis jerárquico realizado en el trabajo con el que nos comparamos, se realiza una comparación con el mejor de las 5 familias de GBML y los 6 clásicos.

Los resultados en ranking por cada base de datos así como los rankings globales para cada uno de los métodos se muestran en la siguiente tabla.

#BDs	Rankings											
33	GBML					Classical methods						CTC Tam=Max
Data-set	UCS	SIA	OCEC	Gassist	Oblique-DT	CART	AQ	CN2	C45	C45-Rules	Ripper	N_S=200
Abalone19	9,00	9,00	1,00	9,00	5,00	9,00	2,00	9,00	9,00	9,00	4,00	3,00
Yeast6	3,00	10,00	9,00	11,50	6,00	5,00	8,00	11,50	7,00	4,00	2,00	1,00
Yeast5	6,00	8,00	7,00	9,00	5,00	10,00	11,00	12,00	2,00	4,00	3,00	1,00
Yeast4	3,00	9,00	4,00	11,00	5,00	12,00	8,00	10,00	7,00	6,00	2,00	1,00
Yeast2vs8	7,00	11,50	3,00	5,00	4,00	9,00	8,00	6,00	11,50	10,00	2,00	1,00
Glass5	10,00	9,00	4,00	11,00	2,00	1,00	8,00	12,00	5,50	5,50	7,00	3,00
Abalone9vs18	7,00	9,00	4,00	8,00	2,00	10,00	12,00	11,00	6,00	5,00	3,00	1,00
Glass4	12,00	1,00	6,00	8,00	7,00	4,00	5,00	9,00	11,00	10,00	2,00	3,00
Ecoli4	9,00	10,00	8,00	6,00	2,00	7,00	12,00	11,00	5,00	3,00	1,00	4,00
Glass2	10,00	9,00	1,00	11,50	7,00	8,00	2,00	11,50	3,50	3,50	6,00	5,00
Vowel0	4,00	1,00	10,00	8,00	6,00	9,00	11,00	12,00	2,50	2,50	5,00	7,00
Page-blocks0	6,00	5,00	9,00	7,00	4,00	10,00	12,00	11,00	3,00	2,00	1,00	8,00
Ecoli3	11,00	9,00	8,00	7,00	6,00	4,00	12,00	10,00	5,00	3,00	1,00	2,00
Yeast3	3,00	9,00	2,00	6,00	8,00	11,00	10,00	12,00	7,00	4,00	1,00	5,00
Glass6	3,00	1,00	2,00	6,00	8,00	10,00	4,00	9,00	11,00	12,00	7,00	5,00
Segment0	2,00	1,00	10,00	9,00	6,00	5,00	11,00	12,00	8,00	7,00	3,00	4,00
Ecoli2	12,00	2,00	10,00	1,00	3,00	8,00	9,00	11,00	7,00	5,00	4,00	6,00
New-thyroid1	9,00	6,00	12,00	1,00	2,00	4,00	10,00	11,00	8,00	7,00	5,00	3,00
New-thyroid2	9,00	11,00	4,00	1,00	5,00	10,00	7,00	12,00	3,00	6,00	8,00	2,00
Ecoli1	10,00	8,00	9,00	3,00	7,00	6,00	12,00	11,00	5,00	4,00	1,00	2,00
Vehicle0	8,00	9,00	10,00	6,00	7,00	4,00	11,00	12,00	1,00	2,00	5,00	3,00
Glass0123vs456	7,00	8,00	9,00	5,00	6,00	2,00	12,00	11,00	1,00	3,00	4,00	10,00
Haberman	5,00	3,00	7,00	6,00	1,00	10,00	12,00	11,00	8,00	9,00	2,00	4,00
Vehicle1	4,00	7,00	6,00	9,00	2,00	10,00	11,00	12,00	3,00	5,00	1,00	8,00
Vehicle2	7,00	10,00	9,00	3,00	4,00	6,00	8,00	12,00	1,00	5,00	2,00	11,00
Vehicle3	2,00	9,00	5,00	7,00	4,00	10,00	11,00	12,00	6,00	3,00	1,00	8,00
Yeast1	2,00	8,00	10,00	6,00	7,00	9,00	11,00	12,00	5,00	1,00	3,00	4,00
Glass0	11,00	5,00	9,00	2,00	8,00	7,00	10,00	12,00	1,00	4,00	6,00	3,00
Iris0	5,00	1,50	12,00	3,00	8,50	1,50	10,00	11,00	5,00	5,00	8,50	7,00
Pima	3,00	9,00	10,00	7,00	6,00	1,00	12,00	11,00	5,00	4,00	2,00	8,00
Ecoli0vs1	12,00	5,00	10,00	4,00	6,00	7,00	11,00	9,00	1,50	1,50	8,00	3,00
Wisconsin	3,00	1,00	4,00	5,00	9,00	10,00	12,00	6,00	8,00	7,00	2,00	11,00
Glass1	9,00	1,00	10,00	6,00	8,00	3,00	12,00	11,00	4,00	7,00	2,00	5,00
Mean	6,76	6,52	7,09	6,30	5,35	7,05	9,61	10,79	5,35	5,12	3,47	4,61
Median	7,00	8,00	8,00	6,00	6,00	8,00	11,00	11,00	5,00	5,00	3,00	4,00
Ranking	8	7	10	6	5	9	11	12	5	3	1	2

Tabla 10 Los 11 algoritmos seleccionados del estudio en ranking con el CTC en su versión de 200 submuestras y TamMax

Como se desprende de la tabla, el algoritmo CTC es el segundo mejor algoritmo después del Ripper y seguido de los que tiene su base en el C4.5, el C4.5 propiamente dicho y el C4.5 Rules. Llama la atención que es el que mejores resultados obtiene en 5 de las 7 bases de datos más desbalanceadas, en las que los casos minoritarios no superan el 6 % del total.

#BDs	Rankings											
33	GBML					Classical methods						CTC Tam=M
Data-set	UCS	SIA	OCEC	Gassist	Oblique-DT	CART	AQ	CN2	C45	C45-Rules	Ripper	N_S 200%
Abalone19	9,00	9,00	1,00	9,00	5,00	9,00	2,00	9,00	9,00	9,00	4,00	3,00
Yeast6	3,00	10,00	9,00	11,50	6,00	5,00	8,00	11,50	7,00	4,00	1,00	2,00
Yeast5	6,00	8,00	7,00	9,00	5,00	10,00	11,00	12,00	2,00	4,00	3,00	1,00
Yeast4	3,00	9,00	4,00	11,00	5,00	12,00	8,00	10,00	7,00	6,00	2,00	1,00
Yeast2vs8	7,00	11,50	3,00	5,00	4,00	9,00	8,00	6,00	11,50	10,00	2,00	1,00
Glass5	10,00	9,00	4,00	11,00	2,00	1,00	8,00	12,00	5,50	5,50	7,00	3,00
Abalone9vs18	7,00	9,00	4,00	8,00	2,00	10,00	12,00	11,00	6,00	5,00	3,00	1,00
Glass4	12,00	1,00	5,00	8,00	7,00	3,00	4,00	9,00	11,00	10,00	2,00	6,00
Ecoli4	9,00	10,00	8,00	6,00	2,00	7,00	12,00	11,00	4,00	3,00	1,00	5,00
Glass2	10,00	9,00	1,00	11,50	7,00	8,00	2,00	11,50	4,50	4,50	6,00	3,00
Vowel0	4,00	1,00	10,00	8,00	6,00	9,00	11,00	12,00	2,50	2,50	5,00	7,00
Page-blocks0	6,00	5,00	9,00	7,00	4,00	10,00	12,00	11,00	3,00	2,00	1,00	8,00
Ecoli3	11,00	9,00	8,00	7,00	6,00	4,00	12,00	10,00	5,00	3,00	1,00	2,00
Yeast3	3,00	9,00	2,00	6,00	8,00	11,00	10,00	12,00	7,00	4,00	1,00	5,00
Glass6	3,00	1,00	2,00	6,00	8,00	10,00	4,00	9,00	11,00	12,00	7,00	5,00
Segment0	2,00	1,00	10,00	9,00	6,00	5,00	11,00	12,00	8,00	7,00	3,00	4,00
Ecoli2	12,00	3,00	10,00	1,00	4,00	8,00	9,00	11,00	7,00	6,00	5,00	2,00
New-thyroid1	9,00	6,00	12,00	1,00	2,00	4,00	10,00	11,00	8,00	7,00	5,00	3,00
New-thyroid2	9,00	11,00	4,00	1,00	5,00	10,00	7,00	12,00	3,00	6,00	8,00	2,00
Ecoli1	10,00	8,00	9,00	2,00	7,00	6,00	12,00	11,00	5,00	3,00	1,00	4,00
Vehicle0	8,00	9,00	10,00	6,00	7,00	3,00	11,00	12,00	1,00	2,00	5,00	4,00
Glass0123vs456	7,00	8,00	10,00	5,00	6,00	2,00	12,00	11,00	1,00	3,00	4,00	9,00
Haberman	5,00	3,00	7,00	6,00	1,00	10,00	12,00	11,00	8,00	9,00	2,00	4,00
Vehicle1	4,00	7,00	6,00	8,00	2,00	10,00	11,00	12,00	3,00	5,00	1,00	9,00
Vehicle2	7,00	10,00	9,00	3,00	4,00	6,00	8,00	12,00	1,00	5,00	2,00	11,00
Vehicle3	2,00	9,00	5,00	7,00	4,00	10,00	11,00	12,00	6,00	3,00	1,00	8,00
Yeast1	2,00	8,00	10,00	6,00	7,00	9,00	11,00	12,00	5,00	1,00	3,00	4,00
Glass0	11,00	5,00	9,00	2,00	8,00	7,00	10,00	12,00	1,00	4,00	6,00	3,00
Iris0	5,00	1,50	12,00	3,00	8,50	1,50	10,00	11,00	5,00	5,00	8,50	7,00
Pima	3,00	9,00	10,00	8,00	7,00	1,00	12,00	11,00	5,00	4,00	2,00	6,00
Ecoli0vs1	12,00	5,00	10,00	4,00	6,00	7,00	11,00	9,00	1,50	1,50	8,00	3,00
Wisconsin	3,00	1,00	4,00	5,00	9,00	10,00	12,00	6,00	8,00	7,00	2,00	11,00
Glass1	9,00	1,00	10,00	5,00	7,00	3,00	12,00	11,00	4,00	6,00	2,00	8,00
Mean	6,76	6,55	7,09	6,24	5,38	6,98	9,58	10,79	5,35	5,12	3,47	4,70
Median	7,00	8,00	8,00	6,00	6,00	8,00	11,00	11,00	5,00	5,00	3,00	4,00
Ranking	8	7	10	6	5	9	11	12	4	3	1	2

Tabla 11 Los 11 algoritmos seleccionados del estudio en ranking con el CTC en su versión de número submuestras al 200% y TamMax

Exactamente lo mismo sucede con el CTC pero en la versión basada en submuestras TamMax y el número de submuestras al 200%. Sigue obteniendo resultados competentes teniendo como hándicap un coste computacional inferior que depende en qué condiciones puede resultar un punto de valor.

Buscando diferencias estadísticamente significativas mediante el test de Ivan Davenport, el p value es cercano al cero por lo que se decide proceder con un test más potente como es el test de Shaffer.

Comparando estos resultados con los obtenidos en el estudio de Fernández et al, las conclusiones no cambian representativamente ya que el Ripper sigue siendo el mejor algoritmo de todos, con diferencias estadísticamente significativas sobre UCS, SIA, OCEC y GASSIST pero no se encuentran diferencias significativas con los algoritmos que poseen su base en los árboles de clasificación como son los algoritmos Oblique DT, C4.5 y C4.5 rules así como el CTC.

Capítulo 7

Experimentación 2

Análisis del comportamiento del CTC en un contexto de [33 BDS] preprocesadas con SMOTE y en comparación con 22 algoritmos de inducción de reglas [(16 evolutivos y 6 clásicos)]

En este capítulo se muestra el comportamiento del algoritmo de construcción de árboles consolidados, CTC, para las 33 bases de datos, habiéndoles aplicado anteriormente la técnica de preprocesamiento SMOTE y una comparación con los resultados publicados en el artículo de Fernández et al en [FGLBH10].

7.1 Introducción al experimento [2]

Como sucede en la experimentación anterior se han empleado las mismas 33 bases de datos pero esta vez aplicándolas técnicas de preprocesamiento de datos con el fin de balancear la distribución de clases de la misma, en concreto SMOTE. Pueden ser descargadas del enlace http://sci2s.ugr.es/keel/index.php. El estudio lo conforman las siguientes bases de datos: abalone9-18, abalone19, ecoli-0_vs_1, ecoli1, ecoli2, ecoli3, ecoli4, glass0, glass-0-1-2-3_vs_4-5-6, glass1, glass2, glass4, glass5, glass6, haberman, iris0, new-thyroid1, new-thyroid2, page-blocks0, pima, segment0, vehicle0, vehicle1, vehicle2, vehicle3, vowel0, Wisconsin, yeast1, yeast-2_vs_8, yeast3, yeast4, yeast5, yeast6.

Las técnicas empleadas para esta manipulación de los datos son las expuestas en la parte referente a la base teórica, en el capítulo 4 y más concretamente la detallada en la sección referente a oversampling.

7.2 Metodología experimental

Respecto a la metodología empleada para la validación se ha empleado la misma que en la primera experimentación, es decir, realizar 5 validaciones cruzadas de 5 subconjuntos o *folds*. Para cada subconjunto de la validación cruzada se ha aplicado la misma técnica de remuestreo generando un número de submuestras que fijaremos en 5 y con una distribución de clases al 50%, para mantener la misma que en el experimento anterior.

En este caso, se ha decidido emplear 3 tamaños de submuestras distintos que se especifican a continuación.

El primer tamaño se ha establecido al número de elementos pertenecientes a la clase minoritaria del conjunto de entrenamiento (*imbalanced*), como realizaron Weiss y Provost en [WP03]. A partir de ahora haremos referencia a este tamaño mediante TamSizeOfMinClass, igual que en el experimento 1.

El segundo tamaño viene dado por el número de elementos pertenecientes a la clase mayoritaria del conjunto de entrenamiento. En este experimento coincide con la minoritaria porque ambas clases están balanceadas con SMOTE. La nomenclatura empleada será TamSizeOfMajClass para referirnos a este tipo de muestras.

El tercer tamaño de submuesta que se empleará es TamMax (*imbalanced*), empleado en el experimento anterior.

En lo referente al número de muestras se ha empleado un número de muestras fijo, 5 para ser exactos. Puede sorprender a priori que no se haya variado el número de submuestras como en el experimento 1, la razón es que es un análisis preliminar por considerar que con el experimento anterior ya se han cumplido los objetivos del proyecto.

#BDs 33			
Data-set	SizeOf MajClass	TamMax	TamSize OfMinClass
Abalone19	3314	52	26
Yeast6	1158	60	30
Yeast5	1153	71	36
Yeast4	1147	82	41
Yeast2vs8	370	33	17
Glass5	165	15	8
Abalone9vs18	552	67	34
Glass4	161	21	11
Ecoli4	251	37	19
Glass2	157	31	16
Vowel0	720	143	72
Page-blocks0	3930	896	448
Ecoli3	240	59	30
Yeast3	1057	261	131
Glass6	149	47	24
Segment0	1584	527	264
Ecoli2	228	84	42
New-thyroid1	144	57	29
New-thyroid2	143	59	30
Ecoli1	208	124	62
Vehicle0	517	320	160
Glass0123vs456	131	82	41
Haberman	178	135	68
Vehicle1	485	385	193
Vehicle2	485	385	193
Vehicle3	485	385	193
Yeast1	844	687	344
Glass0	116	112	56
Iris0	81	80	40
Pima	401	429	215
Ecoli0vs1	115	124	62
Wisconsin	356	383	192
Glass1	111	122	61
Mean	640,48	192,58	96,61
Median	356,00	84,00	42,00

Tabla 12 Tamaño de submuestras empleadas para cada una de las bases de datos SMOTE

En la siguiente tabla se encuentran detallados los tamaños que toman las submuestras de cada base de datos del estudio para las tres variantes.

Para cada base de datos y cada uno de los tamaños de la submuestras (TamMax, TamSizeOfMajClass y TamSizeOfMinClass) se han construido 25 árboles de clasificación consolidados (5 x *5 cross validation*), lo que genera un total de 75 árboles CT.

Como disponemos de 33 bases de datos para el estudio al final se habrán generado 2475 árboles CT.

Como sucede con el experimento de las bases de datos *imbalanced*, en el primer anexo al finalizar las tablas del mismo, se encuentran los resultados obtenidos para las bases de datos en la modalidad SMOTE.

En la ilustración 63, se muestra el proceso llevado a cabo para generar las submuestras a partir de la muestra original.

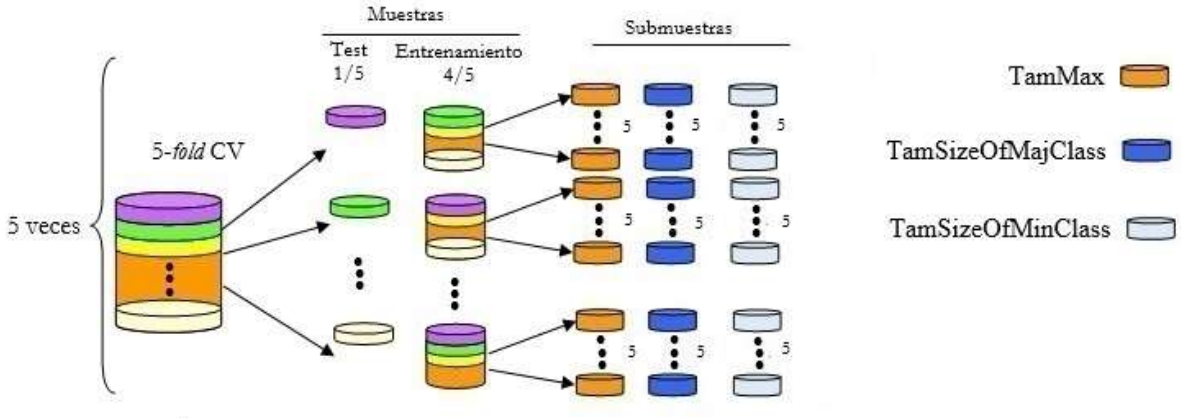


Ilustración 63 Proceso llevado a cabo para la generación de submuestras en las bases de datos SMOTE

Una vez finalizada la exposición de la metodología experimental empleada, se encuentran los resultados obtenidos en el artículo *Genetic based machine learning for rule induction* en la tabla inferior, para las bases de datos en función del GM y a su vez el ranking de los 11 algoritmos, los 5 mejores de cada una de las familias GBML y los 6 clásicos, a destacar, que esta vez los representantes de GBML no son los mismos que en el experimento 1.

#BDs				Results GM Test										
33				GBML					Classical methods					
Data-set	Tam	%Min	SizeOf MinClass	XCS	SIA	CORE	Gassist	DT-GA	CART	AQ	CN2	C45	C45-Rules	Ripper
Abalone19	4174	0,77	32,14	67,08	23,35	67,01	48,18	15,58	0,00	18,26	55,41	8,00	7,99	38,80
Yeast6	1484	2,49	36,95	87,04	83,10	77,62	80,81	80,57	68,03	42,71	60,22	84,18	80,31	73,25
Yeast5	1484	2,96	43,93	96,40	94,79	93,70	95,58	90,93	71,45	34,16	68,16	95,06	95,07	84,71
Yeast4	1484	3,43	50,90	80,56	72,36	80,57	76,72	66,66	10,84	28,71	22,59	73,39	75,83	65,67
Yeast2vs8	482	4,15	20,00	68,61	70,61	72,11	76,77	81,01	50,86	41,69	69,44	82,33	74,18	58,75
Glass5	214	4,2	8,99	91,98	91,39	86,63	91,47	86,78	93,90	90,61	71,89	91,40	91,14	77,77
Abalone9vs18	731	5,65	41,30	69,27	69,18	69,45	70,24	52,79	25,14	7,08	49,71	63,19	66,46	73,80
Glass4	214	6,07	12,99	83,20	87,44	60,98	68,92	87,50	76,37	71,09	76,96	72,13	68,00	68,94
Ecoli4	336	6,74	22,65	90,65	90,70	90,56	86,35	74,55	77,88	55,61	74,59	86,48	84,28	82,39
Glass2	214	8,78	18,79	64,09	68,23	60,78	54,69	51,36	25,08	62,66	65,12	64,04	68,04	62,33
Vowel0	988	9,01	89,02	99,44	99,94	79,89	96,45	93,72	84,68	55,36	49,86	94,69	96,59	97,47
Page-blocks0	5472	10,23	559,79	94,46	92,24	73,41	88,87	94,86	71,04	0,64	73,43	94,51	94,55	94,36
Ecoli3	336	10,88	36,56	83,79	84,43	82,86	83,08	81,43	68,12	39,98	19,99	85,78	81,83	87,16
Yeast3	1484	10,98	162,94	91,84	87,16	86,84	92,03	88,77	80,82	75,14	76,40	92,26	93,55	88,26
Glass6	214	13,55	29,00	86,88	87,11	88,46	89,46	89,02	80,89	60,39	86,59	92,62	89,58	94,95
Segment0	2308	14,26	329,12	98,62	99,19	73,04	98,98	99,11	98,60	91,96	78,02	99,54	98,91	99,14
Ecoli2	336	15,48	52,01	89,29	91,00	87,39	89,77	87,25	81,24	51,91	49,77	89,08	88,64	84,06
New-thyroid1	215	16,28	35,00	97,41	98,30	93,46	94,84	96,06	94,62	83,25	85,35	93,39	94,85	94,34
New-thyroid2	215	16,89	36,31	96,57	98,60	89,60	95,72	96,49	91,18	83,27	86,12	95,93	97,70	97,11
Ecoli1	336	22,92	77,01	89,04	84,38	89,44	85,25	90,01	82,91	40,79	42,14	88,26	88,26	85,09
Vehicle0	846	23,64	199,99	93,99	84,52	75,47	94,42	90,86	92,28	75,29	76,30	92,62	92,25	93,52
Glass0123vs456	214	23,83	51,00	89,48	93,48	86,03	93,13	92,35	91,01	55,14	80,57	90,05	87,11	85,53
Haberman	306	27,42	83,91	56,47	50,86	65,93	60,69	61,59	47,07	2,98	39,58	64,03	66,90	34,78
Vehicle1	846	28,37	240,01	74,04	64,71	62,71	76,28	67,85	53,63	41,24	52,38	71,90	72,13	66,52
Vehicle2	846	28,37	240,01	96,88	77,41	39,73	95,24	94,29	93,52	71,49	68,12	95,79	95,46	96,70
Vehicle3	846	28,37	240,01	74,18	66,66	19,55	73,67	66,85	50,07	35,35	56,74	72,62	69,18	64,85
Yeast1	1484	28,91	429,02	69,74	63,95	65,85	72,73	70,76	53,97	24,93	27,93	72,23	69,94	67,92
Glass0	214	32,71	70,00	81,30	83,92	69,70	84,27	77,90	74,60	58,84	72,23	79,02	75,76	79,52
Iris0	150	33,33	50,00	100,00	99,49	100,00	100,00	98,97	100,00	95,34	94,34	98,97	98,97	97,89
Pima	768	34,84	267,57	70,27	67,08	70,36	73,98	70,97	75,96	8,78	47,90	72,92	70,47	69,44
Ecoli0vs1	220	35	77,00	97,63	95,15	97,97	97,94	98,31	96,86	86,65	86,65	97,24	97,58	95,13
Wisconsin	683	35	239,05	96,73	96,05	94,72	95,10	94,80	92,23	94,55	93,98	96,19	95,58	94,41
Glass1	214	35,51	75,99	75,53	82,33	62,43	80,07	70,03	73,36	47,25	40,77	70,49	72,04	68,37
Mean	**919,94**	**17,61**	**119,97**	**84,92**	**81,79**	**76,19**	**83,69**	**80,61**	**70,55**	**52,52**	**63,61**	**82,43**	**81,79**	**79,48**
Median	482,00	15,48	52,01	89,04	84,52	77,62	86,35	87,25	76,37	55,14	68,16	88,26	87,11	84,06

Tabla 13 Resultado en función de GM para los 33 imbalanced datasets y los 11 algoritmos seleccionados del artículo

Observando la media a primera vista se observa que los resultados obtenidos para las bases de datos SMOTE son diferentes a las bases de datos *imbalanced*, ya que en rasgos generales los basados en GBML obtienen mejores resultados en global que los métodos clásicos.

Si nos centramos únicamente en los 5 mejores, 3 pertenecen a los denominados GMBL. Con el XCS y Gassist siendo los dos mejores, seguidamente se sitúa el C4.5 (que es el mejor colocado de los clásicos) y después SIA y C4.5 Rules con l con los mismos resultados.

	Results GM Test										
	GBML					Classical methods					
	XCS	SIA	CORE	Gassist	DT-GA	CART	AQ	CN2	C45	C45-Rules	Ripper
Mean	3,41	4,55	6,74	3,80	5,33	7,74	10,02	9,20	4,23	4,77	6,21
Median	3,00	5,00	7,00	3,00	5,00	9,00	11,00	10,00	4,00	4,50	7,00
Ranking	**1**	**4**	**8**	**2**	**6**	**9**	**11**	**10**	**3**	**5**	**7**

Tabla 14 Rankinsg de los 11 algoritmos seleccionados (5 GBLM + 6 classical)

Si utilizamos como parámetro de medición el ranking en vez de la media, se corroboran los mismos resultados ya que en el caso de los 5 mejores algoritmos mantiene el orden.

7.3 Resultados

Se ha realizado una experimentación menos robusta que la anterior con el fin de comparar el desempeño del CTC con los resultados de los 22 algoritmos (6 métodos clásicos y 12 GBML) y obtener una primera idea del comportamiento del CTC para las bases de datos SMOTE. Concretamente se desea estudiar el efecto del tamaño de las submuestras en los resultados dejando a un lado el número de submuestras óptimos para futuros trabajos. Por lo tanto se han evaluado estos 23 algoritmos sobre las 33 bases de datos SMOTE.

Esta comparación, como la realizada en el capítulo anterior, únicamente es desde el punto de vista de la capacidad de aprendizaje dejando al lado otros posibles puntos de vista.

Igualmente para medir la capacidad de aprendizaje se ha empleado el valor de la media geométrica o GM.

Se ha empleado la misma metodología jerárquica en la comparación empleada hasta ahora. Se comienza con el análisis de las 3 diferentes estrategias empleadas para construir los árboles consolidados -variando el tamaño de la submuestra- , para después seleccionar entre ellas la que mejor se comporta.

En la Tabla 15 se encuentran los resultados obtenidos para todas las bases de datos que componen el estudio y los 3 tamaños de submuestras expuestos en la metodología experimental.

En la parte derecha de la misma tabla se muestra el ranking medio de las 3 variantes del CTC probadas contra los resultados obtenidos por el C4.5, su principal competidor.

Si nos fijamos en los resultados arrojados por el ranking dos de las tres configuraciones probadas del CTC obtienen mejor ranking medio que el C4.5. Los mejores resultados se han logrado empleado el tamaño TamMax. Por el contrario los peores han sido con tamaños de submuestras al TamSizeOfMinClass.

#BDs				Results GM Test				Rankings			
33				Classical r	Tam=SizeOfMajClass	Tam=SizeOfMinClass	Tam=Max	Classica	Tam=SizeOfMajClass	Tam=SizeOfMinClass	Tam=Max
Data-set	Tam	%Min	SizeOf MinClass	C45	N_S=5	N_S=5	N_S=5	C45	N_S=5	N_S=5	N_S=5
Abalone19	4174	0,77	32,14	8,00	9,46	58,81	61,07	4	3	2	1
Yeast6	1484	2,49	36,95	84,18	57,93	65,83	65,87	1	4	3	2
Yeast5	1484	2,96	43,93	95,06	97,08	97,59	97,32	4	3	1	2
Yeast4	1484	3,43	50,90	73,39	86,88	88,14	88,12	4	3	1	2
Yeast2vs8	482	4,15	20,00	82,33	86,28	85,33	85,10	4	1	2	3
Glass5	214	4,2	8,99	91,40	82,38	85,21	86,81	1	4	3	2
Abalone9vs18	731	5,65	41,30	63,19	84,16	87,76	83,91	4	2	1	3
Glass4	214	6,07	12,99	72,13	77,63	76,97	81,68	4	2	3	1
Ecoli4	336	6,74	22,65	86,48	89,88	88,59	88,94	4	1	3	2
Glass2	214	8,78	18,79	64,04	70,62	69,95	71,78	4	2	3	1
Vowel0	988	9,01	89,02	94,69	61,56	54,03	57,07	1	2	4	3
Page-blocks0	5472	10,23	559,79	94,51	81,11	79,60	83,59	1	3	4	2
Ecoli3	336	10,88	36,56	85,78	75,82	92,67	87,35	3	4	1	2
Yeast3	1484	10,98	162,94	92,26	87,44	89,15	88,97	1	4	2	3
Glass6	214	13,55	29,00	92,62	61,11	17,28	45,92	1	2	4	3
Segment0	2308	14,26	329,12	99,54	98,77	98,36	98,36	1	2	3,5	3,5
Ecoli2	336	15,48	52,01	89,08	96,89	91,85	93,96	4	1	3	2
New-thyroid1	215	16,28	35,00	93,39	94,69	92,66	93,57	3	1	4	2
New-thyroid2	215	16,89	36,31	95,93	88,99	87,17	90,01	1	3	4	2
Ecoli1	336	22,92	77,01	88,26	70,33	72,10	69,82	1	3	2	4
Vehicle0	846	23,64	199,99	92,62	99,10	98,31	98,77	4	1	3	2
Glass0123vs456	214	23,83	51,00	90,05	91,81	91,15	91,85	4	2	3	1
Haberman	306	27,42	83,91	64,03	21,27	2,81	19,56	1	2	4	3
Vehicle1	846	28,37	240,01	71,90	84,02	83,09	83,73	4	1	3	2
Vehicle2	846	28,37	240,01	95,79	19,73	2,58	23,63	1	3	4	2
Vehicle3	846	28,37	240,01	72,62	94,92	93,05	94,28	4	1	3	2
Yeast1	1484	28,91	429,02	72,23	94,19	93,65	94,30	4	2	3	1
Glass0	214	32,71	70,00	79,02	69,54	70,00	69,54	1	4	2	3
Iris0	150	33,33	50,00	98,97	78,99	72,87	75,46	1	2	4	3
Pima	768	34,84	267,57	72,92	89,86	92,31	91,99	4	3	1	2
Ecoli0vs1	220	35	77,00	97,24	66,18	82,44	79,02	1	4	2	3
Wisconsin	683	35	239,05	96,19	91,29	95,20	95,08	1	4	2	3
Glass1	214	35,51	75,99	70,49	75,25	87,53	83,68	4	3	1	2
Mean	**919,94**	**17,61**	**119,97**	**82,43**	**76,82**	**77,09**	**79,40**	**2,58**	**2,48**	**2,68**	**2,26**
Median	482,00	15,48	52,01	88,26	84,02	87,17	85,10	3,00	2,00	3,00	2,00
								3	2	4	1

Tabla 15 Resultado en función de GM para los 33 imbalanced datasets y las 3 distintas configuraciones del algoritmo CTC

Con el objetivo de detectar si existen diferencias significativas entre las distintas variantes en las que se ha probado el CTC se aplica el test de Ivan Davenport. Se muestran los resultados para el test de Nenemyi y Bonferroni-Dunn para detectar, en caso de que existan diferencias significativas, entre qué casos se dan.

De la ilustración 64 se desprende que en base a las líneas mostradas para el test de Nemenyi que no existen diferencias significativas entre las tres configuraciones del CTC. En la parte inferior de la imagen se muestra el valor de la distancia crítica, que en este caso es de 0,5768.

En cuanto al test de Bonferroni-Dunn tampoco existen diferencias significativas, ya que no encontramos ningún algoritmo fuera de la línea. Junto a la línea se encuentra la leyenda que muestra la distancia crítica, 0,5517.

.

TamMax	1,80
TamSizeOfMajClass	2,00
TamSizeOfMinClass	2,20

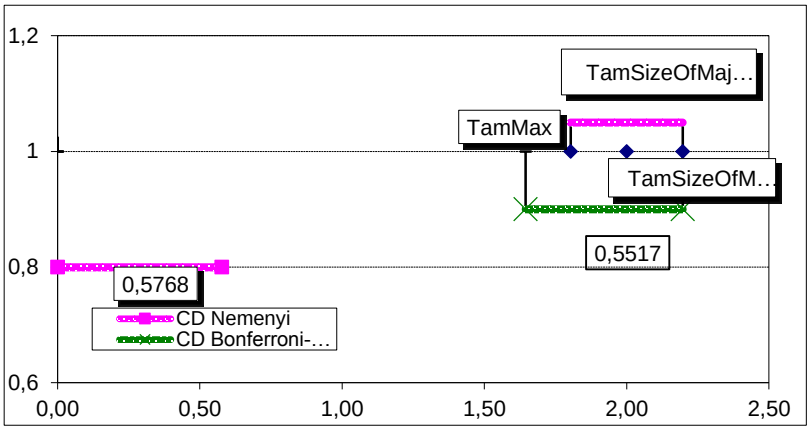


Ilustración 64 Diagrama CD para los test de Nemenyi y Bonferroni-Dunn para el algoritmo CTC en sus 3 tamaños de submuestra

#BDs	Results GM Test		Rankings	
33	Classical n	Tam=SizeOfMajClass	Classica	Tam=S
Data-set	C45	N_S=5	C45	N_S=5
Abalone19	8,00	61,07	2	1
Yeast6	84,18	65,87	1	2
Yeast5	95,06	97,32	2	1
Yeast4	73,39	88,12	2	1
Yeast2vs8	82,33	85,10	2	1
Glass5	91,40	86,81	1	2
Abalone9vs18	63,19	83,91	2	1
Glass4	72,13	81,68	2	1
Ecoli4	86,48	88,94	2	1
Glass2	64,04	71,78	2	1
Vowel0	94,69	57,07	1	2
Page-blocks0	94,51	83,59	1	2
Ecoli3	85,78	87,35	2	1
Yeast3	92,26	88,97	1	2
Glass6	92,62	45,92	1	2
Segment0	99,54	98,36	1	2
Ecoli2	89,08	93,96	2	1
New-thyroid1	93,39	93,57	2	1
New-thyroid2	95,93	90,01	1	2
Ecoli1	88,26	69,82	1	2
Vehicle0	92,62	98,77	2	1
Glass0123vs456	90,05	91,85	2	1
Haberman	64,03	19,56	1	2
Vehicle1	71,90	83,73	2	1
Vehicle2	95,79	23,63	1	2
Vehicle3	72,62	94,28	2	1
Yeast1	72,23	94,30	2	1
Glass0	79,02	69,54	1	2
Iris0	98,97	75,46	1	2
Pima	72,92	91,99	2	1
Ecoli0vs1	97,24	79,02	1	2
Wisconsin	96,19	95,08	1	2
Glass1	70,49	83,68	2	1
Mean	82,43	79,40	1,55	1,45
Median	88,26	85,10	2,00	1,00

Tabla 16 Resultados medios y de ranking para el CTC y TamMax

Si nos centramos en una comparación en rankings medios entre el C4.5 y la mejor estrategia probada del CTC, se ve que de las 19 bases de datos más desbalanceadas, el CTC gana en 11 de ellas, si nos centramos en los casos más extremos, como podían ser las primeras 10 bases de datos con un desbalanceo de datos al 9%, la diferencia es mayor ya que el CTC supera el C4.5 en 8 de ellas.

Como resultado de todo este análisis se ha elegido CTC con TamMax como la mejor y más representativa estrategia. Ahora es momento se comparar las actitudes de clasificación que posee esta configuración en el ámbito del desbalanceo de clases frente a los 22 algoritmos.

Basándonos en el mismo análisis jerárquico realizado en el trabajo con el que nos comparamos, se emplea el mismo método de búsqueda en cada una de las familias, así seleccionamos los 11 mejores algoritmos en función de los resultados del artículo de Fernández et al, con los que compararemos el CTC. El resultado de esta comparación se encuentra en la ilustración que se sitúa en la página siguiente.

Si nos basamos en la media como medida de rendimiento se aprecia que el CTC se encuentra próximo, sin llegar a alcanzar los resultados obtenidos por el C4.5 y el C4.5 rules pero que de los métodos propuestos bajo la denominación de GBML sólo es capaz de ganar al CORE.

Si por el contrario miramos los rankings a pesar de obtener peor media que la mayoría del resto de los clasificadores en ranking medio se sitúa en la zona media ya que obtiene la posición 6 de 12.

Es de esperar que aumentando el número de submuestras los resultados mejoren considerablemente como se ha demostrado en la primera experimentación para las bases de datos *imbalanced*.

Queda pendiente realizar pruebas más exhaustivas pero se escapan a los objetivos que busca este proyecto, que es tomar una primera toma de contacto con la finalidad de ver el efecto del tamaño de las submuestras en el rendimiento del CTC.

#BDs				Results GM Test												
33				GBML					Classical methods							Tam=5MOTR
Data-set	Tam	%Min	SizeOf MinClass	XCS	SIA	CORE	Gassist	DT-GA	CART	AQ	CN2	C45	C45-Rules	Ripper	N_S=5
Abalone19	4174	0,77	32,14	67,08	23,35	67,01	48,18	15,58	0,00	18,26	55,41	8,00	7,99	38,80	61,07
Yeast6	1484	2,49	36,95	87,04	83,10	77,62	80,81	80,57	68,03	42,71	60,22	84,18	80,31	73,25	65,87
Yeast5	1484	2,96	43,93	96,40	94,79	93,70	95,58	90,93	71,45	34,16	68,16	95,06	95,07	84,71	97,32
Yeast4	1484	3,43	50,90	80,56	72,36	80,57	76,72	66,66	10,84	28,71	22,59	73,39	75,83	65,67	88,12
Yeast2vs8	482	4,15	20,00	68,61	70,61	72,11	76,77	81,01	50,86	41,69	69,44	82,33	74,18	58,75	85,10
Glass5	214	4,2	8,99	91,98	91,39	86,63	91,47	86,78	93,90	90,61	71,89	91,40	91,14	77,77	86,81
Abalone9vs18	731	5,65	41,30	69,27	69,18	69,45	70,24	52,79	25,14	7,08	49,71	63,19	66,46	73,80	83,91
Glass4	214	6,07	12,99	83,20	87,44	60,98	68,92	87,50	76,37	71,09	76,96	72,13	68,00	68,94	81,68
Ecoli4	336	6,74	22,65	90,65	90,70	90,56	86,35	74,55	77,88	55,61	74,59	86,48	84,28	82,39	88,94
Glass2	214	8,78	18,79	64,09	68,23	60,78	54,69	51,36	25,08	62,66	65,12	64,04	68,04	62,33	71,78
Vowel0	988	9,01	89,02	99,44	99,94	79,89	96,45	93,72	84,68	55,36	49,86	94,69	96,59	97,47	57,07
Page-blocks0	5472	10,23	559,79	94,46	92,24	73,41	88,87	94,86	71,04	0,64	73,43	94,51	94,55	94,36	83,59
Ecoli3	336	10,88	36,56	83,79	84,43	82,86	83,08	81,43	68,12	39,98	19,99	85,78	81,83	87,16	87,35
Yeast3	1484	10,98	162,94	91,84	87,16	86,84	92,03	88,77	80,82	75,14	76,40	92,26	93,55	88,26	88,97
Glass6	214	13,55	29,00	86,88	87,11	88,46	89,46	89,02	80,89	60,39	86,59	92,62	89,58	94,95	45,92
Segment0	2308	14,26	329,12	98,62	99,19	73,04	98,98	99,11	98,60	91,96	78,02	99,54	98,91	99,14	98,36
Ecoli2	336	15,48	52,01	89,29	91,00	87,39	89,77	87,25	81,24	51,91	49,77	89,08	88,64	84,06	93,96
New-thyroid1	215	16,28	35,00	97,41	98,30	93,46	94,84	96,06	94,62	83,25	85,35	93,39	94,85	94,34	93,57
New-thyroid2	215	16,89	36,31	96,57	98,60	89,60	95,72	96,49	91,18	83,27	86,12	95,93	97,70	97,11	90,01
Ecoli1	336	22,92	77,01	89,04	84,38	89,44	85,25	90,01	82,91	40,79	42,14	88,26	88,26	85,09	69,82
Vehicle0	846	23,64	199,99	93,99	84,52	75,47	94,42	90,86	92,28	75,29	76,30	92,62	92,25	93,52	98,77
Glass0123vs456	214	23,83	51,00	89,48	93,48	86,03	93,13	92,35	91,01	55,14	80,57	90,05	87,11	85,53	91,85
Haberman	306	27,42	83,91	56,47	50,86	65,93	60,69	61,59	47,07	2,98	39,58	64,03	66,90	34,78	19,56
Vehicle1	846	28,37	240,01	74,04	64,71	62,71	76,28	67,85	53,63	41,24	52,38	71,90	72,13	66,52	83,73
Vehicle2	846	28,37	240,01	96,88	77,41	39,73	95,24	94,29	93,52	71,49	68,12	95,79	95,46	96,70	23,63
Vehicle3	846	28,37	240,01	74,18	66,66	19,55	73,67	66,85	50,07	35,35	56,74	72,62	69,18	64,85	94,28
Yeast1	1484	28,91	429,02	69,74	63,95	65,85	72,73	70,76	53,97	24,93	27,93	72,23	69,94	67,92	94,30
Glass0	214	32,71	70,00	81,30	83,92	69,70	84,27	77,90	74,60	58,84	72,23	79,02	75,76	79,52	69,54
Iris0	150	33,33	50,00	100,00	99,49	100,00	100,00	98,97	100,00	95,34	94,34	98,97	98,97	97,89	75,46
Pima	768	34,84	267,57	70,27	67,08	70,36	73,98	70,97	75,96	8,78	47,90	72,92	70,47	69,44	91,99
Ecoli0vs1	220	35	77,00	97,63	95,15	97,97	97,94	98,31	96,86	86,65	86,65	97,24	97,58	95,13	79,02
Wisconsin	683	35	239,05	96,73	96,05	94,72	95,10	94,80	92,23	94,55	93,98	96,19	95,58	94,41	95,08
Glass1	214	35,51	75,99	75,53	82,33	62,43	80,07	70,03	73,36	47,25	40,77	70,49	72,04	68,37	83,68
Mean	**919,94**	**17,61**	**119,97**	**84,92**	**81,79**	**76,19**	**83,69**	**80,61**	**70,55**	**52,52**	**63,61**	**82,43**	**81,79**	**79,48**	**79,40**
Median	482,00	15,48	52,01	89,04	84,52	77,62	86,35	87,25	76,37	55,14	68,16	88,26	87,11	84,06	85,10
				1	**4**	**9**	**2**	**7**	**10**	**12**	**11**	**3**	**5**	**8**	**6**

Tabla 17 Resultado en función de GM para los 33 imbalanced datasets y el CTC al tamaño de submuestras TamMax

Para finalizar esta experimentación se muestran los resultados de ranking concreto para cada una de las bases de datos.

De la última tabla que pone fin a este capítulo destaca que es el clasificador que más veces queda en primer lugar obteniendo esta posición en el ranking 13 veces por 5 veces que quedan primero XCS y SIA, por lo que si luego queda peor es porque, en media, queda mucho peor.

#BDs	Rankings											
33	GBML					Classical methods						Tam=5 SMOTE-M
Data-set	XCS	SIA	CORE	Gassist	DT-GA	CART	AQ	CN2	C45	C45-Rules	Ripper	N_S=5
Abalone19	1,00	7,00	2,00	5,00	9,00	12,00	8,00	4,00	10,00	11,00	6,00	3,00
Yeast6	1,00	3,00	7,00	4,00	5,00	9,00	12,00	11,00	2,00	6,00	8,00	10,00
Yeast5	2,00	6,00	7,00	3,00	8,00	10,00	12,00	11,00	5,00	4,00	9,00	1,00
Yeast4	3,00	7,00	2,00	4,00	8,00	12,00	10,00	11,00	6,00	5,00	9,00	1,00
Yeast2vs8	9,00	7,00	6,00	4,00	3,00	11,00	12,00	8,00	2,00	5,00	10,00	1,00
Glass5	2,00	5,00	10,00	3,00	9,00	1,00	7,00	12,00	4,00	6,00	11,00	8,00
Abalone9vs18	5,00	6,00	4,00	3,00	9,00	11,00	12,00	10,00	8,00	7,00	2,00	1,00
Glass4	3,00	2,00	12,00	10,00	1,00	6,00	8,00	5,00	7,00	11,00	9,00	4,00
Ecoli4	2,00	1,00	3,00	6,00	11,00	9,00	12,00	10,00	5,00	7,00	8,00	4,00
Glass2	5,00	2,00	9,00	10,00	11,00	12,00	7,00	4,00	6,00	3,00	8,00	1,00
Vowel0	2,00	1,00	9,00	5,00	7,00	8,00	11,00	12,00	6,00	4,00	3,00	10,00
Page-blocks0	4,00	6,00	10,00	7,00	1,00	11,00	12,00	9,00	3,00	2,00	5,00	8,00
Ecoli3	5,00	4,00	7,00	6,00	9,00	10,00	11,00	12,00	3,00	8,00	2,00	1,00
Yeast3	4,00	8,00	9,00	3,00	6,00	10,00	12,00	11,00	2,00	1,00	7,00	5,00
Glass6	8,00	7,00	6,00	4,00	5,00	10,00	11,00	9,00	2,00	3,00	1,00	12,00
Segment0	7,00	2,00	12,00	5,00	4,00	8,00	10,00	11,00	1,00	6,00	3,00	9,00
Ecoli2	4,00	2,00	7,00	3,00	8,00	10,00	11,00	12,00	5,00	6,00	9,00	1,00
New-thyroid1	2,00	1,00	9,00	5,00	3,00	6,00	12,00	11,00	10,00	4,00	7,00	8,00
New-thyroid2	4,00	1,00	10,00	7,00	5,00	8,00	12,00	11,00	6,00	2,00	3,00	9,00
Ecoli1	3,00	8,00	2,00	6,00	1,00	9,00	12,00	11,00	4,50	4,50	7,00	10,00
Vehicle0	3,00	9,00	11,00	2,00	8,00	6,00	12,00	10,00	5,00	7,00	4,00	1,00
Glass0123vs456	7,00	1,00	9,00	2,00	3,00	5,00	12,00	11,00	6,00	8,00	10,00	4,00
Haberman	6,00	7,00	2,00	5,00	4,00	8,00	12,00	9,00	3,00	1,00	10,00	11,00
Vehicle1	3,00	8,00	9,00	2,00	6,00	10,00	12,00	11,00	5,00	4,00	7,00	1,00
Vehicle2	1,00	8,00	11,00	5,00	6,00	7,00	9,00	10,00	3,00	4,00	2,00	12,00
Vehicle3	2,00	7,00	12,00	3,00	6,00	10,00	11,00	9,00	4,00	5,00	8,00	1,00
Yeast1	6,00	9,00	8,00	2,00	4,00	10,00	12,00	11,00	3,00	5,00	7,00	1,00
Glass0	3,00	2,00	10,00	1,00	6,00	8,00	12,00	9,00	5,00	7,00	4,00	11,00
Iris0	2,50	5,00	2,50	2,50	7,00	2,50	10,00	11,00	7,00	7,00	9,00	12,00
Pima	8,00	10,00	7,00	3,00	5,00	2,00	12,00	11,00	4,00	6,00	9,00	1,00
Ecoli0vs1	4,00	8,00	2,00	3,00	1,00	7,00	10,50	10,50	6,00	5,00	9,00	12,00
Wisconsin	1,00	3,00	8,00	5,00	7,00	12,00	9,00	11,00	2,00	4,00	10,00	6,00
Glass1	4,00	2,00	10,00	3,00	8,00	5,00	11,00	12,00	7,00	6,00	9,00	1,00
Mean	3,83	5,00	7,41	4,29	5,88	8,35	10,86	10,02	4,77	5,29	6,82	5,48
Median	3,00	6,00	8,00	4,00	6,00	9,00	12,00	11,00	5,00	5,00	8,00	4,00
Ranking	1	4	9	2	7	10	12	11	3	5	8	6

Tabla 18 Los 11 algoritmos seleccionados del estudio en ranking con el CTC en su versión con submuestras al tamaño TamMax

Capítulo 8

Conclusiones

Finalmente en este último capítulo se recogen las principales conclusiones obtenidas en las experimentaciones llevadas a cabo en este proyecto y se apuntan algunas líneas de trabajo que han quedado abiertas.

8.1 Resultados y discusión

En este trabajo se ha analizado el desempeño o rendimiento del algoritmo CTC. El CTC es un algoritmo de construcción de árboles de clasificación basado en múltiples muestras, que genera un único árbol de clasificación y proporciona una explicación de la clasificación realizada, abordando uno de los problemas más complejos de clasificación, *class imbalance*. En concreto, en este trabajo nos hemos centrado en bases de datos de tamaño reducido y muy desbalanceadas, desde el punto de vista de la representación de las clases, complicando el aprendizaje a partir de las mismas.

Para este trabajo se han empleado los resultados anteriormente publicados en el trabajo realizado por Fernández et al, en donde 22 diferentes algoritmos, de los cuales 16 son evolutivos y 6 denominados clásicos para la generación de reglas inductivas, son testeados frente a 33 bases de datos desbalanceadas. El conjunto de bases de datos, pertenecen al proyecto KEEL, pertenecen a distintos ámbitos de la vida real, si bien la mayoría se encuadran en el ámbito médico.

Esta comparación es directa ya que además de poner a disposición de los investigadores las mismas bases de datos, se da la opción de descargar exactamente las mismas muestras de entrenamiento y test de una validación cruzada, posibilitando reproducir el experimento bajo las mismas condiciones.

Los resultados indican que en el primer experimento, el CTC muestra un rendimiento mejor que casi la totalidad de los algoritmos, obteniendo la segunda posición en el ranking medio de entre todos los algoritmos. Mediante el test de Shaffer se concluye que además no se encuentran diferencias estadísticamente significativas frente al Ripper que es el primero. Con los resultados obtenidos en este estudio, se ha presentado un artículo al CAEPIA2013 y ha sido seleccionado para su exposición en el congreso.

El método de remuestreo empleado –obtención de muestras equilibradas al 50 % de ambas clases – así como los parámetros del algoritmo CTC derivan de las conclusiones obtenidas en otros trabajos desarrollados por el grupo ALDAPA y que en su momento ya se han detallado.

Se ha realizado esta primera experimentación variando el número de submuestas con el objetivo de estudiar el impacto de las mismas en bases de datos reducidas. Para ello hemos empleado 6 valores distintos y además se ha experimentado con el tamaño de la muestra que también ha resultado ser determinante.

Destaca que con tamaños de submuestras al tamaño máximo que se pueden generar sin repetir casos se obtienen mejores resultados, que empleando tamaños de submuestras basándonos en el tamaño dado por los casos de la clase minoritaria. El número de submuestras también es un parámetro influyente en el rendimiento.

Por lo que a mi refiere este proyecto ha sido una grata experiencia. Me ha dado la oportunidad de afrontar un proyecto que podría calificarse como de investigación, en el que al comienzo tenía mis dudas por desconocer cómo plantearlo y que posteriormente con mi trabajo y la ayuda de Txus ha resultado apasionarme. A su vez me ha permitido profundizar en el paradigma de los árboles de clasificación completando los conocimientos obtenidos en la asignatura m.m.c.c, además de familiarizarme con las herramientas necesarias para llevar a cabo un estudio de esta índole, dándome la posibilidad de emplear herramientas fruto del trabajo y del esfuerzo de un gran número de personas y aprender a tratar los resultados de una manera formal con el objetivo de obtener conclusiones.

Además me ha servido para conocer en primera persona lo que es enfrentarse a un proyecto de larga duración en el que mantener la motivación en todo momento ha sido clave.

Gracias a este proyecto he aprendido a ir documentando y tomando referencias desde el primer momento, descatalogando las fuentes poco serias y profundizando en las que posteriormente emplearía como fuente.

El hecho de haberme leído el citado artículo de Quinlan sobre *induction trees* en m.m.c.c hace un par de años, ha hecho que desde la primera toma de contacto y reunión con Txus me haya interesado el tema.

8.2 Líneas abiertas

Como primer paso en el futuro trabajo, había que completar la experimentación realizada para el conjunto de bases de datos SMOTE ya que únicamente se han realizado 3 ejecuciones variando el tamaño de las muestras pero únicamente con 5 submuestras y el CTC obtiene la sexta posición en el ranking medio. Presuponemos que variando el número de submuestras se podrían alcanzar resultados más competitivos.

Precisamente sería interesante realizar un análisis más profundo sobre la relación que existe entre la representación de los casos de ambas clases mediante el número de submuestras y el rendimiento obtenido en los árboles de clasificación. Esta relación se conoce como *coverage* y mediante fórmulas matemáticas se puede calcular el número de casos, tanto de la clase minoritaria, como mayoritaria, que se cubren para una cantidad concreta de muestras. Analizando este parámetro se observa que en las experimentaciones realizadas para cantidades grandes de muestras se consiguen recopilar el 100% de los casos minoritarios y es en estos donde se alcanzan los mejores resultados. No obstante no se cubren el 100% de los casos pertenecientes a la clase mayoritaria por lo que podrían obtenerse a priori mejores resultados optimizando este parámetro.

También sería interesante afrontar el problema de las bases de datos multiclásicas (todas bi-clásicas hasta ahora) y poder así completar la comparación del CTC en los tres contextos analizados en el artículo de Fernández et al [FGLBH10].

Otra línea abierta seria aplicar el proceso de consolidación a otros algoritmos como el Ripper.

Bibliografía

[AN07] *Asuncion A, Newman DJ UCI machine learning repository. University of California, School of Information and Computer Science, Irvine. http://www.ics.uci.edu/~mlearn/MLRepository.html*

[ALB12] Albisua I. "Análisis del efecto del cambio de la distribución de clases mediante métodos de remuestreo inteligente en árboles de clasificación simples y consolidados". Phd Thesis University of Basque Country, Donostia, Spain (in spanish)

[AAGMMPP09] Albisua I., Arbelaitz O., Gurrutxaga I., Martín J.I., Muguerza J., Pérez J.M., Perona I.: Ob-taining optimal class distribution for decision trees: comparative analysis of CTC and C4.5. In: P. Meseguer, L. Mandow, R. M. Gasca (Eds.). Current Topics in Artficial Intelli-gence, CAEPIA 2009 Selected Papers. LNAI, Vol. 5988, Springer, pp. 101-110

[AAGMP11] Albisua I., Arbelaitz O., Gurrutxaga, I. Muguerza J.,. Pérez J.M. "C4.5 Consolidation Process: An Alternative to Intelligent Oversampling Methods in Class Imbalance Problems". Lecture Notes in Computer Science 7023, Advances in Artificial Intelligence, CAEPIA 2011. Springer-Verlag. Jose A. Lozano, José A. Gámez and José A. Moreno (Eds.), 74-83.

[AMP10] Albisua I., Muguerza J., Pérez J.M. "SMOTE versus submuestreo aleatorio en la búsqueda de la distribución de clases óptima". Actas del V Simposio de Teoría y Aplicaciones de Minería de Datos (TAMIDA 2010) dentro del III Congreso Español de Informática (CEDI 2010), Valencia, Spain, 111-120.

[BPM04] Batista G. E. A. P. A., Prati R. C., Monard M. C. A study of the behavior of several methods for balancing machine learning training data SIGKDD Explorations Newsletter,

[AFLDGSH11] Alcalá-Fdez J., Fernandez A., Luengo J., Derrac J., García S., Sánchez L., Herrera F.: KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17:2-3, 255-287 (2011)

[CJK04] Chawla, N.V., Japkowicz, N., Kolcz, A.: Editorial: special issue on learning from imbalanced data sets. SIGKDD Explorations, 6:1 (2004) 1–6.

[CBHK02] Chawla N.V.,. Bowyer K.W, Hall L.O.,. Kegelmeyer W.P. SMOTE: synthetic minority oversampling technique. Journal of Artificial Intelligence Research 16 (2002) 321-357

[Dem06] Demšar J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Ma-chine Learning Research, 7, 1-30 (2006)

[FGLBH10] Fernandez A., García S., Luengo J., Bernadó-Mansilla E., Herrera F., "Genetics-Based Machine Learning for Rule Induction: State of the Art, Taxonomy and Comparative Study". IEEE Transactions on Evolutionary Computation 14:6 (2010) 913-941, http://dx.doi.org/10.1109/TEVC.2009.2039140

[GH08] García S., Herrera F.: An Extension on "Statistical Comparisons of Classifiers over Multi-ple Data Sets for all Pairwise Comparisons. Journal of Machine Learning Reserarch, 9, 2677-2694 (2008)

[GFLH10] García S., Fernández A., Luengo J., Herrera F.: Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental Analysis of Power. Information Sciences, 180, 2044–2064 (2010)

[GCFH06]García S., Cano J.R., Fernández A., Herrera F.. A proposal of evolutionary prototype selection for class imbalance problems. International Conference on Intelligent Data Engineering and Automated Learning (IDEAL06). Lecture Notes in Computer Science 4224, Springer-Verlag 2006, Burgos (Spain, 2006) 1415-1423

[HG09] He, H., Garcia, E.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. 21(9), 1263–1284 (2009)

[HK06] Han J, Kamber M: Data Mining: Concepts and Techniques The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor Morgan Kaufmann Publishers, March 2006

[MR] *Maimon O, Rokach L Data mining and knowledge discovery handbook*

[Men99] Mena, J. (1999). Data Mining your website. Digital Press

[OZ99] Osmar R. Zaïane, 1999 CMPUT690 Principles of Knowledge Discovery in Databases

[Per06] Pérez. J.M. "Árboles Consolidados: Construcción de un árbol de clasificación basado en múltiples submuestras sin renunciar a la

explicación". Phd Thesis University of Basque Country, Donostia, Spain (in spanish)

[Qui86] Quinlan J.R.. Induction of decision trees. Machine Learning, 1:81{106, 1986.

[Qui92] Quinlan. J. R. C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning). Morgan Kaufmann, 1 edition, 1992.

[Sto74] Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society Series B, 36:111{147

[YW06] Yang Q., Wu X.: 10 challenging problems in data mining research. International Journal of Information Technology & Decision Making. 5(4), 597–604 (2006)

[WP03] Weiss G.M., Provost F.: Learning when Training Data are Costly: The Effect of Class Dis-tribution on Tree Induction. Journal of Artificial Intelligence Research 19, 315-354 (2003)

APENDICE 1

Resumen de resultados de las Bases de datos

En las siguientes páginas se encuentran los resultados obtenidos para todas las ejecuciones realizadas sobre el conjunto de 33 bases de datos. Están organizadas por número y tamaño de las submuestas. Las tablas más reducidas muestran los resultados para las bases de datos que se han decidido no podar fruto de los malos resultados obtenidos podándola.

Apéndice 1
Resumen de resultados de las bases de datos

Resultados para Tamaño de submuestras al máximo y 5 submuestras

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,50000	0,00000	0,05811	0,00000	0,97007	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
abalone19	0,50000	0,00000	0,00767	0,00000	0,99615	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
ecoli-0_vs_1	0,98158	0,00182	0,01545	0,00249	0,97742	0,00339	0,98159	0,00193	0,96566	0,00536	0,97333	0,00000
ecoli1	0,90938	0,01801	0,09889	0,00949	0,93345	0,00651	0,87567	0,01482	0,73157	0,02402	0,91864	0,01278
ecoli2	0,86950	0,01526	0,07076	0,00949	0,95787	0,00556	0,87067	0,02666	0,73566	0,03871	0,95353	0,00385
ecoli3	0,87532	0,04049	0,09581	0,00899	0,94621	0,00529	0,69281	0,08244	0,47470	0,06422	0,94212	0,01334
ecoli4	0,75002	0,05965	0,03987	0,00654	0,97896	0,00343	0,66486	0,12479	0,54648	0,11273	0,98417	0,00498
glass0	0,82708	0,02956	0,17475	0,02079	0,86782	0,01459	0,80367	0,03017	0,60846	0,05098	0,85818	0,02052
glass4-1-2-3_vs_4-5	0,87954	0,01749	0,08321	0,01298	0,94591	0,00817	0,86777	0,03010	0,76364	0,04115	0,95447	0,01411
glass1	0,71326	0,03495	0,25590	0,01384	0,80485	0,00965	0,69618	0,02219	0,43044	0,03390	0,82323	0,00810
glass2	0,51295	0,02895	0,08033	0,00208	0,95804	0,00130	0,01948	0,04356	0,00895	0,02002	0,99795	0,00459
glass4	0,72267	0,10958	0,04948	0,00708	0,97384	0,00352	0,47275	0,20898	0,37484	0,17422	0,98200	0,00758
glass5	0,74024	0,05000	0,03823	0,00390	0,97991	0,00199	0,49837	0,10170	0,37206	0,10097	0,98049	0,00000
glass6	0,85335	0,05483	0,05703	0,01489	0,96682	0,00882	0,87991	0,03304	0,75888	0,05897	0,96324	0,01291
haberman	0,50000	0,00000	0,26467	0,00000	0,84747	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
iris0	0,99000	0,00000	0,00667	0,00000	0,99512	0,00000	0,98974	0,00000	0,98462	0,00000	1,00000	0,00000
new-thyroid1	0,95833	0,01037	0,04744	0,00895	0,97062	0,00566	0,95715	0,01020	0,84687	0,02740	0,95000	0,00878
new-thyroid2	0,94968	0,01095	0,05581	0,01740	0,96468	0,01210	0,94721	0,01082	0,83005	0,03666	0,94222	0,02173
page-blocks0	0,84734	0,03070	0,05249	0,00565	0,97130	0,00291	0,75026	0,07127	0,66053	0,06636	0,98771	0,00047
pima	0,72860	0,01668	0,26978	0,00740	0,79355	0,00753	0,69122	0,00445	0,40287	0,01203	0,79920	0,01591
segment0	0,98763	0,00186	0,00832	0,00094	0,99514	0,00055	0,98595	0,00176	0,96619	0,00373	0,99394	0,00095
vehicle0	0,95662	0,00219	0,08131	0,00505	0,94515	0,00379	0,91980	0,00151	0,78821	0,01034	0,91711	0,00853
vehicle1	0,50000	0,00000	0,25650	0,00000	0,85288	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vehicle2	0,90764	0,01331	0,15316	0,02152	0,89713	0,01383	0,78206	0,03857	0,59178	0,06150	0,89947	0,01195
vehicle3	0,50000	0,00000	0,25059	0,00000	0,85676	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vowel0	0,95601	0,00517	0,03523	0,00492	0,98039	0,00283	0,93826	0,01398	0,80480	0,02149	0,96904	0,00588
wisconsin	0,96646	0,00628	0,06819	0,00222	0,94832	0,00161	0,91818	0,00325	0,84798	0,00512	0,95895	0,00100
yeast1	0,71147	0,01114	0,24690	0,00707	0,83106	0,00543	0,65324	0,01046	0,37232	0,01621	0,85536	0,00940
yeast-2_vs_8	0,77391	0,00000	0,02077	0,00000	0,98927	0,00000	0,72831	0,00000	0,65905	0,00000	0,99783	0,00000
yeast3	0,92585	0,00694	0,05431	0,00194	0,96935	0,00113	0,87218	0,00366	0,73023	0,00762	0,96472	0,00237
yeast4	0,52508	0,03552	0,03464	0,00037	0,98237	0,00021	0,03335	0,04788	0,01816	0,02772	0,99916	0,00125
yeast5	0,72394	0,08729	0,02601	0,00582	0,98667	0,00301	0,43306	0,16847	0,33305	0,15864	0,99167	0,00354
yeast6	0,50000	0,00000	0,02358	0,00000	0,98807	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,74307	0,03502	0,21030	0,01686	0,87723	0,01095	0,68776	0,03414	0,17849	0,03154	0,80069	0,01784
abalone19	0,58431	0,07085	0,23421	0,01377	0,86609	0,00910	0,52375	0,12184	0,01277	0,00971	0,76838	0,01344
glass2	0,74228	0,04654	0,22640	0,00566	0,86243	0,00604	0,65930	0,06609	0,21089	0,02499	0,78662	0,00389
haberman	0,54622	0,01901	0,51439	0,01362	0,56407	0,01675	0,50404	0,01291	0,01663	0,01963	0,45778	0,02061
vehicle1	0,56124	0,01084	0,43664	0,01931	0,65802	0,02396	0,55705	0,00353	0,09415	0,00947	0,56732	0,03667
vehicle3	0,56008	0,00670	0,42220	0,01868	0,67338	0,02316	0,57062	0,00731	0,11469	0,00988	0,58335	0,03725
yeast4	0,82701	0,02608	0,14690	0,00763	0,91845	0,00449	0,77025	0,04289	0,20143	0,02896	0,85948	0,00763
yeast6	0,84070	0,02334	0,13084	0,01606	0,92796	0,00914	0,82092	0,01183	0,28205	0,03130	0,87125	0,01679

Apéndice 1
Resumen de resultados de las bases de datos

Resultados para Tamaño de submuestras al máximo y 120% submuestras

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,51902	0,02613	0,05914	0,00230	0,96949	0,00125	0,04224	0,05930	0,02123	0,03364	0,99783	0,00324
abalone19	0,50000	0,00000	0,00767	0,00000	0,99615	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
ecoli-0_vs_1	0,98179	0,00196	0,01545	0,00249	0,97735	0,00332	0,98163	0,00197	0,96561	0,00531	0,97333	0,00000
ecoli1	0,87137	0,05185	0,10829	0,02061	0,92861	0,01392	0,84759	0,03032	0,70093	0,05530	0,92404	0,01906
ecoli2	0,84975	0,02018	0,06898	0,00998	0,95928	0,00608	0,85500	0,00971	0,73268	0,03268	0,96193	0,01037
ecoli3	0,87897	0,02581	0,08515	0,01032	0,95264	0,00595	0,71188	0,03870	0,51977	0,05146	0,95816	0,01053
ecoli4	0,80570	0,01151	0,03687	0,00493	0,98028	0,00273	0,83216	0,01741	0,68170	0,02831	0,97853	0,00514
glass0	0,80215	0,02209	0,19680	0,01153	0,85883	0,00991	0,78588	0,01123	0,57944	0,02352	0,85010	0,01530
glass-0-1-2-3_vs_45-6	0,86311	0,02318	0,08505	0,01694	0,94436	0,01108	0,87051	0,02558	0,76239	0,04734	0,94958	0,01093
glass1	0,72604	0,04283	0,25032	0,02200	0,80888	0,01678	0,70164	0,03043	0,44318	0,05149	0,83069	0,02719
glass2	0,54328	0,06462	0,08219	0,00416	0,95668	0,00279	0,07668	0,11266	0,04525	0,07071	0,99179	0,01124
glass4	0,86538	0,08433	0,04667	0,00741	0,97485	0,00389	0,76283	0,17020	0,58556	0,14058	0,96910	0,00961
glass5	0,89951	0,09904	0,02607	0,00414	0,98620	0,00220	0,79639	0,17683	0,67392	0,13945	0,98244	0,00816
glass6	0,84121	0,03395	0,05978	0,01205	0,96522	0,00741	0,86647	0,02372	0,74603	0,04040	0,96432	0,01648
haberman	0,50000	0,00000	0,26467	0,00000	0,84747	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
iris0	0,99000	0,00000	0,00667	0,00000	0,99512	0,00000	0,98974	0,00000	0,98462	0,00000	1,00000	0,00000
new-thyroid1	0,95865	0,01160	0,03814	0,01061	0,97648	0,00700	0,95566	0,01119	0,87390	0,02896	0,96444	0,01083
new-thyroid2	0,94310	0,02430	0,05023	0,02011	0,96876	0,01318	0,94132	0,02536	0,83883	0,05548	0,95333	0,01826
page-blocks0	0,86335	0,00188	0,04934	0,00151	0,97292	0,00079	0,78739	0,01199	0,69580	0,01298	0,98734	0,00068
pima	0,70397	0,02042	0,28047	0,00705	0,78692	0,00505	0,67451	0,01346	0,37550	0,01960	0,79760	0,01244
segment0	0,98691	0,00256	0,00815	0,00177	0,99524	0,00104	0,98502	0,00244	0,96686	0,00703	0,99454	0,00209
vehicle0	0,95893	0,00296	0,08178	0,00328	0,94485	0,00245	0,91720	0,00172	0,78656	0,00700	0,91834	0,00530
vehicle1	0,50000	0,00000	0,25650	0,00000	0,85288	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vehicle2	0,90373	0,00660	0,15930	0,01205	0,89284	0,00792	0,77328	0,02359	0,57675	0,03555	0,89631	0,01067
vehicle3	0,50000	0,00000	0,25059	0,00000	0,85676	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vowel0	0,95556	0,00356	0,02835	0,00101	0,98430	0,00060	0,94325	0,00393	0,83752	0,00325	0,97751	0,00199
wisconsin	0,96935	0,00015	0,06731	0,00000	0,94896	0,00000	0,91932	0,00000	0,84999	0,00000	0,95940	0,00000
yeast1	0,71617	0,01258	0,24608	0,00422	0,83124	0,00293	0,65693	0,02008	0,37659	0,01982	0,85403	0,01307
yeast-2_vs_8	0,77391	0,00000	0,02077	0,00000	0,98927	0,00000	0,72831	0,00000	0,65905	0,00000	0,99783	0,00000
yeast3	0,92077	0,00673	0,05417	0,00335	0,96939	0,00192	0,87331	0,00637	0,73271	0,01477	0,96473	0,00291
yeast4	0,66772	0,07483	0,03504	0,00158	0,98209	0,00085	0,23788	0,11976	0,14503	0,07622	0,99511	0,00346
yeast5	0,95825	0,04273	0,01819	0,00309	0,99062	0,00159	0,83527	0,07500	0,67931	0,07363	0,98903	0,00266
yeast6	0,59801	0,06864	0,02480	0,00121	0,98740	0,00063	0,19903	0,14985	0,12493	0,10239	0,99558	0,00300

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,73563	0,02679	0,15739	0,00526	0,91081	0,00299	0,69814	0,04774	0,25286	0,03685	0,85763	0,00491
abalone19	0,49152	0,04362	0,14730	0,01190	0,91993	0,00702	0,44437	0,07033	0,01643	0,00688	0,85703	0,01211
glass2	0,73804	0,04515	0,16281	0,03084	0,90566	0,02015	0,64425	0,07907	0,26494	0,07304	0,86372	0,03264
haberman	0,52731	0,01110	0,52738	0,01739	0,54888	0,02200	0,49202	0,01395	0,00055	0,01998	0,44089	0,02522
vehicle1	0,56306	0,00613	0,43566	0,01800	0,65883	0,02471	0,55508	0,01081	0,09342	0,00807	0,56955	0,04200
vehicle3	0,56230	0,00795	0,42480	0,02104	0,67160	0,02441	0,56886	0,00802	0,10856	0,01357	0,58175	0,03801
yeast4	0,82088	0,02343	0,09610	0,01221	0,94812	0,00708	0,75824	0,02463	0,28120	0,01688	0,91317	0,01424
yeast6	0,82251	0,01540	0,07250	0,00578	0,96152	0,00323	0,81431	0,02145	0,30065	0,02513	0,93238	0,00550

Apéndice 1
Resumen de resultados de las bases de datos

Resultados para Tamaño de submuestras al máximo y 200% submuestras

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,54158	0,03682	0,05983	0,00435	0,96909	0,00227	0,06812	0,07001	0,03864	0,05835	0,99601	0,00298
abalone19	0,50000	0,00000	0,00767	0,00000	0,99615	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
ecoli-0_vs_1	0,98179	0,00196	0,01545	0,00249	0,97735	0,00332	0,98163	0,00197	0,96561	0,00531	0,97333	0,00000
ecoli1	0,88916	0,01562	0,09810	0,00469	0,93567	0,00523	0,85541	0,01101	0,72618	0,01301	0,93643	0,00806
ecoli2	0,96150	0,02767	0,05947	0,00865	0,96485	0,00523	0,87599	0,01077	0,77136	0,02394	0,96759	0,00912
ecoli3	0,89657	0,03006	0,08817	0,00771	0,95061	0,00432	0,73776	0,03057	0,52817	0,04520	0,94943	0,00753
ecoli4	0,79877	0,01069	0,03153	0,00340	0,98326	0,00187	0,81452	0,01833	0,69792	0,02769	0,98611	0,00362
glass0	0,82005	0,00963	0,17652	0,01022	0,86712	0,00856	0,79856	0,02093	0,60138	0,02678	0,85690	0,02386
glass-0-1-2-3_vs_4-5-6	0,86658	0,02079	0,08516	0,00839	0,94457	0,00521	0,86854	0,01678	0,75961	0,02670	0,95080	0,00442
glass1	0,69293	0,02607	0,25978	0,01921	0,80408	0,01377	0,68146	0,02863	0,41672	0,04611	0,83476	0,01961
glass2	0,57506	0,05043	0,07754	0,00255	0,95942	0,00135	0,13429	0,09211	0,09260	0,06102	0,99595	0,00424
glass4	0,84020	0,07128	0,05050	0,01528	0,97290	0,00822	0,67013	0,16127	0,49446	0,17185	0,96902	0,01084
glass5	0,93878	0,06900	0,02423	0,00212	0,98701	0,00109	0,89690	0,12568	0,74664	0,10533	0,97756	0,00436
glass6	0,86688	0,05840	0,05697	0,01606	0,96716	0,00929	0,85043	0,04854	0,74670	0,07321	0,97297	0,01011
haberman	0,50000	0,00000	0,26467	0,00000	0,84747	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
iris0	0,99000	0,00000	0,00667	0,00000	0,99512	0,00000	0,98974	0,00000	0,98462	0,00000	1,00000	0,00000
new-thyroid1	0,95992	0,00544	0,03349	0,00208	0,97978	0,00121	0,95580	0,01004	0,88193	0,00898	0,97111	0,00248
new-thyroid2	0,95492	0,01394	0,05023	0,01783	0,96864	0,01192	0,94597	0,01604	0,84079	0,04436	0,95111	0,02201
page-blocks0	0,86257	0,00279	0,04850	0,00070	0,97336	0,00037	0,79383	0,01403	0,70282	0,00934	0,98705	0,00245
pima	0,69065	0,02429	0,27971	0,01179	0,78714	0,00783	0,67852	0,02151	0,37870	0,03165	0,79440	0,01438
segment0	0,98768	0,00227	0,00745	0,00169	0,99565	0,00099	0,98748	0,00238	0,96971	0,00678	0,99454	0,00149
vehicle0	0,95758	0,00169	0,07871	0,00657	0,94679	0,00487	0,92242	0,00386	0,79560	0,01414	0,91959	0,00937
vehicle1	0,50000	0,00000	0,25650	0,00000	0,85288	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vehicle2	0,90998	0,00501	0,14584	0,00906	0,90063	0,00697	0,80961	0,00804	0,62332	0,01917	0,89277	0,01234
vehicle3	0,50000	0,00000	0,25059	0,00000	0,85676	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vowel0	0,96033	0,00457	0,02633	0,00357	0,98540	0,00202	0,95144	0,00518	0,85021	0,01706	0,97818	0,00390
wisconsin	0,97178	0,00152	0,06672	0,00513	0,94918	0,00403	0,92167	0,00575	0,85186	0,01116	0,95626	0,00670
yeast1	0,69760	0,00975	0,25673	0,00854	0,82299	0,00765	0,64840	0,01114	0,35474	0,01578	0,84114	0,01779
yeast-2_vs_8	0,77391	0,00000	0,02077	0,00000	0,98927	0,00000	0,72831	0,00000	0,65905	0,00000	0,99783	0,00000
yeast3	0,89387	0,01511	0,05619	0,00246	0,96838	0,00137	0,85689	0,01339	0,71543	0,01415	0,96609	0,00281
yeast4	0,68694	0,08428	0,03491	0,00130	0,98214	0,00069	0,27262	0,13329	0,16993	0,08982	0,99427	0,00302
yeast5	0,97577	0,00814	0,01792	0,00194	0,99073	0,00101	0,89404	0,02770	0,72007	0,03094	0,98722	0,00167
yeast6	0,76342	0,06404	0,02291	0,00135	0,98832	0,00069	0,46897	0,13383	0,33406	0,09884	0,99282	0,00261

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,71874	0,04210	0,16970	0,01230	0,90308	0,00745	0,68486	0,04346	0,22923	0,03178	0,84565	0,01269
abalone19	0,47611	0,01652	0,13580	0,00760	0,92672	0,00436	0,43430	0,05194	0,01637	0,00560	0,86881	0,00780
glass2	0,73783	0,06527	0,15728	0,03327	0,90932	0,02078	0,62030	0,05907	0,26344	0,08752	0,87603	0,03535
haberman	0,53762	0,01044	0,51036	0,01940	0,57299	0,02682	0,50278	0,00978	0,01145	0,01425	0,47022	0,03376
vehicle1	0,56445	0,00247	0,42907	0,01364	0,66728	0,01823	0,55915	0,00633	0,09865	0,00583	0,58069	0,03050
vehicle3	0,56295	0,00901	0,41393	0,02627	0,68378	0,02849	0,56986	0,00833	0,11670	0,02104	0,60006	0,04471
yeast4	0,80277	0,02922	0,09827	0,01191	0,94696	0,00686	0,74795	0,03960	0,26970	0,01797	0,91178	0,01451
yeast6	0,80512	0,03529	0,06913	0,01242	0,96330	0,00685	0,79110	0,03007	0,31176	0,04703	0,93680	0,01189

Apéndice 1
Resumen de resultados de las bases de datos

Resultados para Tamaño de submuestras al máximo y 400% submuestras

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,53113	0,05401	0,05949	0,00223	0,96928	0,00121	0,05051	0,11294	0,02976	0,07092	0,99674	0,00486
abalone19	0,50000	0,00000	0,00767	0,00000	0,99615	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
ecoli-0_vs_1	0,98108	0,00196	0,01636	0,00249	0,97613	0,00332	0,98091	0,00197	0,96367	0,00531	0,97333	0,00000
ecoli1	0,88101	0,03156	0,09815	0,00627	0,93530	0,00392	0,86401	0,01423	0,72945	0,01899	0,93020	0,00474
ecoli2	0,86430	0,02745	0,05951	0,00729	0,96492	0,00414	0,87291	0,02706	0,76792	0,03404	0,96828	0,00354
ecoli3	0,86554	0,05270	0,08579	0,00922	0,95221	0,00542	0,71921	0,02741	0,52487	0,03941	0,95672	0,01172
ecoli4	0,80449	0,01267	0,02973	0,00208	0,98420	0,00114	0,83562	0,01656	0,72335	0,01632	0,98612	0,00281
glass0	0,79864	0,01911	0,18144	0,01541	0,86511	0,01087	0,78565	0,02414	0,58561	0,03776	0,86616	0,01149
glass-0-1-2-3_vs_4-5-6	0,86643	0,03207	0,08888	0,01712	0,94219	0,01118	0,85935	0,03173	0,74727	0,04958	0,94958	0,01341
glass1	0,73284	0,01478	0,24959	0,01692	0,80697	0,01344	0,71151	0,02318	0,45117	0,03837	0,81868	0,02209
glass2	0,63275	0,08036	0,08591	0,00964	0,95441	0,00568	0,18617	0,13398	0,11104	0,08172	0,98272	0,01625
glass4	0,89685	0,02053	0,04120	0,00384	0,97780	0,00212	0,79134	0,07581	0,62723	0,06976	0,97102	0,00421
glass5	0,97049	0,04218	0,02047	0,00416	0,98903	0,00224	0,93888	0,08421	0,81348	0,06907	0,98146	0,00636
glass6	0,81011	0,06970	0,06166	0,01172	0,96424	0,00676	0,85544	0,03500	0,73511	0,05443	0,96541	0,00725
haberman	0,50000	0,00000	0,26467	0,00000	0,84747	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
iris0	0,99000	0,00000	0,00667	0,00000	0,99512	0,00000	0,98974	0,00000	0,98462	0,00000	1,00000	0,00000
new-thyroid1	0,96571	0,00792	0,03163	0,00389	0,98087	0,00235	0,96200	0,00861	0,88931	0,01382	0,97111	0,00248
new-thyroid2	0,95500	0,01774	0,03907	0,01878	0,97570	0,01251	0,95516	0,01726	0,87427	0,04782	0,96333	0,02137
page-blocks0	0,86209	0,00185	0,04799	0,00090	0,97361	0,00046	0,80151	0,01061	0,70909	0,00941	0,98616	0,00109
pima	0,68188	0,01060	0,28257	0,00930	0,78406	0,00700	0,67713	0,01267	0,37430	0,02164	0,78940	0,00829
segment0	0,98628	0,00346	0,00745	0,00108	0,99565	0,00063	0,98567	0,00338	0,96958	0,00447	0,99525	0,00045
vehicle0	0,95805	0,00161	0,07942	0,00525	0,94642	0,00388	0,92128	0,00326	0,79299	0,01141	0,91927	0,00786
vehicle1	0,50000	0,00000	0,25650	0,00000	0,85288	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vehicle2	0,90908	0,00708	0,15152	0,01151	0,89741	0,00791	0,79172	0,02102	0,60299	0,03155	0,89667	0,01297
vehicle3	0,50000	0,00000	0,25059	0,00000	0,85676	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vowel0	0,95804	0,00518	0,02815	0,00280	0,98441	0,00161	0,94121	0,00787	0,83866	0,01216	0,97818	0,00420
wisconsin	0,96993	0,00131	0,06819	0,00196	0,94821	0,00167	0,91885	0,00104	0,84824	0,00392	0,95760	0,00402
yeast1	0,69726	0,00740	0,25080	0,00588	0,82789	0,00509	0,65199	0,00289	0,36551	0,00908	0,84948	0,01028
yeast-2_vs_8	0,77391	0,00000	0,02077	0,00000	0,98927	0,00000	0,72831	0,00000	0,65905	0,00000	0,99783	0,00000
yeast3	0,89751	0,01970	0,05538	0,00240	0,96878	0,00140	0,86463	0,01069	0,72275	0,01063	0,96518	0,00371
yeast4	0,70184	0,05961	0,03841	0,00357	0,98029	0,00189	0,30025	0,09209	0,16838	0,04842	0,99036	0,00426
yeast5	0,96710	0,00976	0,01738	0,00056	0,99102	0,00030	0,87537	0,01828	0,71775	0,00471	0,98889	0,00155
yeast6	0,79334	0,07591	0,02332	0,00139	0,98809	0,00071	0,52278	0,15145	0,36850	0,10951	0,99116	0,00314

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,73163	0,01992	0,13983	0,00405	0,92158	0,00247	0,69015	0,02798	0,27144	0,01939	0,87765	0,00506
abalone19	0,49243	0,04007	0,13455	0,02097	0,92740	0,01213	0,39843	0,10225	0,01319	0,00658	0,87016	0,02151
glass2	0,72004	0,04119	0,13668	0,00888	0,92364	0,00569	0,54441	0,12537	0,24523	0,06284	0,90441	0,01574
haberman	0,54926	0,00520	0,51832	0,00674	0,56398	0,01147	0,49768	0,00500	0,00337	0,00855	0,45956	0,01621
vehicle1	0,56632	0,00512	0,42149	0,01545	0,67592	0,01986	0,56169	0,01071	0,10455	0,01169	0,59307	0,03384
vehicle3	0,56728	0,00305	0,40921	0,01872	0,68833	0,02188	0,57399	0,00293	0,12343	0,00820	0,60544	0,03588
yeast4	0,79065	0,01592	0,09801	0,00395	0,95299	0,00216	0,70365	0,02082	0,26108	0,01963	0,92519	0,00367
yeast6	0,82868	0,03154	0,07520	0,00818	0,96003	0,00466	0,81930	0,02203	0,29537	0,01613	0,92947	0,00901

Apéndice 1
Resumen de resultados de las bases de datos

Resultados para Tamaño de submuestras al máximo y 600% submuestras

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,52579	0,02646	0,06120	0,00306	0,96838	0,00168	0,05226	0,04771	0,01762	0,01637	0,99565	0,00417
abalone19	0,50000	0,00000	0,00767	0,00000	0,99615	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
ecoli-0_vs_1	0,98108	0,00196	0,01636	0,00249	0,97613	0,00332	0,98091	0,00197	0,96367	0,00531	0,97333	0,00000
ecoli1	0,86292	0,03592	0,09880	0,01183	0,93490	0,00793	0,86923	0,01573	0,72852	0,03143	0,92566	0,01000
ecoli2	0,85780	0,03998	0,06124	0,00976	0,96396	0,00568	0,86665	0,02631	0,75964	0,04046	0,96831	0,00559
ecoli3	0,89351	0,02783	0,08692	0,00791	0,95110	0,00462	0,76980	0,02690	0,55415	0,03456	0,94556	0,00894
ecoli4	0,80394	0,01459	0,03090	0,00449	0,98357	0,00240	0,83517	0,01823	0,71597	0,03673	0,98486	0,00346
glass0	0,81117	0,02205	0,17654	0,01122	0,86888	0,00751	0,78945	0,02426	0,59580	0,03076	0,87069	0,01048
glass-0-1-2-3_vs_4-5-6	0,86286	0,03539	0,09262	0,01201	0,93930	0,00828	0,86280	0,01407	0,74218	0,02973	0,94216	0,01284
glass1	0,71386	0,02907	0,25138	0,02102	0,80792	0,01691	0,70182	0,02532	0,44151	0,04575	0,82735	0,02377
glass2	0,60022	0,05464	0,08126	0,00705	0,95708	0,00392	0,19110	0,11149	0,12250	0,08954	0,98774	0,00690
glass4	0,86802	0,02071	0,05156	0,00809	0,97207	0,00459	0,75399	0,04824	0,55682	0,02783	0,96105	0,01082
glass5	0,97122	0,02718	0,01953	0,00208	0,98949	0,00110	0,94470	0,02540	0,83620	0,02471	0,98341	0,00267
glass6	0,81723	0,06906	0,05980	0,01461	0,96542	0,00858	0,85016	0,03055	0,73949	0,06163	0,96973	0,01302
haberman	0,50000	0,00000	0,26467	0,00000	0,84747	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
iris0	0,99000	0,00000	0,00667	0,00000	0,99512	0,00000	0,98974	0,00000	0,98462	0,00000	1,00000	0,00000
new-thyroid1	0,96627	0,01242	0,02977	0,00416	0,98200	0,00245	0,96554	0,01299	0,89584	0,01601	0,97222	0,00000
new-thyroid2	0,95175	0,01396	0,04279	0,01521	0,97313	0,01036	0,95057	0,01419	0,86564	0,03606	0,96000	0,01730
page-blocks0	0,85943	0,00166	0,04781	0,00129	0,97373	0,00069	0,79979	0,00901	0,70903	0,00986	0,98673	0,00072
pima	0,68021	0,01176	0,27816	0,00940	0,79014	0,00813	0,67377	0,00743	0,37749	0,01822	0,80560	0,01276
segment0	0,98860	0,00126	0,00546	0,00084	0,99682	0,00049	0,98889	0,00153	0,97764	0,00340	0,99677	0,00111
vehicle0	0,95789	0,00287	0,08179	0,00817	0,94450	0,00592	0,92196	0,00457	0,78907	0,01837	0,91400	0,01178
vehicle1	0,50000	0,00000	0,25650	0,00000	0,85288	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vehicle2	0,91155	0,01076	0,13992	0,00815	0,90527	0,00499	0,81040	0,01761	0,63442	0,02563	0,90363	0,00618
vehicle3	0,50000	0,00000	0,25059	0,00000	0,85676	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vowel0	0,95222	0,00894	0,02856	0,00400	0,98420	0,00224	0,93037	0,01277	0,83475	0,02206	0,97996	0,00431
wisconsin	0,96935	0,00015	0,06731	0,00000	0,94896	0,00000	0,91932	0,00000	0,84999	0,00000	0,95940	0,00000
yeast1	0,69810	0,01016	0,24731	0,00588	0,83098	0,00392	0,65157	0,01339	0,37026	0,01790	0,85611	0,00742
yeast1-2_vs_8	0,77391	0,00000	0,02077	0,00000	0,98927	0,00000	0,72831	0,00000	0,65905	0,00000	0,99783	0,00000
yeast3	0,88944	0,03164	0,05794	0,00278	0,96738	0,00153	0,85110	0,01347	0,70712	0,01681	0,96533	0,00135
yeast4	0,71686	0,07214	0,03477	0,00169	0,98218	0,00090	0,36080	0,11766	0,22925	0,07161	0,99274	0,00337
yeast5	0,96686	0,01744	0,01658	0,00194	0,99143	0,00101	0,88564	0,03656	0,73205	0,03609	0,98903	0,00238
yeast6	0,87193	0,02564	0,02197	0,00131	0,98874	0,00067	0,72727	0,03732	0,52586	0,03616	0,98827	0,00138

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,71974	0,02767	0,15872	0,00502	0,91010	0,00279	0,68746	0,04027	0,23255	0,03501	0,85694	0,00432
abalone19	0,45562	0,04893	0,12257	0,00521	0,93429	0,00312	0,34868	0,07365	0,01215	0,00645	0,88257	0,00555
glass2	0,72281	0,02491	0,13938	0,02881	0,92150	0,01729	0,56126	0,07987	0,25027	0,10148	0,90056	0,02803
haberman	0,53432	0,01017	0,52088	0,01096	0,56089	0,01432	0,49426	0,00946	-0,00069	0,01570	0,45689	0,01761
vehicle1	0,56657	0,00134	0,42174	0,01602	0,67513	0,02182	0,56216	0,01236	0,10560	0,01005	0,59182	0,03896
vehicle3	0,56348	0,00359	0,41417	0,01622	0,68418	0,01998	0,56931	0,00496	0,11494	0,00329	0,60038	0,03399
yeast4	0,79858	0,02158	0,09314	0,00338	0,95002	0,00194	0,73460	0,03638	0,26843	0,02297	0,91806	0,00438
yeast6	0,78801	0,03945	0,06630	0,00733	0,96501	0,00398	0,77102	0,04616	0,29423	0,04607	0,94024	0,00653

Apéndice 1
Resumen de resultados de las bases de datos

Resultados para Tamaño de submuestras al máximo y 200 submuestras

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,52434	0,02120	0,06085	0,00260	0,96856	0,00141	0,06994	0,03910	0,02752	0,01678	0,99565	0,00885
abalone19	0,50000	0,00000	0,00767	0,00000	0,99615	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
ecoli-0_vs_1	0,98250	0,00160	0,01455	0,00203	0,97856	0,00271	0,98236	0,00161	0,96755	0,00433	0,97333	0,00951
ecoli1	0,87330	0,01871	0,09820	0,00594	0,93532	0,00379	0,86783	0,01267	0,72936	0,01736	0,92716	0,01411
ecoli2	0,82852	0,01437	0,05944	0,00366	0,96524	0,00214	0,85815	0,01014	0,76009	0,01491	0,97397	0,00951
ecoli3	0,85319	0,02805	0,09051	0,00823	0,94912	0,00469	0,74826	0,01891	0,53143	0,03849	0,94550	0,01411
ecoli4	0,80245	0,01240	0,03210	0,00386	0,98294	0,00207	0,82845	0,01493	0,70342	0,03100	0,98423	0,00951
glass0	0,81335	0,02021	0,17205	0,01527	0,87038	0,01211	0,80028	0,01860	0,61139	0,03383	0,86621	0,01411
glass-0-1-2-3_vs_4-5-6	0,84905	0,00860	0,09539	0,00419	0,93789	0,00289	0,85244	0,00534	0,73037	0,01062	0,94455	0,00951
glass1	0,72510	0,00840	0,25590	0,01211	0,80127	0,01081	0,70941	0,01123	0,44015	0,02421	0,80937	0,01411
glass2	0,66275	0,03860	0,08219	0,00964	0,95639	0,00532	0,26276	0,05529	0,16579	0,05611	0,98364	0,00951
glass4	0,88450	0,03239	0,04870	0,01082	0,97364	0,00595	0,78102	0,08810	0,58834	0,10513	0,96600	0,01953
glass5	0,98171	0,02077	0,01860	0,00735	0,99007	0,00399	0,93415	0,00371	0,82653	0,04903	0,98537	0,01411
glass6	0,81775	0,03619	0,05522	0,00389	0,96832	0,00219	0,84785	0,01597	0,75062	0,01933	0,97622	0,00951
haberman	0,50000	0,00000	0,26467	0,00000	0,84747	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
iris0	0,99000	0,00000	0,00667	0,00000	0,99512	0,00000	0,98974	0,00000	0,98462	0,00000	1,00000	0,00000
new-thyroid1	0,95579	0,01174	0,04372	0,01299	0,97306	0,00831	0,95188	0,01232	0,85557	0,03741	0,95778	0,01411
new-thyroid2	0,96365	0,01286	0,03721	0,01918	0,97678	0,01282	0,96329	0,01273	0,88162	0,04737	0,96222	0,01411
page-blocks0	0,86218	0,00132	0,04711	0,00114	0,97410	0,00060	0,80336	0,00890	0,71380	0,00946	0,98685	0,00951
pima	0,67356	0,01104	0,29585	0,00251	0,77587	0,00203	0,65507	0,00252	0,33958	0,00517	0,78680	0,00951
segment0	0,98764	0,00089	0,00546	0,00079	0,99682	0,00046	0,98784	0,00046	0,97758	0,00311	0,99717	0,00951
vehicle0	0,95893	0,00222	0,07398	0,00179	0,95032	0,00134	0,92507	0,00088	0,80558	0,00402	0,92640	0,00951
vehicle1	0,50000	0,00000	0,25650	0,00000	0,85288	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vehicle2	0,91706	0,00721	0,13427	0,00725	0,90883	0,00450	0,82162	0,02263	0,65150	0,02591	0,90526	0,01411
vehicle3	0,50000	0,00000	0,25059	0,00000	0,85676	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vowel0	0,95777	0,00341	0,02997	0,00153	0,98337	0,00089	0,94441	0,00373	0,83081	0,00552	0,97528	0,00951
wisconsin	0,96942	0,00000	0,06731	0,00000	0,94896	0,00000	0,91932	0,00000	0,84999	0,00000	0,95940	0,00000
yeast1	0,68563	0,01670	0,25471	0,00685	0,82521	0,00431	0,64557	0,01641	0,35518	0,02179	0,84701	0,01411
yeast-2_vs_8	0,77391	0,00000	0,02077	0,00000	0,98927	0,00000	0,72831	0,00000	0,65905	0,00000	0,99783	0,00000
yeast3	0,88513	0,00845	0,05565	0,00155	0,96865	0,00094	0,85996	0,00712	0,71990	0,00441	0,96609	0,00951
yeast4	0,72049	0,03505	0,03693	0,00250	0,98105	0,00132	0,35105	0,03798	0,20228	0,02339	0,99106	0,01411
yeast5	0,97526	0,01141	0,01711	0,00246	0,99115	0,00127	0,88271	0,02832	0,72541	0,03853	0,98875	0,01411
yeast6	0,85958	0,05490	0,02049	0,00216	0,98952	0,00110	0,67849	0,10422	0,50594	0,09570	0,99047	0,00951

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,75904	0,04243	0,13205	0,01147	0,92618	0,00664	0,72590	0,05241	0,29944	0,05742	0,88312	0,00885
abalone19	0,52962	0,07288	0,13905	0,01410	0,92470	0,00822	0,45632	0,08625	0,01961	0,00970	0,86509	0,01411
glass2	0,72865	0,03358	0,12164	0,00873	0,93245	0,00487	0,59142	0,07869	0,28919	0,06853	0,91885	0,00951
haberman	0,53878	0,00696	0,51636	0,00831	0,56660	0,01441	0,49745	0,00455	0,00349	0,00762	0,46400	0,01953
vehicle1	0,57148	0,00502	0,42481	0,02025	0,66974	0,02589	0,56534	0,00539	0,10942	0,00436	0,58227	0,04273
vehicle3	0,56474	0,00360	0,41134	0,01660	0,68693	0,02024	0,57130	0,00819	0,11846	0,00843	0,60382	0,03489
yeast4	0,79478	0,01969	0,09287	0,00367	0,95020	0,00201	0,71095	0,02080	0,25706	0,01557	0,91918	0,00399
yeast6	0,85639	0,01637	0,06901	0,00784	0,96348	0,00424	0,83890	0,00901	0,32387	0,03423	0,95512	0,00789

Apéndice 1
Resumen de resultados de las bases de datos

Resultados para submuestras al tamaño de la clase minoritaria y 5 submuestras

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,50000	0,00000	0,05811	0,00000	0,97007	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
abalone19	0,50000	0,00000	0,00767	0,00000	0,99615	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
ecoli-0_vs_1	0,97632	0,00655	0,02273	0,00850	0,96813	0,01069	0,97601	0,00655	0,95053	0,01757	0,97333	0,00000
ecoli1	0,93219	0,01030	0,09219	0,01116	0,93990	0,00720	0,85688	0,02590	0,73920	0,03364	0,94566	0,00768
ecoli2	0,88301	0,01437	0,08148	0,00653	0,95177	0,00398	0,83039	0,02372	0,68629	0,02621	0,95333	0,00906
ecoli3	0,78549	0,13906	0,09350	0,01057	0,94857	0,00509	0,50996	0,24059	0,36195	0,18470	0,96534	0,01574
ecoli4	0,69440	0,12149	0,05410	0,01076	0,97145	0,00569	0,45804	0,22769	0,34178	0,19136	0,98048	0,01050
glass0	0,82128	0,02301	0,19621	0,02066	0,85515	0,01111	0,75970	0,04577	0,54665	0,06189	0,86113	0,01537
glass-0-1-2-3_vs_4-5-6	0,87659	0,03763	0,09723	0,00610	0,93607	0,00441	0,85346	0,00981	0,72931	0,01552	0,93966	0,01011
glass1	0,71319	0,03550	0,27444	0,01300	0,78704	0,00823	0,68130	0,03011	0,39619	0,03865	0,79571	0,02658
glass2	0,50000	0,00000	0,07940	0,00000	0,95682	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
glass4	0,55552	0,07684	0,06627	0,00510	0,96532	0,00306	0,16149	0,18936	0,09067	0,11475	0,98705	0,01298
glass5	0,63610	0,08828	0,03730	0,00739	0,98071	0,00389	0,27604	0,17768	0,22633	0,16580	0,99220	0,00740
glass6	0,91261	0,01166	0,06536	0,01492	0,96115	0,00920	0,89713	0,02648	0,75141	0,04924	0,94811	0,01510
haberman	0,50000	0,00000	0,26467	0,00000	0,84747	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
iris0	0,98800	0,00447	0,00800	0,00298	0,99415	0,00218	0,98768	0,00459	0,98154	0,00688	1,00000	0,00000
new-thyroid1	0,91286	0,04376	0,08000	0,02448	0,95006	0,01645	0,89217	0,08661	0,73652	0,08820	0,92333	0,02897
new-thyroid2	0,93175	0,02022	0,06884	0,02492	0,95672	0,01632	0,93039	0,02079	0,78705	0,06739	0,93111	0,03254
page-blocks0	0,88705	0,03204	0,05406	0,00481	0,97019	0,00245	0,78815	0,07171	0,67334	0,06217	0,97810	0,00199
pima	0,74243	0,02708	0,27633	0,01855	0,78980	0,01251	0,68094	0,02753	0,38557	0,04584	0,79800	0,01334
segment0	0,98678	0,00119	0,01213	0,00243	0,99290	0,00142	0,98066	0,00507	0,95106	0,00980	0,99070	0,00201
vehicle0	0,95671	0,00329	0,09740	0,00520	0,93352	0,00380	0,90836	0,00927	0,75206	0,01244	0,89672	0,00945
vehicle1	0,50000	0,00000	0,25650	0,00000	0,85288	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vehicle2	0,89218	0,01793	0,16619	0,01598	0,88732	0,01110	0,77714	0,02376	0,56701	0,04031	0,88219	0,01684
vehicle3	0,50000	0,00000	0,25059	0,00000	0,85676	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vowel0	0,93044	0,02035	0,04112	0,00793	0,97721	0,00443	0,90457	0,03397	0,76398	0,04503	0,96949	0,00766
wisconsin	0,96710	0,00721	0,06878	0,00655	0,94751	0,00512	0,92018	0,00792	0,84754	0,01430	0,95309	0,00684
yeast1	0,73115	0,01674	0,24771	0,00583	0,83356	0,00387	0,62970	0,01388	0,35242	0,01812	0,87242	0,00705
yeast-2_vs_8	0,77391	0,00000	0,02077	0,00000	0,98927	0,00000	0,72831	0,00000	0,65905	0,00000	0,99783	0,00000
yeast3	0,94163	0,00481	0,05108	0,00424	0,97111	0,00236	0,88528	0,01937	0,74919	0,02312	0,96548	0,00315
yeast4	0,51359	0,03040	0,03437	0,00000	0,98251	0,00001	0,01783	0,03986	0,01087	0,02430	0,99972	0,00063
yeast5	0,59332	0,06719	0,02817	0,00181	0,98568	0,00092	0,13423	0,09850	0,10255	0,08216	0,99833	0,00233
yeast6	0,50000	0,00000	0,02358	0,00000	0,98807	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,75439	0,02928	0,27549	0,01924	0,83001	0,01455	0,69623	0,03811	0,14971	0,02887	0,72674	0,01937
abalone19	0,63789	0,05954	0,29722	0,01746	0,82326	0,01277	0,58991	0,07636	0,01292	0,00439	0,70386	0,01802
glass2	0,65417	0,07682	0,33663	0,04728	0,77725	0,03726	0,57224	0,06568	0,11778	0,06124	0,67033	0,04799
haberman	0,51824	0,03787	0,53082	0,02296	0,54156	0,02597	0,49458	0,02248	0,00582	0,03301	0,43022	0,02561
vehicle1	0,55118	0,00815	0,46927	0,01088	0,61923	0,01921	0,54029	0,01354	0,06685	0,01966	0,51612	0,03195
vehicle3	0,56550	0,01742	0,44372	0,02540	0,64714	0,02954	0,56497	0,01396	0,10250	0,02354	0,54545	0,04225
yeast4	0,85583	0,01221	0,18759	0,00912	0,89295	0,00590	0,79764	0,02354	0,17883	0,01232	0,81314	0,01018
yeast6	0,87729	0,01878	0,12319	0,02262	0,93213	0,01383	0,83925	0,02980	0,23877	0,02878	0,87839	0,02348

Apéndice 1
Resumen de resultados de las bases de datos

Resultados para submuestras al tamaño de la clase minoritaria y 120% submuestras

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,50485	0,01084	0,05811	0,00000	0,97007	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
abalone19	0,50000	0,00000	0,00767	0,00000	0,99615	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
ecoli-0_vs_1	0,96875	0,00883	0,03273	0,01132	0,95545	0,01438	0,96834	0,00863	0,92979	0,02349	0,97333	0,00000
ecoli1	0,93904	0,00896	0,09219	0,00913	0,94008	0,00599	0,85823	0,02431	0,73813	0,02751	0,94501	0,01236
ecoli2	0,87989	0,02375	0,07967	0,01383	0,95252	0,00845	0,84671	0,01931	0,70171	0,04632	0,95000	0,01281
ecoli3	0,89563	0,04316	0,08883	0,01230	0,94990	0,00760	0,73660	0,08286	0,53444	0,04836	0,94604	0,02147
ecoli4	0,83963	0,06780	0,04338	0,00805	0,97688	0,00429	0,73909	0,15692	0,58480	0,13455	0,97731	0,00877
glass0	0,84143	0,01740	0,17925	0,01770	0,86488	0,01333	0,79494	0,02785	0,59577	0,04349	0,85576	0,02266
glass-0-1-2-3_vs_4-5-6	0,91524	0,02645	0,08600	0,01346	0,94374	0,00858	0,86966	0,02801	0,75935	0,04000	0,94837	0,01115
glass1	0,73970	0,02298	0,24955	0,00796	0,81070	0,00474	0,69873	0,02411	0,44148	0,02851	0,83735	0,02198
glass2	0,52397	0,03739	0,08219	0,00416	0,95678	0,00262	0,05203	0,07520	0,02630	0,04221	0,99282	0,01000
glass4	0,69154	0,07704	0,05790	0,00530	0,96944	0,00304	0,43068	0,10420	0,31194	0,08182	0,97907	0,01024
glass5	0,75171	0,05734	0,03260	0,00741	0,98313	0,00391	0,43556	0,16722	0,37080	0,16764	0,99122	0,00636
glass6	0,89831	0,03717	0,06357	0,01743	0,96260	0,01042	0,88035	0,05277	0,74365	0,07231	0,95459	0,01896
haberman	0,50000	0,00000	0,26467	0,00000	0,84747	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
iris0	0,98600	0,00548	0,00933	0,00365	0,99317	0,00267	0,98563	0,00562	0,97846	0,00843	1,00000	0,00000
new-thyroid1	0,94198	0,01656	0,06326	0,02293	0,96031	0,01503	0,94011	0,01817	0,80333	0,06142	0,93444	0,02761
new-thyroid2	0,93452	0,01187	0,07070	0,01372	0,95488	0,00933	0,93384	0,01068	0,79124	0,03203	0,92667	0,01730
page-blocks0	0,90750	0,00760	0,05135	0,00246	0,97153	0,00127	0,83190	0,02214	0,70927	0,02160	0,97570	0,00170
pima	0,75723	0,01299	0,26620	0,01365	0,80053	0,01111	0,68254	0,01803	0,40003	0,03057	0,82080	0,01869
segment0	0,98862	0,00108	0,01005	0,00157	0,99413	0,00092	0,98084	0,00352	0,95908	0,00636	0,99353	0,00104
vehicle0	0,95634	0,00131	0,09527	0,00492	0,93492	0,00359	0,91109	0,00944	0,75757	0,01202	0,89796	0,01063
vehicle1	0,50000	0,00000	0,25650	0,00000	0,85288	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vehicle2	0,85910	0,05177	0,17779	0,02402	0,88329	0,01181	0,68790	0,12729	0,49509	0,11514	0,90169	0,02596
vehicle3	0,50000	0,00000	0,25059	0,00000	0,85676	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vowel0	0,91890	0,00362	0,03849	0,00303	0,97877	0,00176	0,89361	0,00799	0,76970	0,01202	0,97485	0,00530
wisconsin	0,97101	0,00688	0,06409	0,00984	0,95110	0,00786	0,92517	0,00965	0,85796	0,02100	0,95763	0,01365
yeast1	0,73568	0,00708	0,24838	0,00470	0,83058	0,00441	0,64618	0,01083	0,36497	0,01132	0,85801	0,01198
yeast-2_vs_8	0,77391	0,00000	0,02077	0,00000	0,98927	0,00000	0,72831	0,00000	0,65905	0,00000	0,99783	0,00000
yeast3	0,93890	0,00725	0,05377	0,00263	0,96970	0,00150	0,86538	0,01104	0,72994	0,01329	0,96715	0,00286
yeast4	0,54680	0,06296	0,03450	0,00074	0,98243	0,00038	0,06825	0,09931	0,04085	0,06551	0,99874	0,00174
yeast5	0,96215	0,04243	0,02183	0,00400	0,98870	0,00209	0,85286	0,06379	0,65359	0,07453	0,98431	0,00301
yeast6	0,54568	0,06402	0,02382	0,00060	0,98819	0,00029	0,06875	0,10486	0,04936	0,07724	0,99931	0,00098

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,75445	0,02859	0,23118	0,02534	0,86169	0,01843	0,70280	0,04500	0,47384	0,02729	0,77519	0,02837
abalone19	0,54951	0,05384	0,19550	0,01675	0,89088	0,01059	0,51033	0,06978	0,01487	0,03025	0,80783	0,01738
glass2	0,69816	0,07514	0,26489	0,03959	0,83583	0,02665	0,60812	0,08570	0,14962	0,06896	0,74985	0,03600
haberman	0,49167	0,02947	0,54839	0,02325	0,52814	0,02867	0,47026	0,02782	-0,03177	0,04345	0,42133	0,03347
vehicle1	0,55594	0,00992	0,46430	0,02066	0,62441	0,02975	0,54413	0,01001	0,07334	0,01335	0,52282	0,04557
vehicle3	0,55159	0,01005	0,45765	0,01371	0,63233	0,02090	0,55381	0,01094	0,08556	0,01428	0,52785	0,03411
yeast4	0,84286	0,03956	0,14851	0,01265	0,91716	0,00787	0,79465	0,03763	0,21482	0,02291	0,85528	0,01327
yeast6	0,86071	0,03340	0,09515	0,01627	0,94976	0,00940	0,83032	0,02562	0,26059	0,03116	0,90808	0,01708

Apéndice 1
Resumen de resultados de las bases de datos

Resultados para submuestras al tamaño de la clase minoritaria y 200% submuestras

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,50000	0,00000	0,05811	0,00000	0,97007	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
abalone19	0,50000	0,00000	0,00767	0,00000	0,99615	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
ecoli-0_vs_1	0,97514	0,00702	0,02455	0,00886	0,96563	0,01129	0,97462	0,00675	0,94660	0,01841	0,97333	0,00000
ecoli1	0,93060	0,01246	0,10286	0,00453	0,93336	0,00275	0,83476	0,01215	0,70466	0,01688	0,94262	0,00320
ecoli2	0,88774	0,02161	0,06957	0,01345	0,95871	0,00785	0,85733	0,03253	0,73485	0,05396	0,95987	0,00735
ecoli3	0,91125	0,01221	0,08932	0,00467	0,95018	0,00268	0,70961	0,04052	0,50643	0,03773	0,95415	0,00865
ecoli4	0,87005	0,07650	0,04339	0,00575	0,97683	0,00305	0,77090	0,10746	0,60805	0,08123	0,97601	0,00368
glass0	0,82907	0,00979	0,18419	0,00797	0,86057	0,00767	0,79176	0,01347	0,58717	0,01840	0,85133	0,02229
glass-0-1-2-3_vs_4-5-6	0,88123	0,03466	0,09623	0,01350	0,93670	0,00889	0,85908	0,02197	0,73423	0,03837	0,93867	0,00973
glass1	0,74950	0,01837	0,24370	0,02140	0,81362	0,01620	0,71126	0,03415	0,45829	0,05230	0,83201	0,02764
glass2	0,50808	0,01806	0,08219	0,00624	0,95694	0,00377	0,01895	0,04237	0,00530	0,01186	0,99590	0,00917
glass4	0,68347	0,08877	0,05792	0,00430	0,96931	0,00253	0,42713	0,17193	0,31413	0,11388	0,97905	0,01382
glass5	0,84780	0,09022	0,03169	0,00695	0,98321	0,00356	0,69754	0,18971	0,55637	0,18568	0,97854	0,00267
glass6	0,89267	0,04182	0,06064	0,01191	0,96451	0,00725	0,87381	0,02079	0,74990	0,04416	0,96108	0,01399
haberman	0,50000	0,00000	0,26467	0,00000	0,84747	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
iris0	0,98200	0,00447	0,01200	0,00298	0,99122	0,00218	0,98153	0,00459	0,97231	0,00688	1,00000	0,00000
new-thyroid1	0,94286	0,00796	0,06419	0,01449	0,95967	0,00950	0,93950	0,00991	0,80167	0,03934	0,93333	0,02115
new-thyroid2	0,93587	0,01783	0,07814	0,02142	0,95004	0,01424	0,93408	0,01722	0,77110	0,05387	0,91556	0,02558
page-blocks0	0,90802	0,00296	0,05285	0,00300	0,97068	0,00160	0,82918	0,01976	0,70240	0,02236	0,97456	0,00111
pima	0,75572	0,01835	0,26227	0,01880	0,80518	0,01380	0,68088	0,02361	0,40451	0,04341	0,83400	0,01334
segment0	0,98815	0,00097	0,00815	0,00103	0,99524	0,00061	0,98605	0,00221	0,96693	0,00409	0,99414	0,00145
vehicle0	0,93745	0,03986	0,10048	0,00772	0,93247	0,00354	0,87460	0,00712	0,72771	0,05476	0,90258	0,01630
vehicle1	0,50000	0,00000	0,25650	0,00000	0,85288	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vehicle2	0,90533	0,00676	0,15528	0,01343	0,89553	0,01101	0,77945	0,02382	0,58791	0,02655	0,89916	0,02902
vehicle3	0,50000	0,00000	0,25059	0,00000	0,85676	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vowel0	0,93722	0,01863	0,03504	0,00663	0,98059	0,00372	0,91890	0,01829	0,79878	0,03450	0,97417	0,00670
wisconsin	0,97610	0,00495	0,05443	0,00843	0,95852	0,00645	0,93686	0,01127	0,87927	0,01878	0,96304	0,01219
yeast1	0,73475	0,00825	0,24394	0,00721	0,83614	0,00514	0,63342	0,01070	0,36155	0,01802	0,87640	0,00800
yeast-2_vs_8	0,77391	0,00000	0,02077	0,00000	0,98927	0,00000	0,72831	0,00000	0,65905	0,00000	0,99783	0,00000
yeast3	0,92722	0,01451	0,05647	0,00199	0,96822	0,00112	0,85467	0,00826	0,71392	0,01096	0,96624	0,00157
yeast4	0,57758	0,04978	0,03531	0,00090	0,98200	0,00047	0,08694	0,06914	0,04319	0,04034	0,99776	0,00151
yeast5	0,98491	0,00531	0,02088	0,00048	0,98916	0,00026	0,89949	0,01858	0,69500	0,01153	0,98361	0,00105
yeast6	0,62373	0,13236	0,02345	0,00263	0,98809	0,00133	0,21292	0,23560	0,15099	0,18153	0,99655	0,00331

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,77258	0,02371	0,21377	0,00804	0,87412	0,00543	0,71288	0,03520	0,19439	0,03049	0,79373	0,00820
abalone19	0,60906	0,03219	0,18356	0,02011	0,89831	0,01210	0,52630	0,07724	0,01833	0,00704	0,81956	0,02032
glass2	0,66478	0,08589	0,25198	0,01932	0,84567	0,01271	0,60044	0,08285	0,15524	0,07039	0,76828	0,01768
haberman	0,49731	0,02699	0,53483	0,02874	0,54930	0,03548	0,47276	0,02096	-0,02970	0,03379	0,44689	0,04263
vehicle1	0,55616	0,01481	0,44515	0,01642	0,64953	0,02263	0,54888	0,01500	0,08157	0,01811	0,55713	0,03685
vehicle3	0,55414	0,00550	0,43400	0,01626	0,66090	0,02040	0,56322	0,00583	0,10149	0,00903	0,56724	0,03108
yeast4	0,83811	0,04055	0,14016	0,01081	0,92229	0,00658	0,77654	0,02310	0,21568	0,02070	0,86532	0,01090
yeast6	0,85044	0,03514	0,09366	0,00767	0,94964	0,00438	0,83527	0,02340	0,26161	0,02202	0,90946	0,00783

Apéndice 1
Resumen de resultados de las bases de datos

Resultados para submuestras al tamaño de la clase minoritaria y 400% submuestras

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,50000	0,00000	0,05811	0,00000	0,97007	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
abalone19	0,50000	0,00000	0,00767	0,00000	0,99615	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
ecoli-0_vs_1	0,97997	0,00035	0,01818	0,00000	0,97371	0,00000	0,97947	0,00000	0,95980	0,00000	0,97333	0,00000
ecoli1	0,93271	0,01508	0,09695	0,01022	0,93677	0,00674	0,85427	0,01432	0,72671	0,02791	0,93955	0,01207
ecoli2	0,88400	0,02967	0,06966	0,00543	0,95915	0,00321	0,83862	0,01445	0,72170	0,02156	0,96685	0,00320
ecoli3	0,90714	0,02112	0,08935	0,00839	0,95003	0,00489	0,72645	0,03720	0,51835	0,04090	0,95077	0,01036
ecoli4	0,83749	0,05785	0,04040	0,01099	0,97843	0,00604	0,79925	0,05609	0,64295	0,07218	0,97793	0,00984
glass0	0,83198	0,01866	0,17081	0,01363	0,87283	0,01030	0,79959	0,01981	0,61045	0,03180	0,87094	0,01642
glass-0-1-2-3_vs_4-5-6	0,88657	0,02606	0,09344	0,01370	0,93823	0,00945	0,86632	0,02147	0,74365	0,03471	0,93723	0,01463
glass1	0,73527	0,03196	0,25118	0,03266	0,80843	0,02371	0,70523	0,04747	0,44145	0,07804	0,82349	0,02658
glass2	0,52051	0,02952	0,08033	0,00208	0,95800	0,00128	0,04228	0,05819	0,02350	0,03366	0,99692	0,00459
glass4	0,82545	0,07243	0,05050	0,01739	0,97301	0,00939	0,65399	0,11784	0,49230	0,18548	0,97207	0,01566
glass5	0,92683	0,05125	0,02516	0,00417	0,98662	0,00231	0,81854	0,06224	0,68626	0,05291	0,98049	0,00597
glass6	0,89047	0,03794	0,05787	0,01472	0,96649	0,00843	0,86084	0,03929	0,74905	0,07062	0,96865	0,00967
haberman	0,50000	0,00000	0,26467	0,00000	0,84747	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
iris0	0,98800	0,00447	0,00800	0,00298	0,99415	0,00218	0,98768	0,00459	0,98154	0,00688	1,00000	0,00000
new-thyroid1	0,92722	0,01717	0,07163	0,01456	0,95527	0,00966	0,92830	0,01295	0,77533	0,03734	0,92778	0,02079
new-thyroid2	0,92413	0,02078	0,07721	0,01759	0,95070	0,01099	0,92320	0,02099	0,77134	0,05526	0,92222	0,01620
page-blocks0	0,90620	0,00445	0,05172	0,00284	0,97128	0,00152	0,83732	0,01756	0,71104	0,02038	0,97419	0,00075
pima	0,74541	0,01309	0,26745	0,00730	0,79896	0,00678	0,68342	0,00706	0,39914	0,01403	0,81760	0,01337
segment0	0,98970	0,00102	0,00832	0,00169	0,99514	0,00099	0,98440	0,00199	0,96613	0,00666	0,99454	0,00187
vehicle0	0,96016	0,00323	0,08367	0,00594	0,94363	0,00443	0,91468	0,00568	0,78139	0,01252	0,91713	0,01242
vehicle1	0,50000	0,00000	0,25650	0,00000	0,85288	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vehicle2	0,90709	0,00821	0,15506	0,01680	0,89615	0,01131	0,77673	0,02925	0,58548	0,04732	0,90169	0,01315
vehicle3	0,50000	0,00000	0,25059	0,00000	0,85676	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vowel0	0,95296	0,00477	0,02977	0,00209	0,98348	0,00118	0,94365	0,00939	0,83134	0,01102	0,97551	0,00248
wisconsin	0,96832	0,00631	0,06293	0,00557	0,95206	0,00438	0,92667	0,00658	0,86037	0,01213	0,95806	0,00963
yeast1	0,73570	0,01066	0,25027	0,00928	0,82944	0,00729	0,64386	0,01264	0,35998	0,02076	0,85649	0,01301
yeast-2_vs_8	0,77391	0,00000	0,02077	0,00000	0,98927	0,00000	0,72831	0,00000	0,65905	0,00000	0,99783	0,00000
yeast3	0,92272	0,01543	0,05444	0,00400	0,96937	0,00223	0,85589	0,01847	0,72341	0,02265	0,96866	0,00377
yeast4	0,61332	0,12584	0,03356	0,00146	0,98288	0,00071	0,16466	0,18783	0,11370	0,13330	0,99762	0,00245
yeast5	0,98039	0,00639	0,02102	0,00351	0,98911	0,00182	0,88409	0,03838	0,68530	0,05591	0,98444	0,00244
yeast6	0,66698	0,03322	0,02305	0,00249	0,98827	0,00127	0,32435	0,09362	0,23052	0,08333	0,99448	0,00189

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,72511	0,03576	0,18743	0,01392	0,89190	0,00882	0,68021	0,01766	0,19597	0,01640	0,82607	0,01511
abalone19	0,55597	0,03962	0,16905	0,01331	0,90706	0,00813	0,43979	0,04324	0,01332	0,00407	0,83491	0,01348
glass2	0,66758	0,06463	0,23001	0,03290	0,86202	0,02314	0,57718	0,12880	0,15220	0,07505	0,79503	0,03927
haberman	0,48676	0,02412	0,53738	0,01261	0,55013	0,01265	0,46537	0,01927	-0,04273	0,03375	0,45422	0,01584
vehicle1	0,56533	0,00483	0,42504	0,01275	0,67279	0,01769	0,55738	0,01070	0,09790	0,00930	0,58990	0,03147
vehicle3	0,56084	0,00236	0,41629	0,02041	0,68304	0,02412	0,56332	0,00645	0,10742	0,00788	0,60133	0,04085
yeast4	0,81490	0,02335	0,12075	0,00559	0,93403	0,00321	0,76222	0,00720	0,23291	0,01177	0,88696	0,00569
yeast6	0,84845	0,03490	0,08490	0,00499	0,95469	0,00277	0,83312	0,02683	0,27564	0,02331	0,91871	0,00440

Apéndice 1
Resumen de resultados de las bases de datos

Resultados para submuestras al tamaño de la clase minoritaria y 600% submuestras

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,51182	0,01684	0,05914	0,00230	0,96951	0,00126	0,01742	0,03895	0,00583	0,01304	0,99855	0,00324
abalone19	0,50000	0,00000	0,00767	0,00000	0,99615	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
ecoli-0_vs_1	0,97818	0,00499	0,02091	0,00610	0,97032	0,00759	0,97738	0,00468	0,95419	0,01253	0,97333	0,00000
ecoli1	0,93765	0,01189	0,09932	0,01125	0,93568	0,00779	0,84334	0,01439	0,71577	0,02764	0,94342	0,01781
ecoli2	0,88505	0,02511	0,06605	0,00443	0,96100	0,00258	0,86300	0,01655	0,74478	0,01875	0,96337	0,00473
ecoli3	0,90036	0,03409	0,08284	0,00239	0,95393	0,00133	0,72379	0,02533	0,53835	0,02246	0,95940	0,00496
ecoli4	0,85401	0,07743	0,03739	0,01177	0,97999	0,00635	0,82793	0,07913	0,67945	0,10310	0,97794	0,00887
glass0	0,83501	0,02014	0,17654	0,01547	0,86715	0,01290	0,73890	0,01479	0,60252	0,03152	0,86113	0,01780
glass-0-1-2-3_vs_4-5-6	0,89479	0,01975	0,08321	0,01025	0,94521	0,00690	0,88030	0,01701	0,77079	0,02792	0,94580	0,01187
glass1	0,74619	0,02147	0,24771	0,01723	0,81279	0,01264	0,69859	0,02348	0,44430	0,04022	0,84444	0,01923
glass2	0,54529	0,02835	0,08312	0,00389	0,95634	0,00237	0,05818	0,05311	0,02483	0,02304	0,99282	0,00688
glass4	0,83787	0,06914	0,04483	0,00977	0,97606	0,00546	0,66941	0,14163	0,52837	0,10426	0,97602	0,01386
glass5	0,92341	0,05129	0,02981	0,00420	0,98416	0,00213	0,76582	0,14418	0,62261	0,13396	0,97854	0,00267
glass6	0,85607	0,05060	0,05415	0,00716	0,96884	0,00393	0,85175	0,03633	0,75752	0,04154	0,97622	0,00296
haberman	0,50000	0,00000	0,26467	0,00000	0,84747	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
iris0	0,98600	0,00548	0,00933	0,00365	0,99317	0,00267	0,98563	0,00562	0,97846	0,00843	1,00000	0,00000
new-thyroid1	0,93413	0,02202	0,06977	0,01918	0,95661	0,01210	0,92763	0,01144	0,77917	0,05659	0,93111	0,02377
new-thyroid2	0,94024	0,01266	0,06233	0,01340	0,96008	0,00914	0,93886	0,01204	0,81745	0,03092	0,93667	0,01868
page-blocks0	0,89800	0,00931	0,05296	0,00349	0,97064	0,00189	0,82657	0,01937	0,70028	0,02382	0,97492	0,00155
pima	0,73993	0,01507	0,27242	0,00852	0,79489	0,00939	0,67914	0,00385	0,38895	0,01178	0,81160	0,02100
segment0	0,98970	0,00106	0,00667	0,00104	0,99610	0,00061	0,98769	0,00154	0,97286	0,00422	0,99555	0,00110
vehicle0	0,95634	0,00309	0,09875	0,00694	0,94156	0,00472	0,90949	0,00863	0,77327	0,01781	0,91589	0,00625
vehicle1	0,50000	0,00000	0,25650	0,00000	0,85288	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vehicle2	0,90820	0,00675	0,15458	0,00897	0,89649	0,00638	0,77737	0,02188	0,58702	0,02648	0,90266	0,01639
vehicle3	0,50000	0,00000	0,25059	0,00000	0,85676	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vowel0	0,93232	0,02130	0,03222	0,00643	0,98222	0,00357	0,91799	0,01880	0,80922	0,03577	0,97772	0,00439
wisconsin	0,97157	0,00566	0,06703	0,00838	0,94874	0,00660	0,92329	0,01018	0,85172	0,01818	0,95176	0,00921
yeast1	0,71987	0,00960	0,24946	0,00544	0,83103	0,00439	0,63651	0,01123	0,35580	0,01363	0,86445	0,01066
yeast-2_vs_8	0,77391	0,00000	0,02077	0,00000	0,98927	0,00000	0,72831	0,00000	0,65905	0,00000	0,99783	0,00000
yeast3	0,92984	0,00879	0,05552	0,00311	0,96876	0,00172	0,85565	0,01261	0,71705	0,01806	0,96699	0,00137
yeast4	0,61831	0,04801	0,03531	0,00162	0,98198	0,00086	0,15536	0,06894	0,08520	0,04160	0,99665	0,00268
yeast5	0,97330	0,01897	0,02075	0,00210	0,98925	0,00110	0,88161	0,00897	0,68643	0,02299	0,98486	0,00186
yeast6	0,75440	0,08789	0,02251	0,00162	0,98853	0,00083	0,44262	0,12429	0,31863	0,09199	0,99324	0,00179

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,74552	0,03866	0,18949	0,01267	0,89831	0,00880	0,70119	0,03657	0,20884	0,02651	0,82241	0,01343
abalone19	0,54550	0,03158	0,14816	0,00825	0,91945	0,00491	0,44444	0,05897	0,01522	0,00689	0,85616	0,00796
glass2	0,69335	0,07409	0,19296	0,05272	0,88525	0,03599	0,60455	0,08063	0,22151	0,07966	0,83603	0,05863
haberman	0,50574	0,02423	0,51575	0,02616	0,57449	0,03495	0,47593	0,01293	-0,02235	0,02285	0,49444	0,04828
vehicle1	0,56212	0,00231	0,42362	0,01443	0,67403	0,02019	0,55843	0,01166	0,10006	0,00746	0,59148	0,03622
vehicle3	0,56281	0,00396	0,41204	0,02144	0,68676	0,02377	0,56911	0,00715	0,11557	0,01579	0,60476	0,03762
yeast4	0,81914	0,02866	0,12156	0,01014	0,93337	0,00597	0,76523	0,02735	0,23583	0,01818	0,88585	0,01170
yeast6	0,85465	0,02106	0,07870	0,01070	0,95804	0,00609	0,83983	0,00977	0,30054	0,02978	0,92492	0,01119

Apéndice 1
Resumen de resultados de las bases de datos

Resultados para submuestras al tamaño de la clase minoritaria y 200 submuestras

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,51655	0,01551	0,05983	0,00242	0,96913	0,00130	0,03510	0,04806	0,01432	0,02370	0,99746	0,00275
abalone19	0,50000	0,00000	0,00767	0,00000	0,99615	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
ecoli-0_vs_1	0,98034	0,00000	0,01818	0,00000	0,97371	0,00000	0,97947	0,00000	0,95980	0,00000	0,97333	0,00000
ecoli1	0,85116	0,02703	0,12081	0,00581	0,92133	0,00377	0,81958	0,01656	0,65787	0,02002	0,92184	0,00884
ecoli2	0,85804	0,02355	0,06958	0,00339	0,95900	0,00186	0,85109	0,01568	0,72793	0,01749	0,96261	0,00190
ecoli3	0,91070	0,02544	0,08935	0,00210	0,94961	0,00129	0,76983	0,00961	0,54643	0,01018	0,94219	0,00377
ecoli4	0,84929	0,06259	0,03151	0,00496	0,98325	0,00270	0,83818	0,03611	0,71003	0,03831	0,98358	0,00686
glass0	0,80743	0,02939	0,17480	0,01519	0,87067	0,01151	0,79269	0,01890	0,59948	0,03419	0,87355	0,01353
glass-0-1-2-3_vs_4-5-6	0,85929	0,01499	0,09637	0,00422	0,93692	0,00289	0,85320	0,01018	0,73103	0,01131	0,94326	0,00517
glass1	0,72497	0,02729	0,24487	0,02023	0,81249	0,01375	0,71458	0,03484	0,45731	0,05141	0,82487	0,02776
glass2	0,54487	0,05250	0,08312	0,00389	0,95641	0,00223	0,04257	0,05865	0,02294	0,04020	0,98385	0,00562
glass4	0,87305	0,07529	0,04957	0,00793	0,97332	0,00450	0,71972	0,13128	0,54757	0,08431	0,96905	0,01336
glass5	0,94634	0,05352	0,02700	0,00390	0,98555	0,00200	0,89606	0,10176	0,73095	0,10140	0,97561	0,00000
glass6	0,84874	0,05479	0,06064	0,00579	0,96508	0,00321	0,83212	0,03075	0,72919	0,03356	0,97405	0,00452
haberman	0,50000	0,00000	0,26467	0,00000	0,84747	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
iris0	0,98200	0,00447	0,01200	0,00298	0,99122	0,00218	0,98153	0,00459	0,97231	0,00688	1,00000	0,00000
new-thyroid1	0,93198	0,00232	0,07535	0,00389	0,95305	0,00251	0,93147	0,00226	0,76358	0,01087	0,92111	0,00465
new-thyroid2	0,93143	0,00628	0,07349	0,00606	0,95285	0,00396	0,92997	0,00675	0,78600	0,01650	0,92444	0,00745
page-blocks0	0,89905	0,00358	0,05234	0,00295	0,97099	0,00159	0,82764	0,01794	0,70365	0,02129	0,97549	0,00112
pima	0,70319	0,01021	0,26693	0,00514	0,79862	0,00442	0,68678	0,00590	0,40245	0,01079	0,81480	0,00879
segment0	0,99012	0,00164	0,00511	0,00112	0,99702	0,00065	0,98911	0,00228	0,97906	0,00458	0,99717	0,00077
vehicle0	0,95847	0,00254	0,06972	0,00302	0,95338	0,00213	0,92724	0,00123	0,81531	0,00707	0,93261	0,00472
vehicle1	0,50000	0,00000	0,25650	0,00000	0,85288	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vehicle2	0,91958	0,00502	0,13662	0,00828	0,90780	0,00543	0,81142	0,01638	0,64097	0,02428	0,90908	0,00815
vehicle3	0,50000	0,00000	0,25059	0,00000	0,85676	0,00000	0,00000	0,00000	0,00000	0,00000	1,00000	0,00000
vowel0	0,95119	0,00267	0,02937	0,00268	0,98374	0,00150	0,94017	0,00565	0,83026	0,01443	0,97662	0,00261
wisconsin	0,96890	0,00226	0,06527	0,00131	0,95021	0,00075	0,92513	0,00439	0,85536	0,00357	0,95400	0,00492
yeast1	0,69984	0,00918	0,25054	0,00681	0,82973	0,00501	0,64009	0,01033	0,35655	0,01652	0,85953	0,00867
yeast-2_vs_8	0,77391	0,00000	0,02077	0,00000	0,98927	0,00000	0,72831	0,00000	0,65905	0,00000	0,99783	0,00000
yeast3	0,90776	0,01514	0,05431	0,00227	0,96942	0,00128	0,86162	0,01189	0,72583	0,01199	0,96745	0,00278
yeast4	0,64503	0,07699	0,03518	0,00111	0,98202	0,00057	0,21907	0,12061	0,13011	0,07769	0,99525	0,00181
yeast5	0,97724	0,00898	0,02048	0,00077	0,98940	0,00040	0,86469	0,00878	0,67958	0,01368	0,98611	0,00085
yeast6	0,64232	0,08575	0,02264	0,00102	0,98849	0,00051	0,29457	0,17013	0,21609	0,12208	0,99544	0,00287

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,75559	0,03661	0,18603	0,00632	0,89336	0,00396	0,71346	0,02439	0,21870	0,01957	0,82569	0,00561
abalone19	0,56872	0,02718	0,18055	0,00972	0,90018	0,00593	0,44564	0,07420	0,01243	0,00501	0,82332	0,01015
glass2	0,75967	0,05615	0,13105	0,01553	0,92616	0,00891	0,63153	0,10591	0,32023	0,07497	0,90146	0,01203
haberman	0,50288	0,01171	0,54722	0,01581	0,53857	0,02042	0,45640	0,00757	-0,05405	0,01449	0,43911	0,02711
vehicle1	0,56561	0,00610	0,41797	0,01805	0,67935	0,02282	0,56391	0,00985	0,10906	0,00974	0,59784	0,03828
vehicle3	0,56663	0,00660	0,41536	0,02406	0,68238	0,02646	0,56911	0,00310	0,11508	0,01614	0,59818	0,04217
yeast4	0,80738	0,03211	0,11644	0,00991	0,98643	0,00567	0,77013	0,02178	0,24888	0,02303	0,89115	0,01094
yeast6	0,84670	0,03749	0,07843	0,00334	0,95832	0,00189	0,82104	0,02450	0,28510	0,01146	0,92603	0,00411

Resultados para SMOTE TamSizeOfMajClassSub5

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,65558	0,05503	0,11844	0,01242	0,93526	0,00706	0,57925	0,04879	0,21927	0,05838	0,91203	0,00988
abalone19	0,56483	0,03231	0,02889	0,00165	0,98533	0,00085	0,09455	0,03416	0,00907	0,00762	0,97832	0,00163
ecoli-0_vs_1	0,96903	0,00492	0,02909	0,00996	0,95997	0,01198	0,97081	0,00792	0,93709	0,02021	0,97333	0,00000
ecoli1	0,88291	0,02005	0,10888	0,00580	0,92739	0,00422	0,86877	0,00949	0,70888	0,01341	0,90864	0,01006
ecoli2	0,87003	0,01996	0,09749	0,01321	0,94081	0,00808	0,86275	0,02846	0,66404	0,04397	0,91911	0,00896
ecoli3	0,85762	0,03159	0,11428	0,00617	0,93370	0,00330	0,82375	0,03994	0,51841	0,04232	0,90034	0,00470
ecoli4	0,85812	0,02314	0,04759	0,00668	0,97440	0,00365	0,84155	0,02703	0,62468	0,04734	0,96522	0,00596
glass0	0,81473	0,01270	0,22627	0,01843	0,81797	0,01920	0,77625	0,01809	0,52195	0,03216	0,76374	0,02949
glass-0-1-2-3_vs_4-5-6	0,90230	0,01868	0,08605	0,00625	0,94194	0,00439	0,89879	0,01215	0,77467	0,01629	0,92519	0,00806
glass1	0,74626	0,00897	0,27477	0,02091	0,77784	0,02107	0,70616	0,02400	0,41638	0,04071	0,75788	0,03958
glass2	0,77685	0,01943	0,16104	0,02437	0,90695	0,01569	0,61563	0,04979	0,28468	0,02915	0,86877	0,02812
glass4	0,83067	0,06238	0,06281	0,00704	0,96558	0,00378	0,81112	0,09202	0,55860	0,07752	0,94700	0,00570
glass5	0,89634	0,08036	0,04016	0,00416	0,97883	0,00224	0,75823	0,07315	0,55487	0,05183	0,97171	0,00534
glass6	0,89789	0,05792	0,07214	0,01221	0,95756	0,00704	0,87442	0,04658	0,71373	0,05673	0,94595	0,00382
haberman	0,61547	0,02860	0,32391	0,02549	0,76117	0,02887	0,61112	0,02588	0,24113	0,03627	0,72356	0,02988
iris0	0,98800	0,00447	0,00800	0,00298	0,99415	0,00218	0,98768	0,00459	0,98154	0,00688	1,00000	0,00000
new-thyroid1	0,97238	0,01153	0,01953	0,00764	0,98828	0,00466	0,96892	0,01029	0,92939	0,02590	0,98556	0,00745
new-thyroid2	0,94937	0,01830	0,04093	0,01843	0,97395	0,01247	0,94688	0,01774	0,87595	0,04384	0,96444	0,02277
page-blocks0	0,90731	0,00550	0,14777	0,01037	0,91064	0,00677	0,88987	0,00281	0,49787	0,02087	0,84213	0,01249
pima	0,73623	0,02687	0,29217	0,01493	0,75981	0,01145	0,70329	0,01970	0,39031	0,03456	0,71381	0,01658
segment0	0,99416	0,00137	0,00581	0,00049	0,99661	0,00029	0,99100	0,00150	0,97640	0,00200	0,99545	0,00080
vehicle0	0,94864	0,00300	0,10661	0,00269	0,92569	0,00217	0,91806	0,00077	0,73972	0,00473	0,87016	0,00513
vehicle1	0,54893	0,02826	0,61389	0,05931	#DIV/0!	#DIV/0!	0,21269	0,10843	0,05788	0,04079	0,22667	0,09466
vehicle2	0,88785	0,00120	0,18012	0,00558	0,86628	0,00454	0,84023	0,00552	0,59660	0,01137	0,79453	0,00837
vehicle3	0,54155	0,04031	0,61911	0,09132	#DIV/0!	#DIV/0!	0,19733	0,12651	0,06855	0,06937	0,22079	0,14469
vowel0	0,95797	0,01398	0,02612	0,00513	0,98552	0,00285	0,94924	0,02146	0,85103	0,02991	0,97884	0,00539
wisconsin	0,96348	0,01118	0,06323	0,00334	0,91442	0,00437	0,94192	0,00396	0,86456	0,00700	0,96154	0,01305
yeast1	0,72612	0,01466	0,28544	0,00658	0,78533	0,00597	0,69538	0,00663	0,36361	0,01183	0,73706	0,01156
yeast1_vs_8	0,83214	0,01003	0,05436	0,00308	0,97112	0,00170	0,78994	0,01382	0,48346	0,01338	0,95758	0,00328
yeast3	0,94283	0,00542	0,06306	0,00703	0,96388	0,00415	0,89855	0,00924	0,71513	0,02627	0,94717	0,00774
yeast4	0,77640	0,06198	0,06362	0,00468	0,96655	0,00251	0,66175	0,04384	0,30545	0,03756	0,95310	0,00494
yeast5	0,94226	0,01766	0,02385	0,00182	0,98760	0,00094	0,91289	0,01064	0,67319	0,02722	0,97986	0,00147
yeast6	0,83288	0,02014	0,03868	0,00155	0,97998	0,00083	0,75245	0,02493	0,40098	0,01266	0,96991	0,00221

Resultados para SMOTE TamSizeOfMaxClassSub5 (*imbalanced*)

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,69220	0,02709	0,24626	0,02778	0,85213	0,02058	0,65875	0,03025	0,13833	0,01471	0,76336	0,03305
abalone19	0,68996	0,02218	0,28036	0,02595	0,83532	0,01747	0,61067	0,02913	0,01534	0,00265	0,72095	0,02639
ecoli-0_vs_1	0,97342	0,00746	0,02636	0,00985	0,96321	0,01267	0,97316	0,00756	0,94273	0,02055	0,97333	0,00000
ecoli1	0,92011	0,02651	0,11184	0,01161	0,92395	0,00829	0,88124	0,01327	0,71229	0,02772	0,89314	0,01214
ecoli2	0,88038	0,01935	0,09758	0,01449	0,94069	0,00961	0,85103	0,01667	0,66293	0,03523	0,92392	0,01410
ecoli3	0,89536	0,01660	0,12622	0,01004	0,92498	0,00642	0,86814	0,01313	0,52822	0,02247	0,87442	0,01335
ecoli4	0,90353	0,00946	0,09277	0,02421	0,94802	0,01436	0,83915	0,03517	0,49118	0,07775	0,91528	0,02393
glass0	0,83441	0,02892	0,17841	0,02022	0,86005	0,01785	0,81678	0,01836	0,61390	0,03932	0,82512	0,02601
glass-0-1-2-3_vs_4-5-6	0,90269	0,02353	0,09260	0,00779	0,93741	0,00554	0,88940	0,01550	0,75750	0,02156	0,92015	0,01245
glass1	0,75955	0,02277	0,26888	0,01422	0,78040	0,01198	0,71783	0,01645	0,43335	0,03042	0,75360	0,02105
glass2	0,70894	0,04467	0,28155	0,04817	0,82214	0,03812	0,57071	0,09850	0,14030	0,04876	0,73167	0,05463
glass4	0,88319	0,02736	0,11984	0,02436	0,93122	0,01492	0,83592	0,02658	0,42704	0,06564	0,88437	0,02385
glass5	0,90780	0,06889	0,14310	0,04275	0,91752	0,02595	0,87354	0,11593	0,35456	0,11830	0,85366	0,04322
glass6	0,90323	0,04167	0,08689	0,01846	0,94742	0,01247	0,88971	0,01687	0,69295	0,03797	0,92108	0,02824
haberman	0,58054	0,04416	0,42392	0,09420	#¡DIV/0!	#¡DIV/0!	0,45921	0,14790	0,17466	0,07155	0,55289	0,15532
iris0	0,98400	0,00548	0,01067	0,00365	0,99220	0,00267	0,98358	0,00562	0,97538	0,00843	1,00000	0,00000
new-thyroid1	0,95048	0,00998	0,04372	0,01493	0,97314	0,00962	0,93958	0,01412	0,85252	0,04300	0,96333	0,01601
new-thyroid2	0,94294	0,01461	0,06977	0,01356	0,95464	0,00967	0,93572	0,01411	0,79957	0,03201	0,92667	0,01817
page-blocks0	0,93163	0,00409	0,11539	0,00660	0,93185	0,00426	0,90007	0,00355	0,56270	0,01454	0,88056	0,00764
pima	0,73210	0,01593	0,29608	0,01297	0,75779	0,01075	0,69818	0,01370	0,38035	0,02689	0,71342	0,01315
segment0	0,99088	0,00162	0,01101	0,00175	0,99355	0,00103	0,98771	0,00247	0,95603	0,00683	0,98949	0,00194
vehicle0	0,95047	0,00181	0,10519	0,00118	0,92683	0,00087	0,91846	0,00147	0,74225	0,00266	0,87264	0,00139
vehicle1	0,55061	0,02106	0,62425	0,03986	#¡DIV/0!	#¡DIV/0!	0,19561	0,07568	0,06263	0,03362	0,20190	0,05923
vehicle2	0,89199	0,01123	0,18365	0,01340	0,86388	0,01086	0,83726	0,01095	0,58647	0,02578	0,79108	0,01715
vehicle3	0,56453	0,01736	0,59805	0,03543	#¡DIV/0!	#¡DIV/0!	0,23633	0,06150	0,09673	0,02637	0,24441	0,05484
vowel0	0,95479	0,01517	0,04273	0,00535	0,97603	0,00311	0,94284	0,01692	0,77662	0,02442	0,95991	0,00468
wisconsin	0,96254	0,01141	0,06178	0,00318	0,91631	0,00470	0,94304	0,00486	0,86760	0,00709	0,96152	0,01157
yeast1	0,72856	0,01263	0,28775	0,01036	0,78330	0,00877	0,69545	0,01284	0,36075	0,02153	0,73308	0,01092
yeast-2_vs_8	0,82346	0,02445	0,10754	0,02719	0,94075	0,01640	0,75462	0,04074	0,30969	0,03718	0,90295	0,02975
yeast3	0,95452	0,00437	0,07990	0,00736	0,95333	0,00450	0,91995	0,00414	0,67684	0,02216	0,92006	0,00858
yeast4	0,83875	0,02293	0,17009	0,00716	0,90378	0,00432	0,79024	0,02236	0,19241	0,01911	0,83252	0,00743
yeast5	0,95016	0,01755	0,05322	0,00711	0,97174	0,00391	0,95081	0,01057	0,50583	0,03269	0,94639	0,00779
yeast6	0,84415	0,02741	0,09677	0,01195	0,94781	0,00684	0,83681	0,02046	0,26216	0,02326	0,90614	0,01296

Resultados para SMOTE TamSizeOfMinClassSub5 (*imbalanced*)

	AUC		Error		F value		GM		Kappa		TPR	
	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica	Media	Desviacion Tipica
abalone9-18	0,71120	0,04046	0,23916	0,01344	0,85714	0,00993	0,65827	0,02847	0,14171	0,01119	0,77092	0,01659
abalone19	0,67121	0,02375	0,35156	0,02692	0,78344	0,02114	0,58806	0,04735	0,09892	0,00292	0,64892	0,02791
ecoli-0_vs_1	0,97758	0,00305	0,02273	0,00455	0,96831	0,00513	0,97588	0,00343	0,95070	0,00894	0,97333	0,00943
ecoli1	0,92285	0,00718	0,12730	0,00825	0,91133	0,00673	0,88136	0,00820	0,68564	0,01477	0,86294	0,01361
ecoli2	0,88895	0,02215	0,12366	0,01844	0,92244	0,01232	0,85333	0,01424	0,61418	0,04220	0,88519	0,02822
ecoli3	0,88933	0,00996	0,15056	0,01616	0,90904	0,01045	0,85212	0,02874	0,47675	0,03886	0,84791	0,01710
ecoli4	0,90344	0,04451	0,11435	0,00338	0,93461	0,00215	0,87758	0,04239	0,46095	0,03092	0,88599	0,00316
glass0	0,81425	0,01151	0,22720	0,01234	0,81817	0,01001	0,76968	0,01571	0,51628	0,02654	0,76906	0,01149
glass-0-1-2-3_vs_4-5-6	0,89983	0,02171	0,10467	0,01459	0,92854	0,01028	0,88589	0,02148	0,73225	0,03705	0,90174	0,02078
glass1	0,73818	0,03100	0,28955	0,02091	0,76258	0,01726	0,69953	0,02561	0,39149	0,04577	0,72582	0,01618
glass2	0,59781	0,06756	0,40062	0,04105	0,72651	0,03730	0,54028	0,06406	0,06295	0,03579	0,60362	0,04615
glass4	0,81541	0,03013	0,17763	0,04969	0,89272	0,03425	0,79605	0,03745	0,35555	0,05598	0,82402	0,05499
glass5	0,94195	0,01026	0,13364	0,01762	0,92317	0,01187	0,92669	0,01056	0,38460	0,02223	0,86049	0,01845
glass6	0,88921	0,03509	0,08992	0,01837	0,94513	0,01255	0,89153	0,01775	0,69130	0,04335	0,91676	0,02182
haberman	0,51192	0,01316	0,61074	0,06731	#¡DIV/0!	#¡DIV/0!	0,17283	0,07199	0,05018	0,04065	0,23467	0,11798
iris0	0,98400	0,00548	0,01067	0,00365	0,99220	0,00267	0,98358	0,00562	0,97538	0,00843	1,00000	0,00000
new-thyroid1	0,92524	0,01638	0,08000	0,02358	0,94999	0,01512	0,91854	0,01895	0,74972	0,06862	0,92000	0,02794
new-thyroid2	0,93175	0,01785	0,07349	0,02090	0,95296	0,01486	0,92664	0,01992	0,77949	0,04516	0,92556	0,02534
page-blocks0	0,90201	0,00769	0,09580	0,00495	0,94467	0,00308	0,87167	0,00592	0,58970	0,01203	0,91215	0,00686
pima	0,76336	0,01252	0,27811	0,01090	0,77096	0,01283	0,72100	0,00686	0,42111	0,01616	0,72142	0,02471
segment0	0,98744	0,00309	0,01456	0,00432	0,99145	0,00256	0,98308	0,00223	0,94236	0,01617	0,98636	0,00519
vehicle0	0,94718	0,00464	0,11229	0,00465	0,92159	0,00361	0,91147	0,00466	0,72655	0,00956	0,86520	0,00641
vehicle1	0,50798	0,01785	0,72797	0,03472	#¡DIV/0!	#¡DIV/0!	0,02810	0,06283	0,01284	0,02872	0,02286	0,05111
vehicle2	0,88321	0,01371	0,18909	0,00749	0,85939	0,00659	0,83092	0,00608	0,57759	0,01323	0,78632	0,00999
vehicle3	0,50669	0,01495	0,73355	0,03546	#¡DIV/0!	#¡DIV/0!	0,02577	0,05762	0,00931	0,02083	0,02583	0,05775
vowel0	0,93801	0,01202	0,05062	0,00620	0,97158	0,00354	0,93046	0,00899	0,73758	0,02864	0,95279	0,00647
wisconsin	0,96362	0,01058	0,06735	0,00662	0,90836	0,00843	0,93654	0,00647	0,85536	0,01373	0,95142	0,01204
yeast1	0,74892	0,00800	0,29273	0,00414	0,77573	0,00557	0,70000	0,01087	0,36188	0,01317	0,71469	0,01517
yeast-2_vs_8	0,80226	0,03333	0,12375	0,03285	0,92975	0,02014	0,72874	0,03239	0,31009	0,06161	0,88734	0,03544
yeast3	0,94394	0,00536	0,08449	0,01068	0,95036	0,00666	0,92307	0,00997	0,66798	0,02968	0,91325	0,01279
yeast4	0,87467	0,01781	0,18490	0,02873	0,89407	0,01795	0,82436	0,01592	0,19730	0,03569	0,81425	0,02978
yeast5	0,95898	0,01020	0,06617	0,01082	0,96464	0,00600	0,95204	0,00809	0,44906	0,03956	0,93264	0,01177
yeast6	0,87153	0,01830	0,11252	0,01652	0,93847	0,00980	0,87528	0,01545	0,25293	0,02940	0,88794	0,01664

APENDICE 2

Funciones programadas para la ampliación al formato KEEL de GureKS

En las siguientes páginas se encuentran el código de programación necesario para adaptar la plataforma GureKS a la lectura del formato de bases de datos KEEL y la posterior la generación de submuestras con la distribución y técnica de remuestreo deseada.

Función encargada de leer el formato de datos y alamcenarlo en las estructuras correspondiente

```
int CArbolDoc::Leer_Fic_Formato()
/* Lee el fichero de formato de los datos y configuracion de vables. (*.FOR)
 */
/* &&& Vables. Globales: Vec_Vables, n_Fic_Fmt, Ident_VD, N_Var */
{
    char Fic_Fmt[L_NOM];   // Formato de los Campos a leer
    int  cont_var,Tot_Var; // Contador de Variables No Ignoradas y Totales
    char lin[L_LIN]/*,lin_old[L_LIN]*/; // Buffer de Lectura (Linea fichero)
    char lin1[L_LIN];
    BOOL FIN_FICH; // Se ha tratado hasta la ultima linea del fichero
    CString msg;
    char *st;
    char *aux;
    char *aux2;
    char *aux3;
    CVable *p_Vable,*p_Vable_VD;
    CVableF *p_VableF;
    FILE *ffor;
    BOOL FRAUDE_ESP; // Valor de Fraude Especificado en el fichero de formato
    BOOL VD_ESP;     // Variable Dependiente Especificado en el fichero de formato
    int i_vableNoCont,i_vableCont;

    if ((ffor=fopen(Nom_Fich_For,"r"))==NULL)
    {
        msg.Format("El Fichero de Formato de Datos \"%s\" NO existe",Nom_Fich_For);
        AfxMessageBox((LPCSTR)msg,MB_ICONEXCLAMATION);
        return(NULL);
    }
    Texto_Barra_Estado((LPCTSTR)" Leyendo Fichero Formato Variables...");
    // Inicializacion
    // -- Vables. Globales
    // Formato KEEL
    if(Nom_Fich_For.Right(4)==".dat")    m_FormatoKEEL=TRUE;
    n_Fic_Fmt=F_INDEF;
    m_FormatoUCI=FALSE;
    Ident_VD=-1;
    // -- Vables. Locales
    cont_var=0;
    Tot_Var=0;
    FIN_FICH=FALSE;
    FRAUDE_ESP=TRUE;
    VD_ESP=FALSE;
    lin[0]='\0';
    fgets(lin,L_LIN,ffor);
    int con;
//################
    memcpy(lin1,lin,strlen(lin)+1);

    if(m_FormatoKEEL)
    { //KEEL format
        int contador_atributos=0;
        BOOL att_list;
        // @relation
        //strcspn  http://souptonuts.sourceforge.net/code/strcspn_strspn_example.c.html
        st=lin1;
        if(con=strcspn(strupr(lin),ST_KL_RELATION)==strlen(strupr(lin)))
        {
            msg.Format("\nError: '%s' not found!",ST_KL_RELATION);
            AfxMessageBox((LPCSTR)msg,MB_ICONEXCLAMATION);
            fclose(ffor);
            return(NULL);
        }
        st=st+con;
        // Read the attribute list
        att_list=FALSE;
        fgets(lin,L_LIN,ffor);
```

```
        memcpy(lin1,lin,strlen(lin)+1);
         st=lin1;
         if(con=strcspn(_strupr(lin),ST_KL_ATTRIBUTE)!=strlen(_strupr(lin))){
             att_list=TRUE;// true mientras quedan atributos por leer
             st=st+con;
         }
         else
         {
             msg.Format("\nError: '%s' not found!",ST_KL_ATTRIBUTE);
             AfxMessageBox((LPCSTR)msg,MB_ICONEXCLAMATION);
             fclose(ffor);
             return(NULL);
         }
         i_vableNoCont=i_vableCont=0;
         while(att_list)// mientras sea true sigo leyendo atributos
         {
                 st=lin1;
          // @attribute
             char nom_var[L_LIN];
             p_Vable = new CVable;
             p_VableF = new CVableF;
             sscanf((st+strlen(ST_KL_ATTRIBUTE)),"%s",nom_var);

             st=strstr(st,nom_var);
             st+=strlen(nom_var);
             p_Vable->Nom=nom_var;
             p_VableF->Nom=nom_var;
            /*  Skip to first non-space character  */
             st+=SkipFirstNonSpace(st,0);
            // Tratamiento de Tipos de atributos

             if(st[0]=='{')
             {
              //tipo discreto
                 m_vLocalizadorTabDatos.Add(ID_T_Tab_Datos);
                 m_vLocalizadorVableTabDatos.Add(i_vableNoCont);
                 i_vableNoCont++;
                 char cad[80];
                 int cont;
                 st++;
             //  if(!(p_Vable->VD)){// la primera etiqueta para los missing
                     p_Vable->Vec_CatEtiq.Add("?");
             //  }

                 do{// leo lo que viene entre dos llaves
                     st+=SkipFirstNonSpace(st,0);
                     sscanf(st,"%[^,}]", cad);
                     p_Vable->Vec_CatEtiq.Add(cad);
                     st+=strlen(cad);
                     st+=SkipFirstNonSpace(st,0);
                     if(st[0]==','){
                         st++;
                     }
                 }while(st[0]!='}');
                //Actualizo el vector de categorias

                 for (cont = 0; cont<p_Vable->Vec_CatEtiq.GetSize() ; cont++){
                     p_Vable->Vec_Cat.Add(cont);
                 }
             }
             else// tengo un entero o un real
             {
                //guardo en aux por si st=strstr((st),ST_KL_INTEGER))==NULL
                 aux=st;
                 if((st=strstr(_strupr(st),ST_KL_INTEGER))!=NULL)
                 {
                     //tipo discreto(entero)
                     m_vLocalizadorTabDatos.Add(ID_T_Tab_Datos);
                     m_vLocalizadorVableTabDatos.Add(i_vableNoCont);
                     i_vableNoCont++;
```

```
                    st+=strlen(ST_KL_INTEGER);
                    st+=SkipFirstNonSpace(st,0);
                    int numMinInt, numMaxInt=0;
                    char o[10], f[10];
                     // limites
                    if (st[0]=='['){
                        sscanf(st+1,"%[^',], %[^]]",o,f);
                        numMinInt=atoi(o);
                        numMaxInt=atoi(f);
                    }
                    p_Vable->Min=numMinInt;
                    p_Vable->Max=numMaxInt;
                }

                else
                {
                    st=aux;
                    if((st=strstr(_strupr(st),ST_KL_REAL))!=NULL)
                    {
                        //tipo continuo(real)
                        m_vLocalizadorTabDatos.Add(ID_T_Tab_DatosCont);
                        m_vLocalizadorVableTabDatos.Add(i_vableCont);
                        i_vableCont++;
                        st+=strlen(ST_KL_REAL);
                        st+=SkipFirstNonSpace(st,0);
                        float numMinDouble, numMaxDouble=0.0;
                        char o[10], f[10];
                        if (st[0]=='['){
                            sscanf(st+1,"%[^',], %[^]]",o,f);
                            numMinDouble=(float)atof(o);
                            numMaxDouble=(float)atof(f);
                        }
                    }
                    else
                    {
                        msg.Format("\nError: Tipo de atributo NO soportado!");
                        AfxMessageBox((LPCSTR)msg,MB_ICONEXCLAMATION);
                        fclose(ffor);
                        return(NULL);
                    }
                }
            }

           // Leer siguiente la
            do{
                fgets(lin,L_LIN,ffor);
                memcpy(lin1,lin,strlen(lin)+1);


            }while((lin[0]=='\n') && (!feof(ffor)));

            if((st=strstr(_strupr(lin),ST_KL_ATTRIBUTE))==NULL)// he leido @inputs
@outputs o @data
                att_list=FALSE;
           //A la variable a los vectores Vec Vables y Vec VablesF
            Vec_Vables.Add((CObject*)p_Vable);
            Vec_VablesF.Add((CObject*)p_VableF);
            Tot_Var++;

        }// fin del while, la lectura de atributos

       // Actualizaci el Total de Vables No Continuas y Continuas y T Var
        T_Var=Tot_Var;
        T_VarCont=i_vableCont;
        T_VarNoCont=i_vableNoCont;

       // @inputs @outputs o @data


        if((st=strstr(_strupr(lin),ST_KL_INPUTS))!=NULL){// @input     ###OPCIONAL
```

```
            char cad[80];
            st+=strlen(ST_KL_INPUTS);
            st+=SkipFirstNonSpace(st,0);
            CStringArray strVecInputs;
            int numAtributos=0;
            do{// leo todos los atributos
                sscanf(st,"%[^,]", cad);
                strVecInputs.Add(cad);
                strVecInputs[numAtributos].TrimRight();
                st+=strlen(cad);
                numAtributos++;
                st+=SkipFirstNonSpace(st,0);
                if(st[0]==','){
                    st++;
                    st+=SkipFirstNonSpace(st,0);
                }
            }while(st[0]!='\0');

    //Miro si hay variables ignoradas
            int i;
            int j;
            int contadorIgn=0;
            for(i=0; i < (Vec_VablesF.GetSize()-1) ; i++)
            {// -1 por el atributo Class que es de salida
                BOOL esIgnorada=TRUE;
                CString inputActual= ((CVableF*) Vec_VablesF.GetAt(i))->Nom;
                for(j=0; j < strVecInputs.GetSize() ; j++){
                    CString prueba= strVecInputs.GetAt(j);
                    if(inputActual.CompareNoCase(prueba)) {
                        esIgnorada=FALSE;
                     break;
                    }

                }
                if(esIgnorada){
                    ((CVable*) Vec_VablesF.GetAt(i))->Ign=TRUE;
                    Vec_Vables.RemoveAt(i-contadorIgn);
                    contadorIgn++;
                }
            }

        N_Var=T_Var-contadorIgn;
        if(numAtributos!=(Tot_Var-1))
        {
            msg.Format("\n Hay variables ignoradas '%s'  !",ST_KL_INPUTS);
            AfxMessageBox((LPCSTR)msg,MB_ICONEXCLAMATION);
        }

        fgets(lin,L_LIN,ffor);

        }
        else
        {
            st=aux2;
            msg.Format("\n no se ha encontrado el token '%s'  !",ST_KL_INPUTS);
            AfxMessageBox((LPCSTR)msg,MB_ICONEXCLAMATION);
        }

        if((st=strstr(_strupr(lin),ST_KL_OUTPUTS))!=NULL){// @output  ###OPCIONAL
            BOOL estaVD = FALSE;
            CString cad;
            st+=strlen(ST_KL_OUTPUTS);
            st+=SkipFirstNonSpace(st,0);
            CStringArray strVecOutputs;
            do{// leo todos los atributos
                sscanf(st,"%[^,\n]", cad);
                cad.TrimRight();
                strVecOutputs.Add(cad);
                st+=strlen(cad);
                st+=SkipFirstNonSpace(st,0);
```

```
                if(st[0]==','){
                    st++;
                    st+=SkipFirstNonSpace(st,0);
                }
            }while(st[0]!='\0');

            if(strVecOutputs.GetSize()>1){
                msg.Format("\nError:  El problema tiene mas de una clase!");
                AfxMessageBox((LPCSTR)msg,MB_ICONEXCLAMATION);
                fclose(ffor);
                return(NULL);
            }
            else{

                CString vableClase=strVecOutputs.GetAt(0);
                int cont1;
                for (cont1 = 0; cont1<Vec_Vables.GetSize(); cont1++){
                    CString vableActual=((CVable*)Vec_Vables.GetAt(cont1))->Nom;
                    if(vableClase.CompareNoCase(vableActual)==0){
                        ((CVable*)(Vec_Vables.GetAt(cont1)))->Vec_Cat.RemoveAt(0);
                        ((CVable*)(Vec_Vables.GetAt(cont1)))->Vec_CatEtiq.RemoveAt(0);
                        VD_ESP=TRUE;
                        ((CVable*)(Vec_Vables.GetAt(cont1)))->VD=TRUE;
                        NumCatVD=((CVable*)(Vec_Vables.GetAt(cont1)))-
>Vec_Cat.GetSize();
                        Ident_VD=cont1;
                        p_Vable_VD=((CVable*)(Vec_Vables.GetAt(cont1)));
                        break;
                    }
                }
            }
     //leo nueva linea
        fgets(lin,L_LIN,ffor);
        char s=lin[0];
        }
        else
        {
            st=aux3;
            msg.Format("\n No se ha encontrado el la variable dependiente");
            AfxMessageBox((LPCSTR)msg,MB_ICONEXCLAMATION);
        }

     // @data
        if((st=strstr(_strupr(lin),ST_KL_DATA))!=NULL){// @output
            FIN_FICH=TRUE;
        }
        else
        {
            msg.Format("\nError: '%s' not found!",ST_KL_DATA);
            AfxMessageBox((LPCSTR)msg,MB_ICONEXCLAMATION);
            fclose(ffor);
        //  return(NULL);
       }

    // he terminado y cierro el fichero
    fflush(ffor);
    fclose(ffor);
    return(ONGI);

// fin del formato KEEL
```

Función para la lectura y almacenamiento de los datos en la estructura correspondiente y generación del *.names* y *.data* a partir de las mismas.

```
int CMuestra::Leer_Fic_Datos_KEEL(CString nom_fich, BOOL Modificar_Vec_Cat)
{

    FILE *fdat;
    CString msg;
    int caso,var,cont_var,cont_var_NO_Ign;
    CVableF *p_VableF;
    CVable *p_Vable;

    if(pDoc->Vables_Ign!=SIN_IGNORAR){
        msg.Format("Opcion no contemplada para formato KEEL ");
        AfxMessageBox((LPCSTR)msg,MB_ICONEXCLAMATION);
        return(NULL);
    }

   // Traspasar los datos a la Tabla de Datos
    CMiArrayGenericoBase<T_Tab_Datos> *p_fila;
    CMiArrayGenericoBase<T_Tab_DatosCont> *p_filaCont;
    CMiArrayGenericoBase<T_Tab_Datos> *p_fila1;//Para el .names
    CMiArrayGenericoBase<T_Tab_DatosCont> *p_filaCont1;

    if ((fdat=fopen(nom_fich,"r"))==NULL)
    {
        msg.Format("El Fichero de Datos \"%s\" NO existe",nom_fich);
        AfxMessageBox((LPCSTR)msg,MB_ICONEXCLAMATION);
        return(NULL);
    }

    Fic_Datos=nom_fich;
    CString cadena;
    char linea[L_LIN];
    linea[0]='\0';
    char *st;
    fgets(linea,L_LIN,fdat);
   // me situo en la linea siguiente al @data

    while((strstr(_strupr(linea),ST_KL_DATA))==NULL){
        fgets(linea,L_LIN,fdat);
    }

    for(caso=0;(fgets(linea, L_LIN, fdat)!=NULL)&&(caso<N_Casos);caso++)
    {
        if(((caso)%1000)==0)
        {
            msg.Format("Submuestra \"%s\": Leidos %ld
casos...",Extraer_solo_nombre(nom_fich),caso);
            Texto_Barra_Estado((LPCTSTR)msg);
        }
        cont_var=0;
        cont_var_NO_Ign=0;
        if(Tab_Datos_fil_x_col==C_CASOSxVABLES){
            p_fila=&Tab_Datos[caso];
            p_filaCont=&m_Tab_DatosCont[caso];
        }
        st=linea;
        for(var=0;var<pDoc->T_Var;var++)
        {
            sscanf(st,"%[^',']",cadena);
            st+=strlen(cadena);
            cadena.TrimRight();
            if(st[0]==',')st++;
            st+=SkipFirstNonSpace(st,0);
            //Miro el tipo de la varibale Discreta o Continua
                if(pDoc->m_vLocalizadorTabDatos[var]==ID_T_Tab_Datos)//ID_T_Tab_Datos=0
discretos y enteros
                {
```

```
                    p_Vable=(CVable*)(pDoc->Vec_Vables[var]);
                    int u=0;
                    CString st4;
                    BOOL encontrado=FALSE;
                    if(p_Vable->Vec_Cat.GetSize()>0){
                        for(u=0;u<p_Vable->Vec_CatEtiq.GetSize();u++){

                            st4=p_Vable->Vec_CatEtiq[u];
                            if(cadena.CompareNoCase(st4)==0){
                                encontrado=TRUE;
                                break;
                            }
                        }
                    }
                    if(encontrado){// tngo VD o discreta
                        if(var==pDoc->Ident_VD){
                            m_vMatIdentCasoVD[u].Add((T_NCasos)caso);
                            p_fila->SetAt((int)(pDoc-
>m_vLocalizadorVableTabDatos[var]),(T_Tab_Datos)(u));
                        }
                        else{
                            p_fila->SetAt((int)(pDoc-
>m_vLocalizadorVableTabDatos[var]),(T_Tab_Datos)(u));
                        }
                    }
                    else{// entero
                        if(cadena=="?"){
                            p_fila->SetAt((int)(pDoc-
>m_vLocalizadorVableTabDatos[var]),(T_Tab_Datos)UnknownInt);
                        }
                        else

                            p_fila->SetAt((int)(pDoc-
>m_vLocalizadorVableTabDatos[var]),(T_Tab_Datos)(atoi(cadena)));
                    }
                }
                else{//ID_T_Tab_DatosCont=1 reales
                    if(cadena=="?"){
                        p_filaCont->SetAt((int)(pDoc-
>m_vLocalizadorVableTabDatos[var]),Unknown);
                    }
                    else
                        p_filaCont->SetAt((int)(pDoc-
>m_vLocalizadorVableTabDatos[var]),atof(cadena));
                }
            }

    msg.Format("Submuestra \"%s\": Leidos %ld
casos",Extraer_solo_nombre(nom_fich),caso);
    Texto_Barra_Estado((LPCTSTR)msg);
    }
    fclose(fdat);

    int u=pDoc->DameNumCatVD();
    int z=pDoc->NumCatVD;

    if(pDoc->DameNumCatVD()!=pDoc->NumCatVD)
    {
        msg.Format("El nmero de categorde la VD esperadas son %d y\n\rse han encontrado
%d",
            pDoc->NumCatVD,pDoc->DameNumCatVD());
        AfxMessageBox(msg);
    }
    // Generar . data
    FILE *ficheroData;
    CString nombreData=Fic_Datos+".data";
    ficheroData=fopen(nombreData,"w");

    msg.Format("Generando el .data para %s" ,Extraer_solo_nombre(nom_fich));
    Texto_Barra_Estado((LPCTSTR)msg);
```

```
    int datoEntero=0;
    float datoReal;
    CString st2,st_ent,lin;
    for(caso=0;caso<N_Casos;caso++)
    {
        p_fila1=&Tab_Datos[caso];
        p_filaCont1=&m_Tab_DatosCont[caso];
        lin.Empty();

        for(var=0;var<pDoc->T_Var;var++)
        {
            if(pDoc->m_vLocalizadorTabDatos[var]==ID_T_Tab_Datos)//ID_T_Tab_Datos=0
discretos y enteros
            {
                CVable *p_VableActual=(CVable*)(pDoc->Vec_Vables[pDoc-
>m_vLocalizadorVableTabDatos[var]]);
                if(p_VableActual->Vec_CatEtiq.GetSize()>0){// tengo una discreta bien
VD, bien discreta
                // etiquetas
                    int cont=p_fila1->GetAt((int)(pDoc-
>m_vLocalizadorVableTabDatos[var]));
                    if(!p_VableActual->VD){
                        st_ent.Format("%s",p_VableActual->Vec_CatEtiq[cont]);//missing
                        if(var==(pDoc->T_Var)-1)
                            st2+=st_ent;
                        else
                            st2+=st_ent+",";
                    }
                    else{//VD
                        CVable *p_VableClase=(CVable*)(pDoc->Vec_Vables[pDoc-
>Ident_VD]);
                        st_ent.Format("%s",p_VableClase->Vec_CatEtiq[p_fila1-
>GetAt((int)(pDoc->m_vLocalizadorVableTabDatos[var]))]);
                        if(var==(pDoc->T_Var)-1)
                            st2+=st_ent;
                        else
                            st2+=st_ent+",";
                    }
                }
                else{//tengo un entero
                    if(var==pDoc->T_Var-1){//Caso especial
                        CVable *p_VableClase=(CVable*)(pDoc->Vec_Vables[pDoc-
>Ident_VD]);
                        st_ent.Format("%s",p_VableClase->Vec_CatEtiq[p_fila1-
>GetAt((int)(pDoc->m_vLocalizadorVableTabDatos[var]))]);
                        st2+=st_ent;
                    }
                    else{// entero
                        datoEntero=(int)(p_fila1->GetAt((int)(pDoc-
>m_vLocalizadorVableTabDatos[var])));
                        if(datoEntero==UnknownInt)
                            st_ent.Format("?");
                        else
                            st_ent.Format("%d",datoEntero);
                        if(var==(pDoc->T_Var)-1)
                            st2+=st_ent;
                        else
                            st2+=st_ent+",";
                    }
                }
            }
            else{
                 //real
                datoReal=(p_filaCont1->GetAt((int)(pDoc-
>m_vLocalizadorVableTabDatos[var])));
                if(datoReal==Unknown)
                    st_ent.Format("?");
                else
                    st_ent.Format("%f",datoReal);
```

```
                if(var==(pDoc->T_Var)-1)
                    st2+=st_ent;
                else
                    st2+=st_ent+",";
            }
            lin+=st2;
            st2.Empty();
        }
        fprintf(ficheroData,"%s\n",lin);
    }
    fclose(ficheroData);

// Generar .names
    FILE *ficheroNames;
    CString nombreNames=Fic_Datos+".names";
    ficheroNames=fopen(nombreNames,"w");
    msg.Format("Generando el .names para %s" ,Extraer_solo_nombre(nom_fich));
    Texto_Barra_Estado((LPCTSTR)msg);
    CString st3,st8,st_names,line;

    //primera linea de names -> las etiquetas separadas por coma
    CVable *p_VableClase;
    p_VableClase=(CVable*)(pDoc->Vec_Vables[pDoc->Ident_VD]);
    for(int h=0;h<pDoc->NumCatVD;h++){
        if(h==(pDoc->NumCatVD)-1){// ultima etiqueta acabada en punto
            st3+=p_VableClase->Vec_CatEtiq[h]+".";
        }
        else
            st3+=p_VableClase->Vec_CatEtiq[h]+", ";
        }
        fprintf(ficheroNames,"%s\n",st3);
        // paso a las variables

        for(int p=0;p<pDoc->T_Var-1;p++){// todos menos la VD que ya esta
            st3.Empty();
            CVable *p_VableAtributo;
            p_VableAtributo=(CVable*)(pDoc->Vec_Vables[p]);
            st3=p_VableAtributo->Nom+": ";
            if (p_VableAtributo->Ign)
                st3+="ignore.";
            else{// tomo los valores
                if(pDoc->m_vLocalizadorTabDatos[p]==ID_T_Tab_Datos){//ID_T_Tab_Datos=0
discretos y enteros
                    if(p_VableAtributo->Vec_Cat.GetSize()>0){// tengo un discreto
                        for(int l=1;l<p_VableAtributo->Vec_CatEtiq.GetSize();l++){
                            if(l==(p_VableAtributo->Vec_CatEtiq.GetSize()-1))// 1 para
no salir el missing
                                    st3+=p_VableAtributo->Vec_CatEtiq[l]+".";
                            else
                                    st3+=p_VableAtributo->Vec_CatEtiq[l]+", ";
                        }
                    }
                    else{// tengo un entero
                        st3+="discrete 0.";
                        for(int j=p_VableAtributo->Min;j<p_VableAtributo->Max+1;j++){
                            if(p_VableAtributo->Max==j){
                            }
                            else{

                            }
                        }
                        st3+=st8;
                    }
                }
                else{// ID_T_Tab_ContDatos real

                    st3+="continuous.";
                }
            }
            fprintf(ficheroNames,"%s\n",st3);
```

```
        }
        fclose(ficheroNames);
   return(ONGI);
}
```

Función encargada de crear la muestra y volcar los datos en ella a partir de las estructuras manejadas por la función anterior.

```
CMuestra * CMuestra::Construir_Muestra(CString nom_fic_dat,CArbolDoc *pDoc,int
fil_x_col)
{
    FILE *fdat;
    int Tam_lin;
    long Tam_Fich;
    CFileStatus status;
    CString msg;
    long num_casos;
    int c,ant_c;
    CMuestra *p_mu;

    if ((fdat=fopen(nom_fic_dat,"r"))==NULL)
    {
      msg.Format("El Fichero de Datos \"%s\" NO existe",nom_fic_dat);
      AfxMessageBox((LPCSTR)msg,MB_ICONEXCLAMATION);
      return(NULL);
    }

    if(pDoc->m_FormatoKEEL)
    { //KEEL format
        int z=fil_x_col;
        Texto_Barra_Estado((LPCTSTR)" Leyendo Datos..");
        char lin[L_LIN];
        lin[0]='\0';
        num_casos=0;
        fgets(lin,L_LIN,fdat);

        while((strstr(_strupr(lin),ST_KL_DATA))==NULL){
            fgets(lin,L_LIN,fdat);
        }

        while (fgets(lin, L_LIN, fdat) != NULL){
            num_casos++;
        }
    }

    else
    { // Formato de Datos definido para GureKS
        if(nom_fic_dat.Right(4)==".lid")

                .
                .
                .
    }

    fclose(fdat);  // Volver al principio del fichero

    if(pDoc->m_FormatoKEEL){
        if(pDoc->Vables_Ign==SIN_IGNORAR)
            p_mu = new CMuestra(num_casos,pDoc->T_VarNoCont,pDoc-
>T_VarCont,C_CASOSxVABLES,pDoc);
        else
            p_mu = new CMuestra(num_casos,pDoc->T_VarNoCont,pDoc-
>T_VarCont,fil_x_col,pDoc);
        return p_mu;
    }
```

```
    else{

        if(pDoc->Vables_Ign==SIN_IGNORAR)
            p_mu = new CMuestra(num_casos,pDoc->T_Var,fil_x_col,pDoc);
        else
            p_mu = new CMuestra(num_casos,pDoc->N_Var,fil_x_col,pDoc);
        return p_mu;
    }
}
```

APENDICE 3

Introducción a las herramientas de desarrollo

Se muestran de una manera sencilla y sin pretender ser exhaustiva los principales fundamentos y funcionamiento de las herramientas empleadas para el desarrollo del proyecto, Visual Basic Express Edition 6.0 y VisualBasicScripting.

Entorno de desarrollo Visual C++

Tanto para desarrollar una aplicación desde cero, como para realizar ampliaciones o actualizaciones en otras, -como es nuestro caso-, es recomendable emplear un entorno de desarrollo, ya que facilita el trabajo, si bien no es la única manera, ya que se puede programar en ficheros de texto y compilarlos desde la línea de comandos.

El entorno de desarrollo elegido para la tarea de adaptar la plataforma GureKS al formato de bases de datos KEEL, es el Visual C++ 6.0 en su versión Express Edition.

Microsoft Visual C++ es un entorno de desarrollo de aplicaciones en el lenguaje de programación C++ para la plataforma Windows. De dicho entorno destacan las siguientes secciones:

Ilustración 65 Apariencia principal de la aplicación Visual C++ y sus principales secciones

Menu: Posee un menú dinámico que puede adaptarse a los diferentes estados del entorno de desarrollo.

- Menú File, dispone de algunas opciones para todos o los últimos ficheros abiertos.
- Menú Tools, se puede adaptar a las necesidades de cada usuario.
- Menú Help, encargado de ofrecer un sistema de ayuda visual.

Barra de herramientas: Atajo de las opciones del menú principal para proporcionar un acceso rápido. Mediante *customize* permite modificar la estructura de las mismas para adaptarla a las necesidades del usuario. Se nos permite:

- Agregar un botón
- Eliminar un botón

Workspace: Herramienta básica para el desarrollo de una aplicación. Muestra una lista de las clases y recursos de los que está compuesto el proyecto. Permite acceder de forma sencilla a los elementos importantes del entorno de desarrollo: subrutinas y funciones. A la izquierda se muestra la apariencia del *workspace*, en la que se agrupan los ficheros de cabecera por un lado, los ficheros fuente por otro y finalmente una tercera zona destinada a los demás recursos del proyecto.

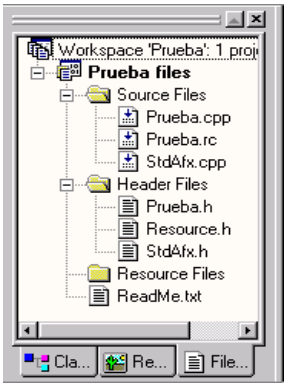


Ilustración 66 Apariencia gráfica del workspace

Una propiedad interesante es la de permitir modificaciones en el código fuente, con tan sólo pulsar dos veces en la función deseada.

Desde el *workspace* podemos acceder mediante la pestaña FileView a los diferentes archivos que pertenecen al proyecto abierto.

Editor del código fuente: Simplifica considerablemente el manejo del código fuente y, por tanto, la programación. Destacan:

• La aplicación automática de sangría en los bloques entre llaves.

• Facilitar la navegación entre el código fuente mediante el cuadro de diálogo Go To.

•Posibilidad de utilizar marcadores de texto para permitir el acceso directo.

• Se resalta la sintaxis de C++ marcando en color las palabras clave.

Ventana de presentación: La pestaña Build proporciona mensajes de estado del compilador de Visual C++, además de mensajes de error, indicando el nombre del archivo, el número de página y el número del error y una descripción del mismo.

• La ficha Debug muestra los mensajes emitidos por el depurador.

Proceso de compilación

Elemento de gran importancia, ya que es el encargado de compilar el código fuente convirtiéndolo en un ejecutable. El resultado del proceso de compilación es la generación de archivos *.obj*.

Para compilar tenemos dos opciones: una de la manera rápida mediante el atajo CTRL + F7 y la otra a través del menú Build y la opción compile.

Los resultados de la compilación se muestran en la ventana de presentación, indicando los errores y posibles errores denominados *warnings*.

Una vez compilados los archivos del código fuente, es necesario vincularnos con el objetivo de agruparlos todos en un único archivo. El encargado de esto es el linker y se encuentra en el menú Build con el Build nombreproyecto.exe.

Ejecución del programa

Una vez compilado el programa sólo falta ejecutarlo, en el menú Build, con la opción Execute NombreProyecto.exe se activa la ejecución del proceso de vinculación.

Otra opción interesante que ofrece esta herramienta de desarrollo es la opción de debugear, que es de una grandísima utilidad cuando la envergadura del proyecto es grande.

Depuración de errores

La depuración de errores es el nombre que recibe el proceso de eliminación de errores de un programa informático. El *debugger* es una herramienta diseñada para poder moverse entre las instrucciones, ayudando al programador a encontrar errores algorítmicos que no sintácticos. Permite mediante una ejecución, pasó a paso, localizar comportamientos no deseados en la aplicación e identificar las fuentes de error.

Para poder utilizar el depurador correctamente durante el proceso de ejecución de un programa, hay que ejecutar dicho programa en modo 'Debug'.

El modo depuración de errores tiene su propia apariencia física como se aprecia en la ilustración 67.

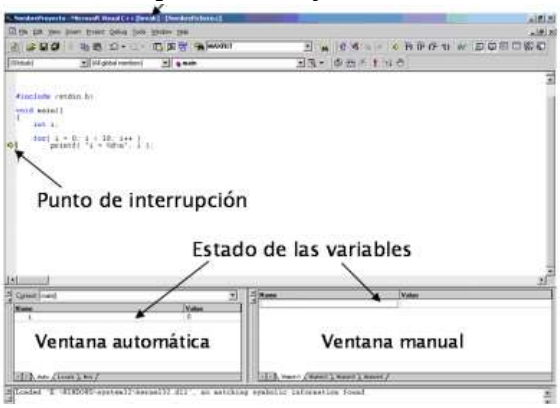


Ilustración 67 Apariencia del Debugger del Visual C++

Entorno de trabajo

El entorno de trabajo para programar en Visual Basic Scripting está formado por distintas secciones.

Como se desprende de la ilustración 68, el entorno de trabajo lo componen 4 zonas diferenciadas:

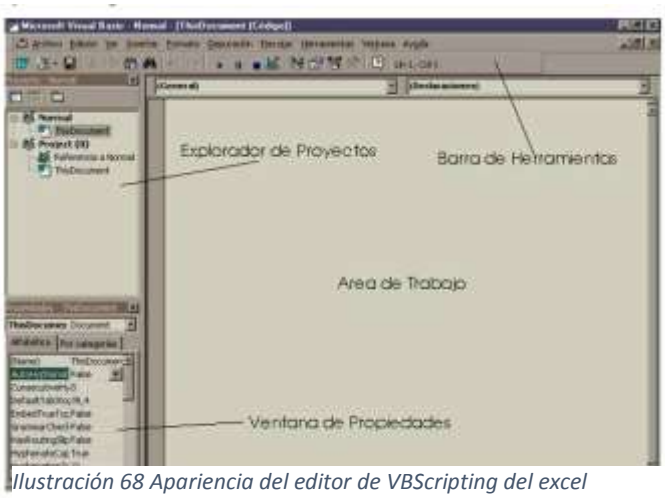


Ilustración 68 Apariencia del editor de VBScripting del excel

Explorador de proyectos: Herramienta de gestión que muestra una lista jerárquica de cada proyecto abierto, a su vez también nos muestra los componentes asociados con los proyectos. Útil para movernos de uno a otro.

Ventana de propiedades: Muestra las propiedades para el objeto activo, mediante la cual podemos configurar y modificar propiedades del mismo.

Barra de herramientas: En ella encontraremos los botones para las funciones más usadas.

Área de trabajo: La zona principal, en la que desarrollaremos nuestros scripts. Se escribe el código en Visual Basic que posteriormente se ejecutará.

Detalladas las zonas y funciones de cada uno de los componentes de este entorno de trabajo integrado en Excel, veremos la estructura que siguen estos programas.

Estructura de un programa en VBscripting

La programación en Visual Basic Scripting se rige por el paradigma de la programación orientada a objetos. Por tanto se disponen de:

- Objetos: Engloba a cualquier entidad de Office funcional y programable, por ejemplo: botones, tablas, consultas, formularios…
- Colecciones: Objetos distintos que son del mismo tipo, concepto similar a la clase, workseets o conjunto de hojas, workbook libros Excel abiertos.
- Miembros: Conjunto formado por el estado y el comportamiento que se puede aplicar a un objeto concreto.

Casi todos los objetos pueden derivarse en otros o subobjetos menores. Esto indica que existe una jerarquía entre los mismos, de modo, que un objeto que forma parte del primer nivel puede derivar en un subobjeto que forma parte del siguiente nivel. Véase la ilustración 69.

En el nivel superior encontramos el objeto *application*, objeto de mayor nivel al que podemos hacer referencia en la aplicación.

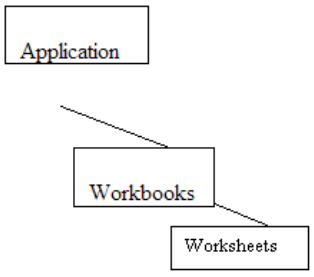


Ilustración 69 Jerarquía de clases en Excel

Los módulos de código constan principalmente de tres partes:

- Instrucciones.
- Declaración de constantes
- Declaración de variables

Las instrucciones se pueden organizar de manera que formen subrutinas y funciones:

Sub: Procedimiento o rutina que no devuelve ningún valor, las líneas de código que incluye realizan alguna acción, pero no devuelven o retornan ningún valor.

Sub

...

End Sub

```
Sub Botón1_AlHacerClic()

End Sub
```

Ilustración 70 Cuerpo de la definición de una subrutina

· Function: Realizan un cierto cálculo y retornan o devuelven un valor que puede ser utilizado en otros módulos, el código de una función está incluido entre estas dos líneas:

Function

…

EndFuction

```
Function PasaRealTipoScript(st As String)
Dim aux As String
    aux = Left(st, InStr(st, ",") - 1)
    aux = aux + " ."
    aux = aux + Right(st, Len(st) - InStr(st, ","))
    PasaRealTipoScript = aux
End Function
```

Ilustración 71 Ejemplo de defínción de una funcion

Como en todo lenguaje de programación, una variable puede ser declarada en distintos ámbitos:

- Nivel de procedimiento: Válido para la función en la que se declaran, y se realiza al comienzo.
- Nivel de módulo: Estas variables son declaradas al comienzo del programa y son utilizables en todo el programa.
- Nivel público: Hay que indicar antes de la variable que es de tipo Public, estarán disponibles para todos los procedimientos de todos los módulos.