*B*ENEMÉRITA
*U*NIVERSIDAD *A*UTÓNOMA DE *P*UEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

APLICACIÓN DE TÉCNICAS DE PREDICCIÓN CON DATOS FINANCIEROS

TESIS PROFESIONAL:
QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

DORIAN RUIZ ALONSO

ASESOR:

Dr. IVO HUMBERTO PINEDA TORRES

MARZO DE 2008

AGRADECIMIENTOS

Gracias a Dios

Por permitirme llegar hasta este momento tan importante de mi vida y lograr una meta más tanto en mi carrera como en la vida.

Gracias a mi Familia

Agradezco a mi madre, abuelos, hermanas, tíos, primos y sobrinas, por todo su apoyo, cariño, comprensión y sobre todo por estar en los momentos que han dejado huella en mi vida. Gracias por guiarme sobre el camino del respeto y educación. Ahora es cuando se deben de sentir orgullosos al ver reflejados todos sus esfuerzos.

Gracias a mi Asesor

Sin el no hubiera podido concluir este trabajo, por tomarse su tiempo y por cada una de las enseñanzas que me brindó.

Gracias a cada uno de los maestros

Que participaron en mi desarrollo profesional durante mi carrera, proporcionándome las herramientas necesarias para la vida profesional.

Gracias a todos mis amigos

Que estuvieron conmigo compartiendo tantas aventuras, experiencias, desveladas, peleas y alegrías, por su apoyo incondicional en las buenas y en las malas, he aprendido tanto de ustedes que siempre los voy a recordar.

Índice

Capítulo 1

Introducción

Los recientes cambios ocurridos en las políticas regulatorias de todo el mundo han favorecido la introducción de la competencia en muchos sectores industriales tradicionalmente regulados. Para conseguir una mejor eficiencia económica, los gobiernos están fomentado el desarrollo de distintos tipos de mercados liberalizados para el comercio de estos bienes. La principal característica de estos mercados es la libre concurrencia de ofertas de compra y venta.

En estos entornos los participantes tratan de construir sus ofertas para maximizar su beneficio esperado. La complejidad de estos nuevos mercados y la incertidumbre que genera su desconocimiento está favoreciendo el desarrollo de nuevas y complejas herramientas informáticas que combinan las técnicas más tradicionales con las últimas tecnologías en optimización, análisis de datos y sistemas distribuidos.

Dentro del mercado de valores el volumen de datos disponible es muy elevado y con información potencialmente muy rica. Dichos datos son reglas cuantitativas para generar señales de venta o compra basadas en la interpretación de dichas reglas. Una forma que se podrían usar estas reglas es responder preguntas del tipo ¿Hoy es un día bueno para vender y lograr una ganancia del 4% dentro de los próximos 63 días? ¿Será hoy el día correcto para vender si quiero evitar una pérdida del 5% en los próximos 5 días?

El problema de predicción que se pretende resolver es predecir si el objetivo siguiente es alcanzable en cualquier día dado:

Objetivo G: el índice se elevará en el 4 % o más dentro de los 63 días siguientes comerciales (3 meses).

En consecuencia cada día de comercio es clasificado en la categoría "de compra" si G es positivo o la categoría "de no-compra" si G es negativo.

Las herramientas de minería de datos predicen futuras tendencias y comportamientos, permitiendo en los negocios tomar decisiones proactivas y conducidas por un conocimiento acabado de la información; pueden responder a preguntas de negocios que tradicionalmente consumen demasiado tiempo para poder ser resueltas y a los cuales los usuarios de esta información casi no están dispuestos a aceptar.

Estas herramientas exploran las bases de datos en busca de patrones ocultos, encontrando información predecible que un experto no puede llegar a encontrar porque se encuentra fuera de sus expectativas.

1.1 Planteamiento

Dado un conjunto grande de valores financieros referente a un producto se trata de generar conocimiento a partir de este; para lo cual se utilizarán algoritmos de máquina de aprendizaje.

Se establecerán criterios de aprendizaje y entrenamiento con los datos disponibles con el objeto de precisar los parámetros de los algoritmos seleccionados.

1.2 Objetivos generales

Son los que tiene por objeto aplicar los algoritmos de predicción de minería de datos para la toma de decisiones, respecto de la compra que se haga de las acciones de Dow Jones, comparándolos entre ellos para encontrar el que más se ajuste al problema que se desea resolver.

1.3 Objetivos específicos

Se analizarán de manera detallada los conceptos de la minería de datos, para comprender los algoritmos de predicción, como objeto de identificar los que mejor se adapten a los datos utilizando la herramienta de minería de datos weka.

1.4 Metodología de la investigación

Una vez que he fijado el planteamiento y los objetivos del problema, pasaré a la recopilación de la información; donde extraeré las ideas principales que analizaré para visualizar la forma en la que se llegará a la solución de encontrar el algoritmo que se adapte a los datos financieros del precio de cierre de las acciones Dow Jones, es decir las diferentes tecnicas de minería de datos, siendo estas las siguientes:

a).- Arboles de descisión C4.5
b).- Naive Bayes
c).- Bagging
d).- Random Committee

Con el propósito de probar los algoritmos en la herramienta de minería de datos Weka, la cual es una extensa colección de algoritmos de máquinas de aprendizaje desarrollados por la Universidad de Waikato (Nueva Zelanda) implementados en Java; que son utilizadass para ser aplicadas sobre datos mediante las interfaces que ofrece. Además Weka contiene las herramientas necesarias para realizar transformaciones sobre los datos, tareas de clasificación, regresión, clustering, asociación y visualización.

Por último analizaré los resultados obtenidos de la herramienta antes mencionada para verificar el comportamiento de los datos en los algoritmos, así como las condiciones bajo las cuales trabajó y se adaptó de forma oficiante y así definir por que los demás no son buenos para la solución del problema.

Capítulo 2

Aprendizaje automático y minería de datos

El Aprendizaje Automático (Machine Learning) es el campo dedicado al desarrollo de métodos computacionales para los procesos de aprendizaje, y a la aplicación de los sistemas informáticos de aprendizaje a problemas prácticos. La minería de Datos (Data Mining) es la búsqueda de patrones e importantes regularidades en bases de datos de gran volumen. Estos dos campos han ido creciendo a lo largo de los años, y han cobrado una importancia considerable.

Hoy en día, como se almacenan grandes volúmenes de información en todas las actividades humanas, la minería de Datos está cobrando gran importancia, se busca obtener información valiosa a partir de los datos guardados. La minería de Datos utiliza métodos y estrategias de otras áreas o ciencias, entre las cuales podemos nombrar al Aprendizaje Automático. Cuando este tipo de técnicas se utilizan para realizar la minería, decimos que estamos ante una minería de Datos Inteligente.

2.1 Aprendizaje automático

Cualquier sistema que se considere inteligente debería poseer la habilidad de aprender, es decir, mejorar automáticamente con la experiencia. La idea principal detrás del aprendizaje, es que aquello que es percibido por el agente, debería ser usado no sólo para actuar, sino también para mejorar sus habilidades en el futuro.

El proceso de aprendizaje de un agente puede resultar de su interacción con el mundo como así también de la observación introspectiva de sus propios procesos internos.

Si consideramos al aprendizaje como la mejora del comportamiento a partir de la experiencia, existirían múltiples actividades que pueden ser consideradas como distintas formas de aprendizaje. Estas actividades podrían ser tan elementales como la memorización directa de una experiencia. En otros casos pueden implicar procesos más complejos como la generalización a partir de ejemplos e inclusive el descubrimiento de nuevos conceptos.

El aprendizaje de máquina o automático (en inglés Machine Learning (ML)), se ha tornado muy importante en muchas aplicaciones prácticas de Inteligencia Artificial, como por ejemplo los Sistemas Expertos, proveyendo una alternativa a las técnicas de adquisición de conocimiento tradicionales.

Los principales aspectos a ser considerados en el proceso de aprendizaje, se encuentran implícitos en la definición de aprendizaje de Herbert Simon:

"cualquier cambio en un sistema que le permite desempeñarse mejor la próxima vez, sobre la misma tarea u otra tomada de la misma población".

Esta definición cubre un amplio espectro de actividades que van desde mejorar el rendimiento de un sistema existente (ya sea en eficiencia o en la no reiteración de errores) hasta la adquisición de nuevos conceptos.

También esta definición habla de cambios en el agente que aprende, lo que implica que debería haber alguna manera de representarlos. En este sentido, serán válidos tanto los métodos que modelan el aprendizaje como la adquisición de conocimiento del dominio representado explícitamente (mediante sentencias en un lenguaje simbólico), como aquellos cuyo conocimiento está implícito (como por ejemplo redes neuronales) y que aprenden modificando su estructura completa (la organización e interacción entre las neuronas).

Otra característica que se observa en la definición de Simon, es la naturaleza empírica del aprendizaje, al involucrar la generalización a partir de la experiencia: el desempeño debería mejorar no sólo sobre la misma tarea, sino también sobre tareas similares en el dominio.

Debido a que los dominios reales suelen ser muy grandes, el sistema que aprende deberá, a partir de una experiencia limitada, adquirir el conocimiento que le permita generalizar correctamente cuando le sean presentados nuevas instancias del dominio (problema de inducción).

Aún cuando ML tiene sus orígenes y principales aplicaciones en Inteligencia Artificial, es un campo multidisciplinario que ha sido influenciado entre otros, por los resultados obtenidos en probabilidad y estadística, teoría de la complejidad computacional, teoría de control, teoría de la Información, etc.

ML ha sido aplicado a un gran número de aplicaciones científicas y comerciales, con una tendencia creciente en su uso.

2.1.1 Aprendizaje supervisado y no supervisado

Existen dos tipos de aprendizaje: el supervisado y el no supervisado. En el aprendizaje supervisado o aprendizaje a partir de ejemplos, el instructor o experto define clases y provee ejemplos de cada una. El sistema debe obtener una descripción clara para cada clase.

Cuando el instructor define una sola clase, provee ejemplos positivos (pertenecen a la clase) y negativos (no pertenecen a la clase). En este caso, los ejemplos importantes son los cercanos al límite, porque proveen información útil sobre los límites de la clase. Cuando el instructor define varias clases, el sistema puede optar por realizar descripciones discriminantes o no.

Un conjunto de descripciones es discriminante si el total de las descripciones cubren todas las clases, pero una descripción cubre una sola clase en particular.

En el aprendizaje no supervisado o aprendizaje a partir de observaciones y descubrimientos, el sistema debe agrupar los conceptos sin ayuda de un instructor.

El sistema recibe los ejemplos, pero no se predefine ninguna clase. Por lo tanto, debe observar los ejemplos y buscar características en común que permitan formar grupos.

Como resultado este tipo de aprendizaje genera un conjunto de descripciones de clases, que juntas cubren todas las clases y en particular describen a una sola clase.

Los algoritmos que se utilizaron en el desarrollo de este trabajo pertenecen al aprendizaje supervisado, en virtud de que en ella se seleccionan las clases y se proporcionan los ejemplos que describen a cada una

2.2 Minería de datos

La minería de datos consiste en la "explotación" de datos en bruto. Su objetivo, perseguido mediante la manipulación automática o semiautomática de los datos, es la obtención de información clave para conseguir beneficios, información más relevante y útil que los propios datos de partida.

La minería de datos se fundamenta en la intersección de diversas áreas de estudio, entre las que cabe destacar: análisis estadístico, bases de datos, inteligencia artificial y visualización gráfica.

Una buena definición de lo que es minería de datos puede ser la siguiente:

"*Es el empleo de algoritmos y procedimientos para sacar a la luz asociaciones, correlaciones, reglas, patrones e incluso excepciones interesantes o potencialmente útiles, desconocidos y escondidos en bases de datos (o almacenes de datos)".*

La importancia de la minería de datos crece, pues, con el tamaño de las bases de datos. En las pequeñas basta con técnicas estadísticas tradicionales y con aplicaciones relativamente sencillas.

A veces se ha apelado al nombre de "descubrimiento de conocimiento en bases de datos" (KDD: Knowledge Discovery in Data Bases) para hacer referencia a la minería de datos; sin embargo, muchos autores prefieren referirse al proceso de minería de datos como al de la aplicación de un algoritmo para extraer patrones de datos y a descubrimiento de conocimiento como al proceso completo (pre-procesamiento, minería, post-procesamiento).

En este sentido, KDD implica un proceso interactivo e iterativo, involucrando la aplicación de métodos de minería de datos, para extraer o identificar lo que se considera como conocimiento de acuerdo a la especificación de ciertos parámetros usando una base de datos, junto con pre-procesamientos, muestreo y transformaciones de la base de datos.

La meta de este proceso es justamente procesar automáticamente grandes cantidades de datos crudos, identificar los patrones más significativos y relevantes, y presentarlos como conocimiento apropiado para satisfacer las metas del usuario.

El proceso de descubrimiento de conocimiento en bases de datos involucra varios pasos:

1. Entender el dominio de aplicación, el conocimiento relevante a usar y las metas del usuario.
2. Seleccionar el conjunto de datos y enfocar la búsqueda en subconjuntos de variables o muestras de datos donde realizar el proceso de descubrimiento.
3. Filtrar (limpiar) y pre-procesar datos, diseñando una estrategia adecuada para manejar ruido, valores incompletos, secuencias de tiempo, etc.
4. Reducir datos y proyecciones para disminuir el número de variables a considerar.
5. Seleccionar la tarea de descubrimiento a realizar, por ejemplo: clasificación, agrupamiento, regresión, etc.
6. Seleccionar el o los algoritmos a utilizar.
7. Llevar a cabo el proceso de minería de datos.
8. Interpretar los resultados y posiblemente regresar a algún paso anterior. Esto puede involucrar repetir el proceso, quizás con otros datos, otros algoritmos, otras metas y otras estrategias.
9. Incorporar el conocimiento descubierto al sistema (normalmente para mejorarlo) lo cual puede incluir resolver conflictos potenciales con el conocimiento existente.

El proceso de minería en sí involucra ajustar modelos o determinar patrones a partir de datos. Este ajuste normalmente es de tipo estadístico, en el sentido que se permite un cierto ruido o error dentro del modelo.

Los algoritmos de minería de datos realizan en general tareas de predicción (de datos desconocidos) y de descripción (de patrones).

Las tareas principales en la minería de datos son:

- *Análisis de dependencias*: El valor de un elemento puede usarse para predecir el valor de otro. La dependencia puede ser probabilística, puede definir una red de dependencias o puede ser funcional. Se ha orientado mucho en los últimos años en el descubrimiento de redes bayesianas o causales en donde la dependencia se da a nivel estructural (dependencias e independencias entre variables) y cuantitativa (fuerza de las dependencias).
- *Identificación de clases (agrupamiento de registros en clases):* Identifica un conjunto finito de categorías o clusters que describen los datos (pueden ser exhaustivas y mutuamente exclusivas o jerárquicas y con superposiciones). Las clases pueden ser relevantes en sí o pueden servir como entradas a otros sistemas de aprendizaje. Se utilizan algoritmos de clustering. Normalmente el usuario tiene una buena capacidad de formar las clases y se han desarrollado herramientas visuales interactivas para ayudar al usuario.
- *Descripción de conceptos:* Se resume un cierto patrón. La descripción puede ser característica (qué registros son comunes entre clases) o discriminatoria (cómo difieren las clases). La mayoría de los sistemas de

aprendizaje encuentran descripciones de conceptos y están enfocados a clasificación: aprender una función que mapea (clasifica) un dato dentro de un conjunto de posibles clases predefinidas. Otra técnica relacionada es regresión: aprender una función que mapea un dato a una variable real. A veces se trata de encontrar descripciones compactas de subconjuntos de datos (media y varianza, leyes físicas) que los resuman de alguna forma.

- *Detección de desviaciones, casos extremos o anomalías:* Detectar los cambios más significativos en los datos con respecto a valores pasados o normales. Sirve para filtrar grandes volúmenes de datos que son menos probables de ser interesantes. El problema está en determinar cuándo una desviación es significativa para ser de interés.

Algunas de las técnicas más comúnmente empleadas en la minería de datos son:

- *Árboles de decisión y reglas de clasificación:* realizan cortes sobre una variable (lo cual limita su expresividad, pero facilita su comprensión). Generalmente se usan técnicas heurísticas en su construcción.
- *Métodos de clasificación y regresiones no-lineales:* tratan de ajustar combinaciones de funciones lineales y no-lineales, como las redes neuronales.
- *Métodos basados en ejemplos prototìpicos:* se hacen aproximaciones en base a los ejemplos o casos más conocidos. El problema es cómo determinar una medida de similitud adecuada.
- *Modelos gráficos de dependencias probabilísticas:* básicamente redes bayesianas, en donde la evaluación se basa en probabilidades.
- *Modelos relacionales:* Programación lógica inductiva (ILP), en donde la búsqueda del modelo se basa en lógica y heurísticas.

La minería de datos parece ser más efectiva cuando los datos tienen elementos que pueden permitir una interpretación y explicación en concordancia con la experiencia humana. Hay muchos métodos de hacer minería de datos. Una manera de clasificarlos es por las técnicas aplicadas. En este caso, pueden ser 6:

- Sin algoritmo de aprendizaje:
  Consultas (SQL)
  Procesamiento analítico en línea (OLAP)
  Análisis estadístico (Correlación, regresiones,...)
- Con algoritmo de aprendizaje:
  Redes neuronales y algoritmos genéticos
  Inducción de árboles y reglas
- Nuevos algoritmos:
  Inducción de reglas de asociación
  Inducción de clasificadores bayesianos

Otra manera de clasificar los métodos de hacer minería de datos es atendiendo a las funciones que realizan y a la clase de aplicaciones que pueden ser usadas. Las más típicas de tales funciones son: asociación, clasificación, agrupación y establecimiento de patrones secuenciales.

En bases de datos con tamaño suficiente, la tecnología de minería de datos puede generar nuevas oportunidades de negocio como las siguientes:

- Predicción automática de tendencias y desempeño: La minería de datos automatiza el proceso de encontrar información predictiva en grandes bases de datos. Un ejemplo típico de esto es el marketing dirigido. Se emplea información de los mensajes electrónicos promociónales pasados para identificar los objetivos, esto con el fin de maximizar el retorno de inversión en próximos mensajes promociónales.
- Descubrimiento automático de patrones previamente no conocidos: Las herramientas de minería de datos barren a través de las bases de datos e identifican patrones previamente desconocidos. Un ejemplo es el análisis de datos de ventas retenidas para identificar productos aparentemente no relacionados y que son comúnmente comprados juntos.

2.3 Técnicas de minería de datos

Las técnicas de minería de datos intentan obtener patrones o modelos a partir de los datos recopilados. Decidir si los modelos obtenidos son útiles o no suele requerir una valoración subjetiva por parte del usuario. Las técnicas de minería de datos se clasifican en dos grandes categorías: supervisadas o predictivas y no supervisadas o descriptivas.

Una técnica constituye el enfoque conceptual para extraer la información de los datos, y, en general es implementada por varios algoritmos. Cada algoritmo representa, en la práctica, la manera de desarrollar una determinada técnica paso a paso, de forma que es preciso un entendimiento de alto nivel de los algoritmos para saber cual es la técnica más apropiada para cada problema. Asimismo es preciso entender los parámetros y las características de los algoritmos para preparar los datos a analizar.

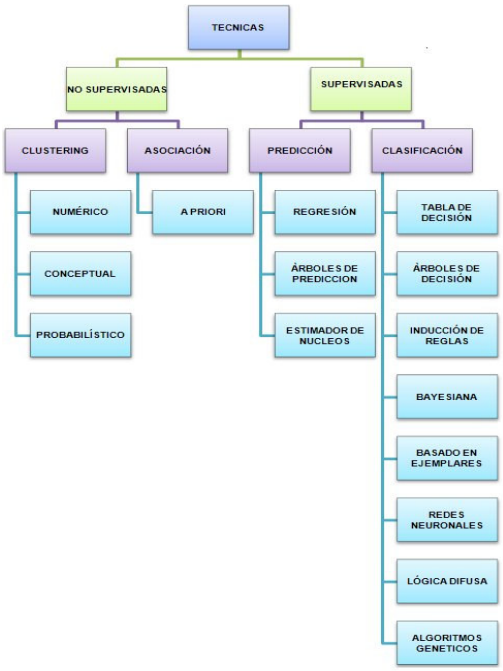

Figura 2.1: Técnicas de minería de datos

Las predicciones se utilizan para preveer el comportamiento futuro de algún tipo de entidad mientras que una descripción puede ayudar a su comprensión. De hecho, los modelos predictivos pueden ser descriptivos y los modelos descriptivos pueden emplearse para realizar predicciones. De esta forma, hay algoritmos o técnicas que pueden servir para distintos propósitos, por lo que la figura 2.1 únicamente representa para qué propósito son más utilizadas las técnicas. Por ejemplo, las redes de neuronas pueden servir para predicción, clasificación e incluso para aprendizaje no supervisado.

A continuación se describen algunos algoritmos de aprendizaje automático que han sido utilizados con éxito en la minería de datos y que serán utilizados para el desarrollo de esta tesis.

2.4 Árboles de Decisión

El aprendizaje de árboles de decisión está enmarcado como una metodología del aprendizaje supervisado. La representación que se utiliza para las descripciones del concepto adquirido es el árbol de decisión, que consiste en una representación del conocimiento relativamente simple y que es una de las causas por la que los procedimientos utilizados en su aprendizaje son más sencillos que los de sistemas que utilizan lenguajes de representación más potentes, como redes semánticas, representaciones en lógica de primer orden etc. No obstante, la potencia expresiva de los árboles de decisión es también menor que la de esos otros sistemas.

El aprendizaje de árboles de decisión suele ser más robusto frente al ruido y conceptualmente sencillo, aunque los sistemas que han resultado del perfeccionamiento y de la evolución de los más antiguos se complican con los procesos que incorporan para ganar fiabilidad. La mayoría de los sistemas de aprendizaje de árboles suelen ser no incrementales.

El primer sistema que construía árboles de decisión fue CLS (Concept Learning System) de Hunt, desarrollado en 1959 y depurado a lo largo de los años sesenta.

El CLS es un sistema desarrollado por psicólogos como un modelo del proceso cognitivo de formación de conceptos sencillos.

Su contribución fundamental fue la propia metodología pero no resultaba computacionalmente eficiente debido al método que empleaba en la extensión de los nodos.

En 1979 Quinlan desarrolla el sistema ID3, que él denominaría simplemente herramienta porque la consideraba experimental.

Conceptualmente es fiel a la metodología de CLS pero le aventaja en el método de expansión de los nodos, basado en una función que utiliza la medida de la información de Shannon. La versión definitiva, presentada por su autor Quinlan como un sistema de aprendizaje, es el sistema C4.5. La evolución comercial de ese sistema es otro denominado C5 del mismo autor, del que se puede obtener una versión de demostración restringida en cuanto a capacidades; por ejemplo, el número máximo de ejemplos de entrenamiento.

2.4.1 Representación de un árbol de decisión

Un árbol de decisión puede interpretarse esencialmente como una serie de reglas compactadas para su representación en forma de árbol. Dado un conjunto de ejemplos, estructurados como vectores de pares ordenados atributo-valor, de acuerdo con el formato general en el aprendizaje inductivo a partir de ejemplos, el concepto que estos sistemas adquieren durante el proceso de aprendizaje consiste en un árbol.

Cada eje está etiquetado con un par atributo-valor y las hojas con una clase, de forma que la trayectoria que determinan desde la raíz los pares de un ejemplo de entrenamiento alcanzan una hoja etiquetada normalmente con la clase del ejemplo. La clasificación de un ejemplo nuevo del que se desconoce su clase se hace con la misma técnica, solamente que en ese caso al atributo clase, cuyo valor se desconoce, se le asigna de acuerdo con la etiqueta de la hoja a la que se accede con ese ejemplo.

2.4.2 Problemas apropiados para este tipo de aprendizaje

Las características de los problemas apropiados para resolver mediante este aprendizaje dependen del sistema de aprendizaje específico utilizado, pero hay

una serie de ellas generales y comunes a la mayoría y que se describen a continuación:

- Que la representación de los ejemplos sea mediante vectores de pares atributo-valor, especialmente cuando los valores son disjuntos y en un número pequeño. Los sistemas actuales están preparados para tratar atributos con valores continuos, valores desconocidos e incluso valores con una distribución de probabilidad.
- Que el atributo que hace el papel de la clase sea de tipo discreto y con un número pequeño de valores, sin embargo existen sistemas que adquieren como concepto aprendido funciones con valores continuos.
- Que las descripciones del concepto adquirido deban ser expresadas en forma normal disyuntiva.
- Que posiblemente existan errores de clasificación en el conjunto de ejemplos de entrenamiento, así como valores desconocidos en algunos de los atributos.

En algunos ejemplos estos sistemas, por lo general, son robustos frente a los errores del tipo mencionado.

A continuación se presentan dos algoritmos de árboles de decisión, ambos diseñados por Quinlan, los sistemas ID3 y C4.5.

2.4.3 Sistema ID3

El sistema ID3 es un algoritmo simple y, sin embargo, potente, cuya misión es la elaboración de un árbol de decisión. El procedimiento para generar un árbol de decisión consiste, como se comentó anteriormente en seleccionar un atributo como raíz del árbol y crear una rama con cada uno de los posibles valores de dicho atributo. Con cada rama resultante (nuevo nodo del árbol), se realiza el mismo proceso, esto es, se selecciona otro atributo y se genera una nueva rama para cada posible valor del atributo.

Este procedimiento continúa hasta que los ejemplos se clasifiquen a través de uno de los caminos del árbol. El nodo final de cada camino será un nodo hoja, al que se le asignará la clase correspondiente. Así, el objetivo de los árboles de decisión es obtener reglas o relaciones que permitan clasificar a partir de los atributos.

En cada nodo del árbol de decisión se debe seleccionar un atributo para seguir dividiendo, y el criterio que se toma para elegirlo es: se selecciona el atributo que mejor separe (ordene) los ejemplos de acuerdo a las clases. Para ello se emplea la entropía, que es una medida de cómo está ordenado el universo.

La teoría de la información (basada en la entropía) calcula el número de bits (información, preguntas sobre atributos) que hace falta suministrar para conocer la clase a la que pertenece un ejemplo.

Cuanto menor sea el valor de la entropía, menor será la incertidumbre y más útil será el atributo para la clasificación. La definición de entropía que da Shannon en su teoría de la Información (1948) es:

Dado un conjunto de eventos $A=\{A1,A2,...,An\}$, con probabilidades $\{ p_1, p_2,..., p_n \}$, la información en el conocimiento de un suceso A_i (bits) se define en la ecuación 2.1, mientras que la información media de A (bits) se muestra en la ecuación 2.2.

$$I(A_i) = \log_2\left(\frac{1}{p_i}\right) = -\log_2(p_i) \tag{2.1}$$

$$I(A) = \sum_{i=1}^{n} p_i I(A_i) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{2.2}$$

Si aplicamos la entropía a los problemas de clasificación se puede medir lo que se discrimina (se gana por usar) un atributo A_i empleando para ello la ecuación 2.3, en la que se define la ganancia de información.

$$G(A_i) = I(A) - I(A_i) \tag{2.3}$$

Siendo I la información antes de utilizar el atributo e $I(A_i)$ la información después de utilizarlo. Se definen ambas en las ecuaciones 2.4 y 2.5.

$$I\_division(n) = -\sum_{c=1}^{nc} \frac{n_c}{n} \log_2\left(\frac{n_c}{n}\right) \tag{2.4}$$

$$I(A_i) = \sum_{j=1}^{nv(A_i)} \frac{n_{ij}}{n} I_{ij}; \quad I_{ij} = -\sum_{k=1}^{nc} \frac{n_{ijk}}{n_{ij}} \log_2\left(\frac{n_{ijk}}{n_{ij}}\right) \tag{2.5}$$

En estas ecuaciones *nc* será el número de clases y n_c el número de ejemplares de la clase *c*, siendo *n* el número total de ejemplos. Será $nv(A_i)$ el número de valores del atributo A_i, n_{ij} el número de ejemplos con el valor *j* en A_i y n_{ijk} el número de ejemplos con valor *j* en A_i y que pertenecen a la clase *k*.

Una vez explicada la heurística empleada para seleccionar el mejor atributo en un nodo del árbol de decisión, se muestra el algoritmo ID3:

1. Seleccionar el atributo A_i que maximice la ganancia $G(A_i)$.
2. Crear un nodo para ese atributo con tantos sucesores como valores tenga.
3. Introducir los ejemplos en los sucesores según el valor que tenga el atributo A_i.
4. Por cada sucesor:
   (a) Si sólo hay ejemplos de una clase, *Ck*, entonces etiquetarlo con *Ck*.
   (b) Si no, llamar a ID3 con una tabla formada por los ejemplos de ese nodo, eliminando la columna del atributo A_i.

2.4.4 SISTEMA C4.5

El ID3 es capaz de tratar con atributos cuyos valores sean discretos o continuos. En el primer caso, el árbol de decisión generado tendrá tantas ramas como valores posibles tome el atributo. Si los valores del atributo son continuos, el ID3 no clasifica correctamente los ejemplos dados. Por ello, Quinlan propuso el C4.5, como extensión del ID3, que permite:

1. Empleo del concepto razón de ganancia (GR, [Gain Ratio])
2. Construir árboles de decisión cuando algunos de los ejemplos presentan valores desconocidos para algunos de los atributos.
3. Trabajar con atributos que presenten valores continuos.
4. La poda de los árboles de decisión.
5. Obtención de Reglas de Clasificación.

Razón de Ganancia

El test basado en el criterio de maximizar la ganancia tiene como sesgo la elección de atributos con muchos valores. Esto es debido a que cuanto más fina sea la participación producida por los valores del atributo, normalmente, la incertidumbre o entropía en cada nuevo nodo será menor, y por lo tanto también será menor la media de la entropía a ese nivel.

C4.5 modifica el criterio de selección del atributo empleando en lugar de la ganancia la razón de ganancia, cuya definición se muestra en la ecuación 2.6.

$$GR(A_i) = \frac{G(A_i)}{I(Division\_A_i)} = \frac{G(A_i)}{-\sum_{j=1}^{nv(A_i)} \frac{n_{ij}}{n} \log_2\left(\frac{n_{ij}}{n}\right)} \tag{2.6}$$

Al término $I(Division\_A_i)$ se le denomina información de ruptura. En esta medida cuando n_{ij} tiende a n, el denominador se hace 0. Esto es un problema aunque según Quinlan, la razón de ganancia elimina el sesgo.

Valores Desconocidos

El sistema C4.5 admite ejemplos con atributos desconocidos tanto en el proceso de aprendizaje como en el de validación. Para calcular, durante el proceso de aprendizaje, la razón de ganancia de un atributo con valores desconocidos, se redefinen sus dos términos, la ganancia, ecuación 2.7, y la información de ruptura, ecuación 2.8.

$$G(A_i) = \frac{n_{ic}}{n}\left(I - I(A_i)\right) \tag{2.7}$$

$$I(Division\_A_i) = -\left(\sum_{j=1}^{nv(A_i)} \frac{n_{ij}}{n} \log_2 \frac{n_{id}}{n}\right) \tag{2.8}$$

En estas ecuaciones, n_{ic} es el número de ejemplos con el atributo *i* conocido, y n_{id} el número de ejemplos con valor desconocido en el mismo atributo. Además, para el cálculo de las entropías $I(A_i)$ se tendrán en cuenta únicamente los ejemplos en los que el atributo A_i tenga un valor definido.

No se toma el valor desconocido como significativo, sino que se supone una distribución probabilística del atributo de acuerdo con los valores de los ejemplos en la muestra de entrenamiento. Cuando se entrena, los casos con valores desconocidos se distribuyen con pesos de acuerdo a la frecuencia de aparición de cada posible valor del atributo en el resto de ejemplos de entrenamiento.

El peso ω_{ij} con que un ejemplo *i* se distribuiría desde un nodo etiquetado con el atributo *A* hacia el hijo con valor *j* en dicho atributo se calcula mediante la ecuación 2.9, en la que ω_i es el peso del ejemplo *i* al llegar al nodo, esto es, antes de distribuirse, y $p(A=j)$ la suma de pesos de todos los ejemplos del nodo con valor *j* en el atributo *A* entre la suma total de pesos de todos los ejemplos del nodo (ω).

$$\omega_{ij} = \omega_i p(A = j) = \omega_i \frac{\omega_{a=j}}{\omega} \tag{2.9}$$

En cuanto a la clasificación de un ejemplo de test, si se alcanza un nodo con un atributo que el ejemplo no tiene (desconocido), se distribuye el ejemplo (divide) en tantos casos como valores tenga el atributo, y se da un peso a cada resultado con el mismo criterio que en el caso del entrenamiento: la frecuencia de aparición de cada posible valor del atributo en los ejemplos de entrenamiento. El resultado de esta técnica es una clasificación con probabilidades, correspondientes a la distribución de ejemplos en cada nodo hoja.

Atributos Continuos

El tratamiento que realiza C4.5 de los atributos continuos está basado en la ganancia de información, al igual que ocurre con los atributos discretos. Si un atributo continuo A_i presenta los valores ordenados $v_1, v_2, ..., v_n$, se comprueba cuál de los valores $z_i = (v_i + v_{i+1})/2$; $1 \le j < n$, supone una ruptura del intervalo $[v_1, v_n]$ en dos subintervalos $[v_1, z_j]$ y $(z_j, v_n]$ con mayor ganancia de información.

El atributo continuo, ahora con dos únicos valores posibles, entrará en competencia con el resto de los atributos disponibles para expandir el nodo.

El algoritmo del método C4.5 para la construcción de árboles de decisión a grandes rasgos muy similar al del ID3. Varía en la manera en que realiza las pruebas sobre los atributos.

Función C4.5

(R: conjunto de atributos no clasificadores, C: atributo clasificador, S: conjunto de entrenamiento) Devuelve un árbol de decisión;

Comienzo

Si S está vacío, devolver un único nodo con Valor Falla;

Si todos los registros de S tienen el mismo valor para el atributo clasificador,

Devolver un único nodo con dicho valor;

Si R está vacío, entonces

Devolver un único nodo con el valor más frecuente del atributo clasificador en los registros de S

[Nota: habrá errores, es decir, registros que no estarán bien clasificados en este caso];

Si R no está vacío, entonces

D! Atributo con mayor Proporción de Ganancia (D, S) entre los atributos de R;

Sean {dj j=1,2,..., m} los valores del atributo D;

Sean {Sj j=1,2,..., m} los subconjuntos de S correspondientes a los valores de dj respectivamente;

Devolver un árbol con la raíz nombrada como D y con los arcos nombrados d1, d2,…, dm que van respectivamente a los árboles C4.5(R-{D}, C, S1), C4.5(R-{D}, C, S2),..., C4.5(R-{D}, C, Sm);

Fin

Poda del árbol de decisión

El árbol de decisión ha sido construido a partir de un conjunto de ejemplos, por tanto, reflejará correctamente todo el grupo de casos. Sin embargo, como esos ejemplos pueden ser muy diferentes entre sí, el árbol resultante puede llegar a ser bastante complejo, con trayectorias largas y muy desiguales. Para facilitar la comprensión del árbol puede realizarse una poda del mismo. C4.5 efectúa la poda después de haber desarrollado el árbol completo (post-poda), a diferencia de otros sistemas que realizan la construcción del árbol y la poda a la vez (pre-poda); es decir, estiman la necesidad de seguir desarrollando un nodo aunque no posea el carácter de hoja.

En C4.5 el proceso de podado comienza en los nodos hoja y recursivamente continúa hasta llegar al nodo raíz. Se consideran dos operaciones de poda en C4.5: reemplazo de sub-árbol por hoja (subtree replacement) y elevación de sub-árbol (subtree raising). En la figura 2.3 se muestra en lo que consiste cada tipo de poda.

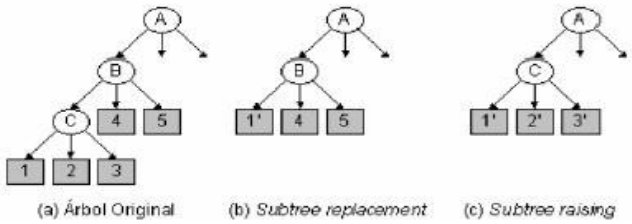

Figura 2.2: Tipos de operaciones de poda en C4.5

En la figura 2.2 tenemos el árbol original antes del podado (a), y las dos posibles acciones de podado a realizar sobre el nodo interno C. En (b) se realiza subtree replacement, en cuyo caso el nodo C es reemplazado por uno de sus subárboles. Por último, en (c) se realiza subtree raising: El nodo B es sustituido por el subárbol con raíz C. En este último caso hay que tener en cuenta que habrá que reclasificar de nuevo los ejemplos a partir del nodo C. Además, subtree raising es muy costoso computacionalmente hablando, por lo que se suele restringir su uso al camino más largo a partir del nodo (hasta la hoja) que estamos podando. Como se comentó anteriormente, el proceso de podado comienza en las hojas y continúa hacia la raíz pero, la cuestión es cómo decidir reemplazar un nodo interno por una hoja (replacement) o reemplazar un nodo interno por uno de sus nodos hijo (raising). Lo que se hace es comparar el error estimado de clasificación en el nodo en el que nos encontramos y compararlo con el error en cada uno de sus hijos y en su padre para realizar alguna de las operaciones o ninguna.

A continuación se muestra el pseudocódigo del proceso de podado que se emplea en C4.5.

```
Podar (raíz) {
    Si raíz No es HOJA Entonces Para cada hijo H de raíz Hacer
            Podar (H)
            Obtener Brazo más largo (B) de raíz // raising
             ErrorBrazo = EstimarErrorArbol (B, raìz.ejemplos)
            ErrorHoja = EstimarError (raíz, raìz.ejemplos) // replacement
            ErrorÁrbol = EstimarErrorArbol (raíz, raìz.ejemplos)
            Si ErrorHoja <= ErrorÁrbol Entonces // replacement raíz es Hoja
                    Fin Poda
            Si ErrorBrazo <= ErrorÁrbol Entonces // raising
                    raíz = B Podar (raíz)
}
EstimarErrorArbol (raíz, ejemplos) {
    Si raíz es HOJA Entonces
            EstimarError (raíz, ejemplos)
    Si no
            Distribuir los ejemplos (ej[]) en los brazos
            Para cada brazo (B)
                    error = error + EstimarErrorArbol (B, ej[B])
}
```

De esta forma, la elevación del subárbol se emplea únicamente para el subárbol más largo. Además, para estimar su error se emplean los ejemplos de entrenamiento, pero los del nodo origen, ya que si se eleva deberá clasificarlos él. En cuanto a la función EstimarError, es la función que estima el error de clasificación de una hoja del árbol.

Así, para tomar la decisión debemos estimar el error de clasificación en un nodo determinado para un conjunto de test independiente.

Habrá que estimarlo tanto para los nodos hoja como para los internos (suma de errores de clasificación de sus hijos).

No se puede tomar como dato el error de clasificación en el conjunto de entrenamiento dado que, lógicamente, el error se subestimaría.

Una técnica para estimar el error de clasificación es la denominada reducederror pruning, que consiste en dividir el conjunto de entrenamiento en n subconjuntos $n-1$ de los cuáles servirán realmente para el entrenamiento del sistema y 1 para la estimación del error. Sin embargo, el problema es que la construcción del clasificador se lleva a cabo con menos ejemplos. Esta no es la técnica empleada en C4.5.

La técnica empleada en C4.5 consiste en estimar el error de clasificación basándose en los propios ejemplos de entrenamiento. Para ello, en el nodo donde queramos estimar el error de clasificación, se toma la clase mayoritaria de sus ejemplos como clase representante. Esto implica que habrá E errores de clasificación de un total de N ejemplos que se clasifican a través de dicho nodo. El error observado será $f=E/N$, siendo q la probabilidad de error de clasificación del nodo y p=1-q la probabilidad de éxito.

Se supone que la función f sigue una distribución binomial de parámetro q. Y lo que se desea obtener es el error e, que será la probabilidad del extremo superior con un intervalo $[f-z, f+z]$ de confianza c. Dado que se trata de una distribución binomial, se obtendrá e mediante las ecuaciones 2.10 y 2.11.

$$P\left[\frac{f-q}{q(1-q)/N} \le z\right] = c \tag{2.10}$$

$$e = \left(\frac{f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}\right) \tag{2.11}$$

Como factor c (factor de confianza) se suele emplear en C4.5 el 25%, dado que es el que mejores resultados suele dar y que corresponde a un z=0.69.

2.5 Clasificación Bayesiana

Los clasificadores Bayesianos son clasificadores estadísticos, que pueden predecir tanto las probabilidades del número de miembros de clase, como la probabilidad de que una muestra dada pertenezca a una clase particular. La clasificación Bayesiana se basa en el teorema de Bayes, y los clasificadores Bayesianos han demostrado una alta exactitud y velocidad en la construcción de modelos cuando se han aplicado a grandes bases de datos. Diferentes estudios comparando los algoritmos de clasificación han determinado que un clasificador Bayesiano sencillo conocido como el clasificador “naive Bayesiano” es

comparable en rendimiento a un árbol de decisión y a clasificadores de redes de neuronas.

A continuación se explica los fundamentos de los clasificadores bayesianos y, más concretamente, del clasificador naive Bayesiano.

2.5.1 Clasificador Naive Bayesiano

Lo que normalmente se quiere saber en aprendizaje es cuál es la mejor hipótesis (más probable) dados los datos. Si denotamos $P(D)$ como la probabilidad a priori de los datos (i.e., cuales datos son más probables que otros), $P(D|h)$ la probabilidad de los datos dada una hipótesis, lo que queremos estimar es: $P(h|D)$, la probabilidad posterior de h dados los datos. Esto se puede estimar con el teorema de Bayes, ecuación 2.12.

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)} \tag{2.12}$$

Para estimar la hipótesis más probable se busca el mayor $P(h \mid D)$ como se muestra en la ecuación 2.13.

$$\begin{aligned} h_{MAP} &= \arg\max_{h \varepsilon H}(P(h \mid D)) \\ &= \arg\max_{h \varepsilon H}\left(\frac{P(D \mid h)P(h)}{P(D)}\right) \\ &= \arg\max_{h \varepsilon H}(P(D \mid h)P(h)) \end{aligned} \tag{2.13}$$

Ya que $P(D)$ es una constante independiente de h. Si se asume que todas las hipótesis son igualmente probables, entonces resulta la hipótesis de máxima verosimilitud (ML) de la ecuación 2.14.

$$h_{ML} = \arg\max_{h \varepsilon H}(P(D \mid h)) \tag{2.14}$$

El clasificador naive Bayesiano se utiliza cuando se quiere clasificar un ejemplo descrito por un conjunto de atributos ($a|s$) en un conjunto finito de clases (V). Clasificar un nuevo ejemplo de acuerdo con el valor más probable dados los valores de sus atributos. Si se aplica 2.13 al problema de la clasificación se obtendrá la ecuación 2.15.

$$\begin{aligned} v_{MAP} &= \arg\max_{v_j \varepsilon V}(P(v_j \mid a_1,...,a_n)) \\ &= \arg\max_{v_j \varepsilon V}\left(\frac{P(a_1,...,a_n \mid v_j)P(v_j)}{P(a_1,...,a_n)}\right) \\ &= \arg\max_{v_j \varepsilon V}(P(a_1,...,a_n \mid v_j)P(v_j)) \end{aligned} \tag{2.15}$$

Además, el clasificador naive Bayesiano asume que los valores de los atributos son condicionalmente independientes dado el valor de la clase, por lo que se hace cierta la ecuación 2.16 y con ella la 2.17.

$$P(a_1,...,a_n \mid v_j) = \prod_i P(a_i \mid v_j) \tag{2.16}$$

$$P(v_j \mid a_1,...,a_n) = \prod_i P(v_j) i P(a_i \mid v_j) \tag{2.17}$$

Los clasificadores naive Bayesianos asumen que el efecto de un valor del atributo en una clase dada es independiente de los valores de los otros atributos. Esta suposición se llama “independencia condicional de clase”.

Ésta simplifica los cálculos involucrados y, en este sentido, es considerado "ingenuo”. Esta asunción es una simplificación de la realidad. A pesar del nombre del clasificador y de la simplificación realizada, el naive Bayesiano funciona muy bien, sobre todo cuando se filtra el conjunto de atributos seleccionado para eliminar redundancia, con lo que se elimina también dependencia entre datos.

Algo que puede ocurrir durante el entrenamiento con este clasificador es que para cada valor de cada atributo no se encuentren ejemplos para todas las clases. Supóngase que para el atributo ai y el valor j de dicho atributo no hay ningún ejemplo de entrenamiento con clase *k*. En este caso, $P(a_{ij} \mid k) = 0$. Esto hace que si se intenta clasificar cualquier ejemplo con el par atributo-valor a_{ij}, la probabilidad asociada para la clase *k* será siempre 0, ya que hay que realizar el productorio de las probabilidades condicionadas para todos los atributos de la instancia. Para resolver este problema se parte de que las probabilidades se contabilizan a partir de las frecuencias de aparición de cada evento o, en nuestro caso, las frecuencias de aparición de cada terna atributo-valor-clase.

El estimador de Laplace, consiste en comenzar a contabilizar la frecuencia de aparición de cada terna a partir del 1 y no del 0, con lo que ninguna probabilidad condicionada será igual a 0.

Una ventaja de este clasificador es la cuestión de los valores perdidos o desconocidos: en el clasificador naive Bayesiano si se intenta clasificar un ejemplo con un atributo sin valor simplemente el atributo en cuestión no entra en el productorio (ecuación 2.17) que sirve para calcular las probabilidades.

Respecto a los atributos numéricos, se suele suponer que siguen una distribución Normal o Gaussiana.

Para estos atributos se calcula la media μ y la desviación típica σ obteniendo los dos parámetros de la distribución $N(\mu,\sigma)$, que sigue la expresión de la ecuación 2.18, donde el parámetro *x* será el valor del atributo numérico en el ejemplo que se quiere clasificar.

$$f(x) = \frac{1}{\sqrt[2]{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.18}$$

2.6 Multiclasificadores

Los multiclasificadores son conjuntos de clasificadores diferentes que realizan predicciones que se fusionan y se obtiene como resultado la combinación de cada una de ellas. El hecho o la idea de combinación hace que los multiclasificadores sean citados en la literatura a través de distintos términos, entre ellos: métodos de ensamble, modelos múltiples, sistemas de múltiples clasificadores, combinación de clasificadores, integración de clasificadores, mezcla de expertos, comité de decisión, etc.

Se dividen en dos grupos: los métodos de ensamble o ensamblaje y los métodos híbridos. Los primeros se refieren a aquellos conjuntos de modelos que se combinan para crear un nuevo modelo, empleando para ello la misma técnica de aprendizaje.

En el caso de los segundos, son combinaciones de algoritmos de aprendizaje que crean a su vez nuevas técnicas de aprendizaje híbridas. Los multiclasificadores se distinguen unos de otros atendiendo a diversas características. Entre ellas podemos citar: el número de clasificadores individuales acoplados, el tipo de cada clasificador (redes neuronales, árboles de decisión, vecino más cercano, etc.), las características de los subconjuntos usados por cada clasificador del conjunto, la agregación de las decisiones particulares (voto mayoritario, asignación de pesos, subespacio de mejor comportamiento, funciones de promedio, máximo, mínimo, producto, etc.) y el tamaño y la naturaleza de los conjuntos de datos de entrenamiento para los clasificadores.

La evaluación y comparación entre los distintos multiclasificadores se establece a través de indicadores de rendimiento que incluyen el grado de generalización, aprendizaje, clasificación correcta e incorrecta y el tiempo real de ejecución.

2.6.1 Bagging

Bagging es un método propuesto por Breiman y está basado en los conceptos de bootstrapping y agregación, de esta forma se reincorporan los beneficios de ambos y se le da nombre al método (Bootstrap AGGregatING). Boostrapping se basa en la generación de conjuntos de entrenamientos aleatorios con reemplazamiento. Una muestra bootstrap se genera al muestrear uniformemente instancias del conjunto de entrenamiento de manera aleatoria. De modo que se crean tantas muestras bootstrap como clasificadores existan, de ahí que cada clasificador se entrena con una réplica bootstrap, véase la figura 2.6.

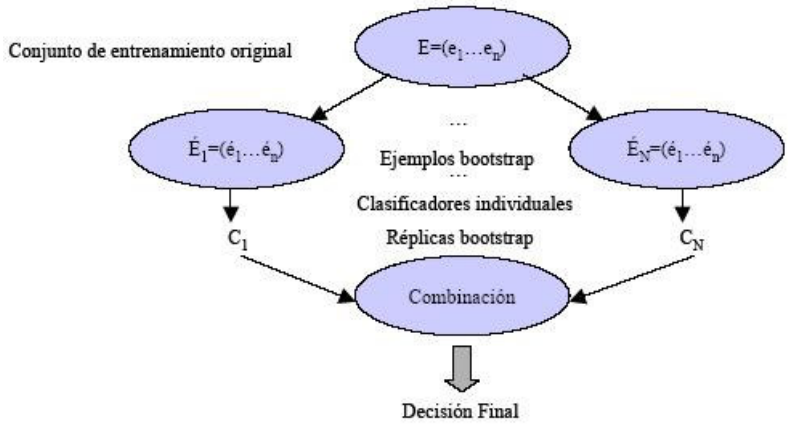

Figura 2.3: Esquema del método bagging

El método consiste en que los clasificadores individuales acoplados durante K iteraciones extraigan cada vez una muestra aleatoria del subconjunto É de n ejemplos con reemplazamiento a partir del conjunto original de ejemplos (pueden haber ejemplos repetidos); el subconjunto extraído aprende un modelo (técnica de aprendizaje a partir de una evidencia). Para clasificar un ejemplo e, se predice la clase de ese ejemplo para cada clasificador y se selecciona la clase con mayor voto.

2.6.2 Comités de validación cruzada (Cross-Validated Committees)

La idea es dividir los datos de entrenamiento en varios conjuntos. Los conjuntos no tienen que ser necesariamente mutuamente exclusivos, ellos pueden compartir parte del conjunto (solaparse). Esta idea es similar a los métodos de remuestreo como la validación cruzada (crossvalidation) que es utilizado en estadística para estimar el error de un predictor a partir de los datos disponibles. En el contexto de *comité*, esta técnica se reforma para construir conjuntos de entrenamiento diferentes del conjunto original, de ahí el nombre de comités de validación cruzada (CVC). El algoritmo primero genera réplicas a partir del conjunto de entrenamiento original mediante la exclusión de un fragmento de los datos. Se denota D como los datos originales y D^{-v} denota los datos con el subconjunto v excluido. El procedimiento consiste en ir rotando para que cada elemento esté por lo menos una vez como fragmento excluido.

Se generan réplicas $D_1^{-v1},...,D_k^{vk}$ y se entrena a cada clasificador que forma parte del comité, véase la figura 2.7.

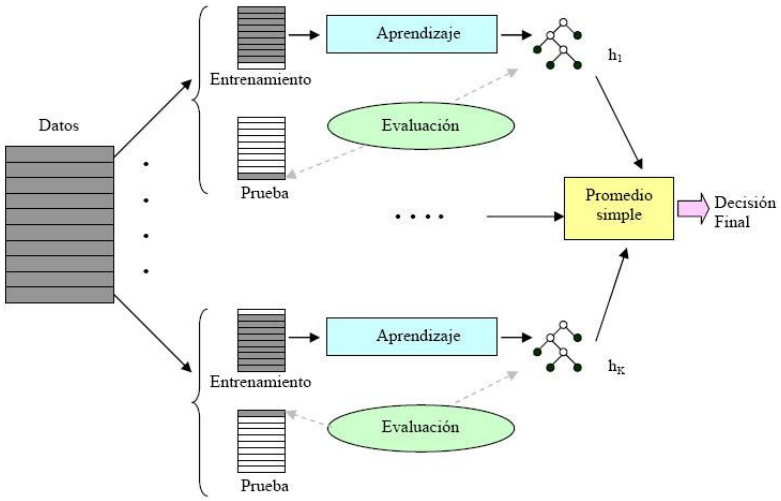

Figura 2.4: Algoritmo de Comités de Validación Cruzada

Los estimadores de error se basan en calcular la proporción de las instancias incorrectamente etiquetadas por el clasificador. Una vez construido el clasificador d y dado una instancia (X_i,c_i):

$$\Delta(X_i,c_i)=\begin{cases}1; \text{ si } d(X_i)\neq c_i \text{ (error)}\\ 0;\text{no } d(X_i)=c_i \text{ (acierto)}\end{cases}$$

Para todo k, k=1,2,..., K, se construye un clasificador d_k, usando D–D_k como conjunto de aprendizaje. Como ninguna de las instancias de D_k se ha usado para construir d_k, el estimador mediante el conjunto de prueba de d_k es $R(d_k)$:

$$R(d_k)=\frac{1}{|D_k|}\sum_{(X_i,c_i)\in D_k} D(X_i,c_i) \tag{2.19}$$

Donde Δ se evalúa sobre d_k y D_k es el conjunto de prueba. Al finalizar se obtiene K clasificadores, d_k, con sus correspondientes estimaciones de error $R(d_k)$.

Posteriormente, se usa el mismo procedimiento para construir el clasificador d con todas las instancias de D. Como cada d_k se construye con D–D_k, si K es grande, $|D-D_k|=N(1-\frac{1}{k})\approx|T|$.

Este procedimiento es estable (todos los clasificadores d_k tienen una tasa de error aproximadamente igual a la del clasificador d). Cuando K=N, el estimador por validación cruzada con N (tamaño del conjunto de aprendizaje) conjuntos se conoce como el estimador que deja uno fuera (del inglés leave-one-out).

Para cada *n*, *n*=1,2,...,*N* el *n–esimo* ejemplo es descartado y el clasificador se construye utilizando los restantes *N*–1 ejemplos. Entonces, el ejemplo descartado se usa para prueba y se estima el error mediante la ecuación:

$$R^{cv}(d) = \frac{1}{K}\sum_{k=1}^{K} R(d_k) \tag{2.20}$$

El algoritmo:

1. Dividir los datos en v-fragmentos *d*1,...,*dv*
2. Leave-one-out (dejar un fragmento fuera) d_k y entrenar la red h_k con el resto de los datos $(D - d_k)$
3. Usar d_k como un criterio de parada de la validación, si es necesario
4. Construir un comité (unir los resultados de los clasificadores) a partir de las redes usando un procedimiento de promedio simple.

Este algoritmo requiere de un gran esfuerzo computacional, ya que todos los ejemplos de D se usan para construir los d_k, y cada uno de ellos se usa exactamente una vez para prueba. No obstante, es adecuado para subconjuntos de pequeño tamaño.

2.7 Métodos de validación

La estrategia para validar un sistema de aprendizaje depende esencialmente de la manera en que dicha división es realizada. Algunos autores utilizan el mismo conjunto para el entrenamiento y el test, lo que produce una tasa de error casi siempre menor a la real y a menudo una estimación demasiado optimista. Para que la estimación sea válida, los conjuntos de entrenamiento y test deben ser independientes, o al menos diferentes. Los métodos de validación más destacados son:

- *Validación cruzada.* La validación cruzada con *k* conjuntos (*k*-fold cross validation) es atribuida a M. Stone. Consiste en dividir los datos en *k* subconjuntos de ejemplos con aproximadamente igual tamaño y evaluar el sistema *k* veces. En cada evaluación, se deja uno de los subconjuntos para el test y se entrena el sistema con los *k*–1 restantes. Así, el error estimado es la media de las *k* tasas obtenidas. A partir de este método de validación cruzada básico, se han desarrollado variantes para aproximar mejor la tasa de error.
- *Validación cruzada completa*. El método de validación cruzada completa (complete *k–fold* cross validation) realiza una exploración completa de todos las posibles combinaciones de $\frac{N}{K}$ ejemplos en el conjunto de test, donde *N* es el número total de ejemplos, dejando el resto para el

entrenamiento. En concreto se llevan a cabo $\binom{N}{N/K}$ evaluaciones, lo cual resulta extremadamente costoso computacionalmente.

- *Validación cruzada estratificada*. Una variante del método básico, denominada validación cruzada estratificada (stratified k-fold cross validation), distribuye los ejemplos intentando mantener la misma proporción de instancias de cada clase en el conjunto de entrenamiento y en el de test. Éste es quizá el método de validación a más recomendable, ya que mantiene para las evaluaciones la misma proporción de clases del conjunto total de datos, además de no aumentar significativamente el coste computacional respecto al método original.
- *Validación dejando uno fuera*. Este método es la validación cruzada llevada al extremo, es decir, tomando *k* igual al número de ejemplos (*N*) del conjunto de datos. Así, el clasificador entrena con *N*–1 ejemplos, dejando uno fuera para realizar el test, de ahí su nombre (leave–one–out). Además de la elevada varianza de la tasa de error obtenida, el mayor inconveniente de este método es su alto coste computacional, por lo que no es recomendable su uso con más de 100 ejemplos.
- *Split Sample*. Split sample es un tipo de validación no cruzada donde sólo existe un conjunto de test sobre el cual se estima el error. El tamaño dicho conjunto suele rondar entre el 20% y 30% del conjunto original, empleándose el resto para entrenamiento. La selección de los ejemplos para cada conjunto es aleatoria, aunque es aconsejable asegurar la misma proporción de ejemplos de distintas clases en ambos.
- *Bootstrapping*. Existen varias formas de aplicar la validación bootstrapping o validación por secuencia. La más simple consiste en realizar un muestreo aleatorio con reemplazo en el conjunto de datos, copiando los ejemplos seleccionados en el conjunto de entrenamiento hasta que éste alcance el tamaño del original. El conjunto de test lo conforman aquellos ejemplos no incluidos en el entrenamiento. Aunque la tasa de error de este conjunto es un estimador del error real, lo habitual es repetir el proceso varias veces y calcular la media.

Capítulo 3

EXPERIMENTOS

En este capitulo analizaré los resultados obtenidos de las pruebas realizadas con la herramienta Weka que fue la que seleccione por que cuenta con los algoritmos estudiados en este trabajo, además de que cuenta con licencia GPL lo cual ha inducido para que sea una de las más utilizadas en el área de minería de datos en los últimos años.

En la sección 3.1 se muestra una descripción de los datos de como están divididos y las clases que se utilizaron en cada uno de las pruebas. En el apartado 3.2 daré una idea de como se deben interpretar cada una de los cuadros que se muestran en las secciones posteriores y explicaré la forma en la que se llevarán a cabo cada una de las pruebas.

Los experimentos están divididos en dos partes que son: en la primera se realizan tomando en cuenta todos los atributos con los que se cuentan y en la segunda solo los más importantes. Estos a su vez están divididos en base a los algoritmos seleccionados (C4.5, Naive Bayes, Bagging y RandomCommitte) en donde utilice los siguientes métodos de validación, siendo éstos los siguientes: validación cruzada a 10 folds, Split Sample 66% y utilizando un conjunto de datos de entrenamiento y otro de prueba; en cada apartado hice una breve explicación de los resultados obtenidos los cuales complementaré en el capitulo 4.

3.1 Descripción de los datos

Los datos que utilice son el historial de los precios de cierre de las acciones de el Índice Promedio Industrial Dow Jones, también conocido como DJIA, Dow 30, o Dow, que es el principal índice bursátil creado por Charles Dow, editor del periódico The Wall Street Journal durante el Siglo XIX, y co-fundador de la empresa Dow Jones & Company. Es el índice más antiguo de Estados Unidos. Mide el desempeño de las 30 mayores empresas transadas en la bolsa de Estados Unidos.

Aunque nació con el nombre "industrial", lo cierto es que en la actualidad sus componentes tienen poca relación con la industria pesada, debido al auge de las compañías financieras y de informática.

Para compensar los efectos de divisiones de acciones y otros ajustes, el promedio es calculado a escala en relación al tamaño e importancia de una empresa.

Previamente, el índice se calculaba dividiendo la suma total en dinero del valor de las acciones, dividido por el número de acciones existentes. Sin embargo, para hacerlo más representativo esto se modificó y se pasó a dividir el total del valor de las acciones por una fórmula matemática que otorga mayor "peso" a las empresas más grandes.

Por ello, por ejemplo si en la Bolsa de Valores de Nueva York caen los títulos de Microsoft, estos tienen un impacto mucho mayor que la caída en el valor de la acción de una empresa con una capitalización de mercado más pequeña.

Entre los sectores económicos que abarca DJIA se encuentra Bienes de Capital con el 23,64%, tecnología 16,31%, producción de venta no Cíclica 13,81%, Salud 10,73%, producción de Venta Cíclica 9,55%, financiero 8,61%, producción de primera necesidad 7,40%, energía 5,32%, Servicios de comunicaciones 4,33% y Otros 0,03% ver figura 3.1.

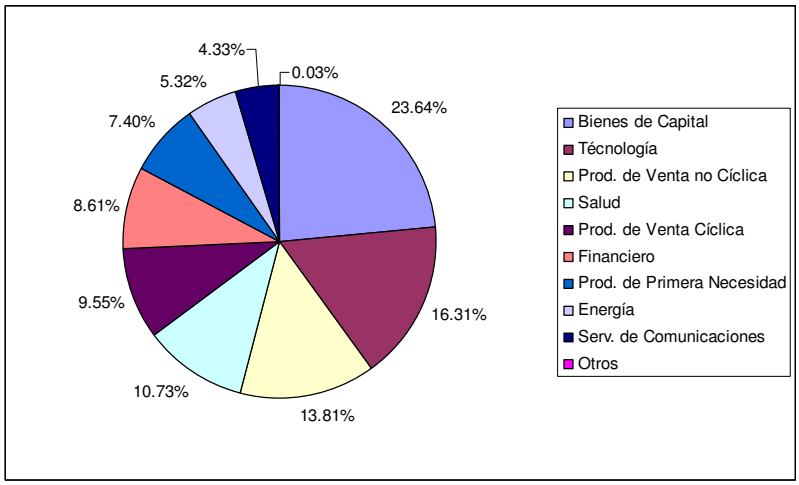

Figura 3.1

Los datos históricos que manejé para el desarrollo de este trabajo abarcan del 7 de abril de 1969 al 29 de octubre de 1984, estos datos fueron divididos en dos archivos las primeras 2800 referencias equivalentes al periodo del 7 de Abril de 1969 al 5 de mayo de 1980 que forman los datos de prueba y el segundo archivo comprendido del 10 de Abril de 1981 al 29 de octubre de 1984 que es el conjunto de entrenamiento. La figura 3.2 presenta una gráfica en la que se aprecia como están divididos los archivos con los que se realizarán los experimentos.

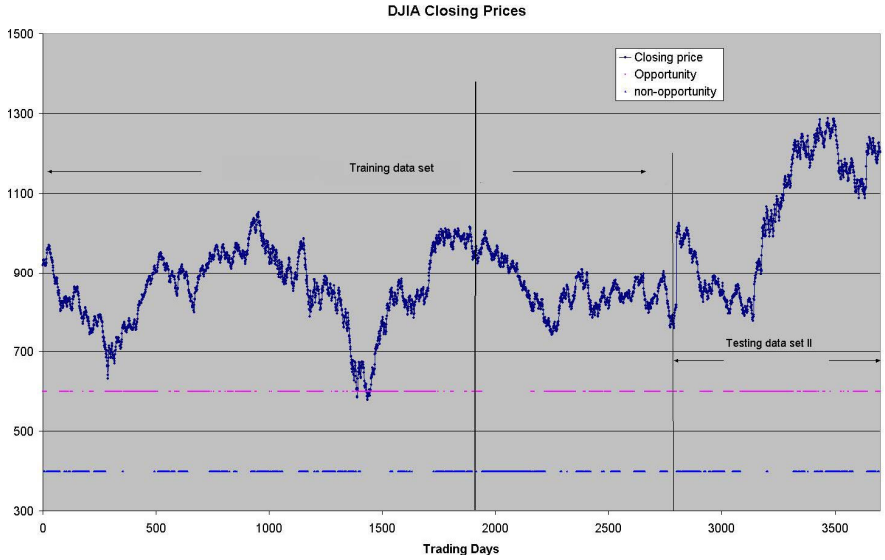

Figura 3.2 Distribución de los datos de prueba y entrenamiento.

3.1.1 Atributos

Los datos están divididos en seis atributos que son:

Cuadro 3.1: Distribución de los datos.

Nombre Atributo	Valores posibles archivo1	Valores posibles archivo2
PMV_12	entre -58.13 y 54.392	entre -58.213 y 82.527
PMV_50	entre -130.564 y 95.951	entre -130.564 y 128.842
TRB_5	entre -66.38 y 29.94	entre -66.38 y 43.41
TRB_50	entre -218.16 y 26.05	entre -218.16 y 36
Filter_5	entre -28.76 y 79.2	entre -28.76 y 109.18
Filter_63	entre -20.91 y 199.7	entre -22.72 y 288.57

Cada atributo representa el promedio del precio de las acciones DJIA que cierran en los 12 y 50 días anteriores, el precio máximo de los últimos 5 y 50 días y por ultimo el precio mínimo de los 5 y 63 días anteriores respectivamente como lo muestra el cuadro 3.1.

3.1.2 Clases

Los datos anteriores se encuentran divididos en dos clases que apoyarán a la decisión de comprar o no dichas clases son: Yes y No

3.1.3 Cantidad de instancias y distribución de clases

Para realizar los experimentos se contó con dos archivos como se muestra en el cuadro 3.2 el primero contendrá los datos de entrenamiento con los que se crearán los modelos que posteriormente serán probados con los del segundo archivo que contiene las instancias de pruebas cada uno de ellos consta de 2800 instancias, la distribución gráfica de los datos se muestra en la figura 3.3 en la que se observa la distribución en cada uno de los archivos mencionados.

Cuadro 3.2: Distribución de clases en cada archivo.

Clases	Conjunto Entrenamiento	Conjunto de Prueba	Total
Yes	1447	1471	2918
No	1353	1329	2682
Total	2800	2800	5600

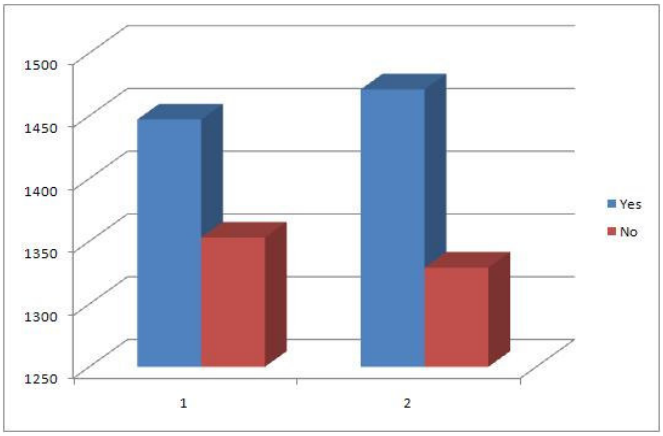

Figura 3.3: Distribución gráfica de los datos

3.2 Interpretación de los resultados

En esta sección mostraré la idea general de como se deben de interpretar cada una de los cuadros y diagramas, en especial el caso de los árboles creados con C4.5 cada hoja tiene asociados dos números N y E, como lo indica la figura 3.4 Cabe destacar que si E es nulo, entonces, no se expresa en el árbol. N es la suma de los casos fraccionarios que llegan a cada hoja y E es la suma de los casos que pertenecen a una clase distinta de la correspondiente a la hoja, los falsos positivos. Es decir, que de los N casos cubiertos por la hoja, E casos son incorrectos. En los árboles podados, N es la cantidad de casos de entrenamiento cubiertos por la hoja, y E es la cantidad de errores predichos si una cantidad N de casos nuevos fuese clasificada por el árbol, según la distribución binomial.

```
Prueba 1 del atributo X: clase A (N/E)
Prueba 2 del atributo X:
       Prueba 1 del atributo Y:
              Prueba 1 del atributo Z: clase A (N)
              Prueba 2 del atributo Z: clase B (N/E)
       Prueba 2 del atributo Y: clase A (N/E)
```

Figura 3.4: Esquema general de un árbol obtenido con C4.5

Para todas las pruebas se obtiene un cuadro de evaluación de idéntico formato: a partir de los datos de prueba. La cual indica en cada caso:

- **Errores** (porcentaje de error): los errores indican la cantidad de casos clasificados erróneamente; mientras que el porcentaje de error es dicha cantidad sobre la cantidad total de casos.
- **Estimación:** es un estimador del éxito del árbol obtenido según la ecuación 3.1.

$$\varepsilon = \frac{correctos}{correctos + errores} \tag{3.1}$$

Además, para cada uno de los métodos se presenta la matriz de confusión como la que se muestra en el cuadro 3.3. Donde se indica para cada clase, la cantidad de casos que fueron clasificados correctamente y la cantidad de casos que no fueron clasificados correctamente.

Cuadro 3.3: Matriz de confusión

Clases	Bien clasificados	Mal Clasificados	Probabilidad de efectividad
Yes			
No			

3.3 Pruebas C4.5

Escogí este algoritmo por ser el que representa el conocimiento de manera simple a través de un árbol donde las ramas simulan el proceso humano para la toma de decisiones.

Además de que nos proporcionan un alto grado de comprensión del conocimiento utilizado en la toma de decisiones a causa de que los modelos que se generan están basados en reglas de decisiones. Por otro lado tenemos que la técnica resume los ejemplos de partida, permitiendo la clasificación de nuevos casos siempre y cuando no existan modificaciones sustanciales en las condiciones bajo las cuales se generaron los ejemplos que sirvieron para su construcción.

Cabe mencionar que la técnica es una de las más utilizadas en el área por su eficacia al generar modelos para predecir y ajustarse a cualquier tipo de datos tanto continuos como discretos.

3.3.1 C4.5 utilizando validación cruzada a 10 FOLDS

Árbol de decisión

Las 24 reglas de decisión que forman al árbol se muestran en la figura 3.5 las cuales se pueden contar de manera sencilla ya que son equivalentes al número de hojas que tiene dicho árbol y que a su vez son la parte donde concluyen las decisiones, este modelo se construyó en 0.13 segundos. También se puede observar que en la regla: Si TRB_50<=-73.22 entonces Yes tienen 159 instancias mal clasificadas de un total de 691 que pertenecían a la clase lo que da como resultado que no es efectiva por que lo se esperaría lograr es que todas las reglas se clasificasen con un 100% de efectividad.

```
TRB_50 <= -73.22: yes (532.0/159.0)
TRB_50 > -73.22
|   Filter_63 <= 71.03
|   |   TRB_50 <= -52.75
|   |   |   PMV_50 <= -9.868: no (329.0/154.0)
|   |   |   PMV_50 > -9.868: yes (48.0/6.0)
|   |   TRB_50 > -52.75: no (1228.0/476.0)
|   Filter_63 > 71.03
|   |   TRB_50 <= -9.14
|   |   |   TRB_5 <= -7.77
|   |   |   |   PMV_50 <= 56.731
|   |   |   |   |   PMV_12 <= -2.557: yes (109.0/30.0)
|   |   |   |   |   PMV_12 > -2.557
|   |   |   |   |   |   PMV_12 <= 12.427
|   |   |   |   |   |   |   Filter_5 <= 7.12
|   |   |   |   |   |   |   |   TRB_5 <= -17.07: no (6.0)
|   |   |   |   |   |   |   |   TRB_5 > -17.07: yes (26.0/11.0)
|   |   |   |   |   |   |   Filter_5 > 7.12: no (17.0/1.0)
|   |   |   |   |   |   PMV_12 > 12.427: yes (7.0)
|   |   |   |   PMV_50 > 56.731: yes (15.0)
|   |   |   TRB_5 > -7.77
|   |   |   |   PMV_12 <= 16.321: yes (71.0/6.0)
|   |   |   |   PMV_12 > 16.321
|   |   |   |   |   Filter_63 <= 75.83: no (4.0)
|   |   |   |   |   Filter_63 > 75.83
|   |   |   |   |   |   PMV_12 <= 20
|   |   |   |   |   |   |   TRB_50 <= -25.41: no (2.0)
|   |   |   |   |   |   |   TRB_50 > -25.41: yes (2.0)
|   |   |   |   |   |   PMV_12 > 20: yes (5.0)
|   |   TRB_50 > -9.14
|   |   |   PMV_50 <= 73.151
|   |   |   |   Filter_5 <= 21.77
|   |   |   |   |   PMV_50 <= 25.281: yes (23.0/3.0)
|   |   |   |   |   PMV_50 > 25.281
|   |   |   |   |   |   Filter_63 <= 106: no (168.0/76.0)
|   |   |   |   |   |   Filter_63 > 106
|   |   |   |   |   |   |   PMV_12 <= 17.51: yes (60.0/14.0)
|   |   |   |   |   |   |   PMV_12 > 17.51
|   |   |   |   |   |   |   |   PMV_50 <= 46.774: no (6.0)
|   |   |   |   |   |   |   |   PMV_50 > 46.774
|   |   |   |   |   |   |   |   |   PMV_50 <= 58.699: yes (3.0)
|   |   |   |   |   |   |   |   |   PMV_50 > 58.699: no
(4.0/1.0)
|   |   |   |   Filter_5 > 21.77
|   |   |   |   |   Filter_5 <= 45.44: no (87.0/24.0)
|   |   |   |   |   Filter_5 > 45.44: yes (12.0/1.0)
|   |   |   PMV_50 > 73.151: yes (36.0/4.0)
```

Figura 3.5: Reglas que forman el árbol de decisión (C4.5 10 Folds)

Evaluación de los resultados del árbol de decisión

En el cuadro 3.4 se muestra la exactitud del modelo respecto al total de instancias evaluadas donde la estimación de los que fueron correctamente clasificados se calculó con la fórmula 3.1 mientras que el error fue obtenido de restar del 100% que sería la mejor clasificación del porcentaje de efectividad quedando entonces que la estimación es igual 60.39.29% con un error de 39.6071%.

Otra manera de visualizar los resultados es a través de la matriz de confusión del cuadro 3.5 donde se puede ver los resultados de forma más detallada ya que se calculan las estimaciones de efectividad para cada clase ocupando nuevamente la ecuación 3.1 es decir, para la clase Yes se tienen 740 correctos de un total de 1471 aplicando la ecuación se obtiene una efectividad del 50.31%.

Cuadro 3.4: Evaluación C4.5 sobre datos de prueba (10 folds)

	Numero de instancias	Porcentaje
Correctos	1691	60.3929%
Incorrectos	1109	39.6071%

Cuadro 3.5: Matriz de confusión C4.5 (10 folds)

Clases	Correctos	Errores	Estimación
Yes	740	731	0.5031
No	951	378	0.7156
Totales	1691	1109	0.603929

En este experimento se puede observar que el modelo construido clasifica de mejor forma la clase NO que supera en gran número a la clase Yes, es decir que cuando se quiera clasificar una nueva instancia hay mayor probabilidad de que la clase NO sea correctamente clasificada que uno de la clase Yes.
El modelo construido no cubre las expectativas deseadas ya que para poder ser eficiente las estimaciones de efectividad deben ser superiores al 90% garantizando que la evaluación de una nueva instancia será correcta para poder así ocuparlo para la toma de decisiones.

3.3.2 C4.5 utilizando Split Sample al 66%

Árbol de decisión

El árbol generado de este experimento se creo en 0.19 segundos. El cual es idéntico al de la prueba de la sección 3.3.1 como se puede ver en la figura 3.6.

Una de las reglas del árbol es la siguiente:

Si TRB_50 > -73.22
Y Filer_63 > 71.03
Y TRB_50 > -9.14
Y PMV_50 <= 73.151
Y Filter_5 > 21.77
Y Filter_5 > 45.44: clase Yes (12.0/1.0)

Lo interesante de esta regla es que no se toman en cuenta los atributos PMB_12 y TRB_5 lo que quiere decir que no importa el valor que contenga en una clasificación cada vez que se cumplan con los parámetros de los otros atributos. Por otro lado la regla maneja diferentes condiciones para un solo atributo como es el caso de Filter_5 esto se debe a la ruptura que hace el algoritmo con los atributos continuos para aumentar la ganancia aunque se repiten los atributos las condiciones que deben cubrir son consistentes.

```
TRB_50 <= -73.22: yes (532.0/159.0)
TRB_50 > -73.22
|   Filter_63 <= 71.03
|   |   TRB_50 <= -52.75
|   |   |   PMV_50 <= -9.868: no (329.0/154.0)
|   |   |   PMV_50 > -9.868: yes (48.0/6.0)
|   |   TRB_50 > -52.75: no (1228.0/476.0)
|   Filter_63 > 71.03
|   |   TRB_50 <= -9.14
|   |   |   TRB_5 <= -7.77
|   |   |   |   PMV_50 <= 56.731
|   |   |   |   |   PMV_12 <= -2.557: yes (109.0/30.0)
|   |   |   |   |   PMV_12 > -2.557
|   |   |   |   |   |   PMV_12 <= 12.427
|   |   |   |   |   |   |   Filter_5 <= 7.12
|   |   |   |   |   |   |   |   TRB_5 <= -17.07: no (6.0)
|   |   |   |   |   |   |   |   TRB_5 > -17.07: yes (26.0/11.0)
|   |   |   |   |   |   |   Filter_5 > 7.12: no (17.0/1.0)
|   |   |   |   |   |   PMV_12 > 12.427: yes (7.0)
|   |   |   |   PMV_50 > 56.731: yes (15.0)
|   |   |   TRB_5 > -7.77
|   |   |   |   PMV_12 <= 16.321: yes (71.0/6.0)
|   |   |   |   PMV_12 > 16.321
|   |   |   |   |   Filter_63 <= 75.83: no (4.0)
|   |   |   |   |   Filter_63 > 75.83
|   |   |   |   |   |   PMV_12 <= 20
|   |   |   |   |   |   |   TRB_50 <= -25.41: no (2.0)
|   |   |   |   |   |   |   TRB_50 > -25.41: yes (2.0)
|   |   |   |   |   |   PMV_12 > 20: yes (5.0)
|   |   TRB_50 > -9.14
|   |   |   PMV_50 <= 73.151
|   |   |   |   Filter_5 <= 21.77
|   |   |   |   |   PMV_50 <= 25.281: yes (23.0/3.0)
|   |   |   |   |   PMV_50 > 25.281
|   |   |   |   |   |   Filter_63 <= 106: no (168.0/76.0)
|   |   |   |   |   |   Filter_63 > 106
|   |   |   |   |   |   |   PMV_12 <= 17.51: yes (60.0/14.0)
|   |   |   |   |   |   |   PMV_12 > 17.51
|   |   |   |   |   |   |   |   PMV_50 <= 46.774: no (6.0)
|   |   |   |   |   |   |   |   PMV_50 > 46.774
|   |   |   |   |   |   |   |   |   PMV_50 <= 58.699: yes (3.0)
|   |   |   |   |   |   |   |   |   PMV_50 > 58.699: no
(4.0/1.0)
|   |   |   |   Filter_5 > 21.77
|   |   |   |   |   Filter_5 <= 45.44: no (87.0/24.0)
|   |   |   |   |   Filter_5 > 45.44: yes (12.0/1.0)
|   |   |   PMV_50 > 73.151: yes (36.0/4.0)
```

Figura 3.6: Reglas que forman el árbol de decisión (split 66%)

Evaluación de los resultados del árbol de decisión

Los resultados que se obtuvieron del experimento se muestran en los cuadros 3.6 y 3.7 cada una de las estimaciones se calculó de la misma forma que como se hizo en el experimento con validación cruzada a 10 folds sección 3.3.1.

Ahora bien el cuadro 3.6 muestra que el error de efectividad del modelo es del 39.6071% el cuál es demasiado alto ya que el ideal seria del 0% ó cuando mucho del 10%. El segundo cuadro por su parte nos señala que la clase mejor clasificada es No con un 71.56% lo cual se impuso sobre la de Yes con solo un 50.31% lo cual es una pobre estimación ya que al realizar las evaluaciones pertenecientes a esta clase tendrán un alto grado de incertidumbre.

Cuadro 3.6: Evaluación C4.5 sobre los datos de prueba (split 66%)

	Numero de instancias	Porcentaje
Correctos	1691	60.3929%
Incorrectos	1109	39.6071%

Cuadro 3.7: Matriz de confusión C4.5 (split 66%)

Clases	Correctos	Errores	Probabilidad
Yes	740	731	0.5031
No	951	378	0.7156
Totales	1691	1109	0.603929

Nuevamente el modelo como en el experimento anterior no cumple con las espectativas deseadas ya que la cantidad de errores en la clasificacion es demasiado alta en comparación con el total de las instancias, una cantidad posible sería que fueran menor de 280 instancias lo cual es solo el 10% de los datos.

3.3.3 C4.5 utilizando conjunto de entrenamiento (suplied test)

El árbol que se creo en este experimento es el mismo que el de las secciones 3.3.2 y 3.3.1 en el que se puede observar la siguiente regla:

Si TRB_50 > -73.22
Y Filter_63 > 71.03
Y TRB_50 > -9.14
PMV_50 <= 73.151
Filter_5 <= 21.77
PMV_50 > 25.281
Filter_63 > 106
PMV_12 > 17.51
PMV_50 > 46.774
PMV_50 > 58.699 : clase NO

Cuando se evalue una instancia que cumpla la regla anerior será clasificada como NO pero en la regla anterior se puede mostrar que no cuenta con los parámetros que nos dicen cuantas se clasificaron bien después de que se construyó el modelo esto se debe a que no había ninguna instancia que cumpliera los valores establecidos pero la regla debió de crearse ya que existe la posibilidad de que en alguna instancia las cumpla y como se dijo anteriormente los árboles de decisión deben cubrir todas las posibilidades.

Cuadro 3.8: Evaluación C4.5 sobre los datos de prueba (conjunto de entrenamiento)

	Numero de instancias	Porcentaje
Correctos	1691	60.3929%
Incorrectos	1109	39.6071%

Cuadro 3.9: Matriz de confusión C4.5 (conjunto de entrenamiento)

Clases	Correctos	Errores	Probabilidad
Yes	740	731	0.5031
No	951	378	0.7156
Totales	1691	1109	0.603929

Los resultados del experimento se muestran en los cuadros 3.8 y 3.9 donde cada una de las estimaciones fue calculada en base a la ecuación 3.1 de igual manera se realizó en las secciones 3.3.2 y 3.3.1.
El experimento resultó desfavorable ya que se requiere obtener un porcentaje de efectividad del modelo no menor de 90% y de igual forma en cada una de las clases por separado.

3.3.4 Comparación de Resultados obtenidos con C4.5

A continuación comentaré los aspectos más destacados de los estudios realizados y las conclusiones que de ellos puedo derivar, así como las diferencias que se obtienen entre las distintas formas de probar un algoritmo.

Cuadro 3.10: Resultados obtenidos de los experimentos con C4.5

	Instancias clasificadas correctamente (%)	Tiempo para construir el modelo (s)
Validación cruzada 10 folds	60.3929	0.13
División 66%	60.3929	0.19
Suplied test	60.3929	0.24

Los resultados obtenidos de los 3 experimentos con C4.5 se muestran en el cuadro 3.10 donde puede verse, que los tres modelos clasificaron la misma cantidad de instancias correctamente; pero si tuviera que elegir por uno de estos algoritmos tendría que tomar en cuenta otro parámetro de evaluación, es por eso que el cuadro contiene otra columna de interés en la construcción de modelos que es el tiempo que puede ser de vital importancia cuando los datos se tienen en un gran volumen ya que esto hará que el tiempo computacional se ve afectado de manera proporcional.
Por otra parte lo que muestran los cuadros de confusión de las secciones 3.3.1, 3.3.2 y 3.3.3 es que hay una mayor probabilidad de clasificar un elemento de clase No, ya que tiene una probabilidad de 71.56% mientras que la clase Yes solo cuenta con el 50.31% de efectividad lo cual nos deja con un alto grado de incertidumbre al querer probar un elemento de esta clase.
Para los tres experimentos se obtuvo una probabilidad de efectividad del modelo de 60.39% lo cual no cumple con las expectativas ya que se pretende lograr que sea mayor al 90%.

3.4 Pruebas Naive Bayes

La principal característica de este algoritmo es que la construccion de los modelos se hace de forma rápida igualando el rendimiento de otros algoritmos como pueden ser los árboles de predicción; en este método cada uno de los ejemplos puede aumentar o disminuir la estimación de una hipótesis, además de que sirve como estandar para la comparación de otros algoritmos.
Por otra parte este método se utiliza cuando queremos clasificar una instancia descrita por un conjunto de atributos sobre un grupo finito de clases que es nuestro caso.
Por lo que mencione anteriormenete respecto de los árboles de predicción los resultados que se obtuvieron no son favorables, pero los ejemplos servirán para comprobar dicha hipótesis.

3.4.1 Naive Bayes utilizando validación cruzada a 10 Folds

El tiempo que tomó la creación del modelo fue de 0.03 segundos en el cuadro 3.11 se muestran la media y la desviación tipica de cada uno de los atributos que son los datos que se ocupan para tomar una decisión sobre la clase a que pertenece una instancia a través de la ecuación 2.18, pero hay que recordar que estos valores se calculan para cada clase, en nuestro caso son 2.

Cuadro 3.11: Parámetros de distribución (10 folds)

	Media (Yes)	Desviación Típica (Yes)	Media (No)	Desviación Típica (No)
PMV_12	-0.6734	16.1537	0.2343	15.4048
PMV_50	-3.1634	37.455	1.0424	29.7532
TRB_5	-9.3531	13.6009	-8.3418	12.8357
TRB_50	-46.4967	42.1706	-35.1276	32.8494
Filter_5	8.7916	13.520	8.5093	12.8094
Filter_63	48.6495	43.2878	45.0047	36.4078

Evaluación de los resultados aplicando los parámtetros de distribución

En el cuadro 3.12 se muestra la exactitud con la que cuenta el modelo una vez que ha sido probado con los datos del segundo archivo de la misma manera de como se hizo con los árboles de decisión en la sección 3.3 usando la ecuación 3.1 para cada una de las estimaciones y restando estas del 100% para calcular el error. Se puede observar que el modelo cuenta con un 55.2143% de efectividad para conocer de forma más detallada el comportamiento del modelo, pasemos al cuadro 3.13 en el cual la matriz de confusión obtenida del modelo se puede observar que la clase No ha sido mejor clasificada con un 63.58% de todas las instancias pertenecientes a este grupo mientras, que la clasificación de Yes fue bastante desfavorable por no superar siquiera el 50%.

Cuadro 3.12: Evaluación Naive Bayes (10 folds)

	Numero de instancias	Porcentaje
Correctos	1546	55.2143%
Incorrectos	1254	48.7857%

Cuadro 3.13: Matriz de confusión Naive Bayes (10 folds)

Clases	Correctos	Errores	Probabilidad
Yes	701	770	0.4765
No	845	484	0.6358
Totales	1546	1254	0.552143

Los resultados que se obtuvieron de esta prueba fueron bastante desfavorables ya que se obtuvo un porcentaje de apenas un 55.2143% de efectividad, aunque esperaba que así fueran ya que como anteriormente manifesté podían alcanzar el rendimiento de los árboles de decisión y en las pruebas no se superaron los requisitos para poderse considerar como un buen modelo.

3.4.2 Naive Bayes utilizando Split Sample 66%

Lo primero que caracterizó a este experimento fue la velocidad con la que se construyó el modelo corroborando que una de las principales características de Naive Bayes es la velocidad ya que solo tardó 0.02 segundos.
Otra de las características del modelo es que se obtuvieron los mismos valores de distribución que en la sección 3.4.1 lo cual nos indicaba que se tendría la misma efectividad del modelo como se puede ver en el cuadro 3.15 y la matriz de confusión del cuadro 3.16

Cuadro 3.14: Parámetros de distribución (10 folds)

	Media (Yes)	Desviación Típica (Yes)	Media (No)	Desviación Típica (No)
PMV_12	-0.6734	16.1537	0.2343	15.4048
PMV_50	-3.1634	37.455	1.0424	29.7532
TRB_5	-9.3531	13.6009	-8.3418	12.8357
TRB_50	-46.4967	42.1706	-35.1276	32.8494
Filter_5	8.7916	13.520	8.5093	12.8094
Filter_63	48.6495	43.2878	45.0047	36.4078

Evaluación de los resultados de las proporciones de clases

Cuadro 3.15: Evaluación Naive Bayes sobre los datos de prueba (split 66%)

	Numero de instancias	Porcentaje error
Correctos	1546	55.2143%
Incorrectos	1254	48.7857%

Cuadro 3.16: Matriz de confusión Naive Bayes (split 66%)

Clases	Correctos	Errores	Probabilidad
Yes	701	770	0.4765
No	845	484	0.6358
Totales	1546	1254	0.552143

Como era de esperarse los resultados de este experimento no mejoraron toda vez que la clasificación de las clases solo logro obtener un 55.2143% de efectividad y lo que se espera encontrar es que fuera mayor al 90%.

3.4.3 Naive Bayes utilizando conjunto de entrenamiento

Nuevamente en la construcción de este modelo se logra ver que el algoritmo es bastante rápido siendo construido en 0.02 segundos; aunque esto no fue nada alentador al observar que de igual forma los parámetros de distribución eran los mismos (cuadro 3.17) trayendo consigo los mismos resultados en cuanto a efectividad en la clasificación como se puede observar en los cuadros 3.18 y 3.19.

Cuadro 3.17: Parámetros de distribución (10 folds)

	Media (Yes)	Desviación Típica (Yes)	Media (No)	Desviación Típica (No)
PMV_12	-0.6734	16.1537	0.2343	15.4048
PMV_50	-3.1634	37.455	1.0424	29.7532
TRB_5	-9.3531	13.6009	-8.3418	12.8357
TRB_50	-46.4967	42.1706	-35.1276	32.8494
Filter_5	8.7916	13.520	8.5093	12.8094
Filter_63	48.6495	43.2878	45.0047	36.4078

Evaluación de los resultados de las proporciones de clases

Los resultados que aparecen en los cuadros 3.18 y 3.19 son los mismos que en las secciones 3.4.2 y 3.4.1 esto se debe a que los parametros de distribucion son los mismos.

Cuadro 3.18: Evaluación Naive Bayes (conjunto de entrenamiento)

	Numero de instancias	Porcentaje error
Correctos	1546	55.2143%
Incorrectos	1254	48.7857%

Cuadro 3.19: Matriz de confusión Naive Bayes (conjunto de entrenamiento)

Clases	Correctos	Errores	Probabilidad
Yes	701	770	0.4765
No	845	484	0.6358
Totales	1546	1254	0.552143

3.4.4 Comparación de Resultados Naive Bayes

En esta sección comentaré las características más relevantes al construir los modelos con Naive Bayes.
En primer lugar como se puede observar en el cuadro 3.20 el algoritmo es bastante rápido para la construcción de los modelos con el inconveniente de que los resultados que se obtienen de el son ineficientes para este caso.
Otra de las características de los experimentos es que no importo el tipo de validación que se ocupó siempre que se obtuvieran los mismos resultados, esto se debe a que se toman en cuenta todos los valores para que el modelo pueda calcularse de una forma más correcta los parámetros de distribucion que son los que nos indicarán a que clase pertenece cada una de las instancias.

Cuadro 3.20: Resultados obtenidos de los experimentos con C4.5

	Instancias clasificadas correctamente (%)	Tiempo para construir el modelo (s)
Validación cruzada 10 folds	55.2143	0.03
División 66%	55.2143	0.02
Suplied test	55.2143	0.02

Toda vez que los tres experimentos clasificaron con el mismo número de instancias, si se quisiera obtener la mejor forma de de evaluación se tendría que descartar al de validación cruzada a 10 folds ya que fue el que más tardo en comparación con las otras dos pruebas.

3.5 Pruebas con Bagging

La idea de ocupar multiclasificadores surgió al observar que los resultados obtenidos en las secciones 3.3 y 3.4 no eran los esperados ya que no superaron el 90% de efectividad y suponiendo que el problema era la forma en que se encontraban los datos por lo que aprovechando que el método bagging genera distintos conjuntos de entrenamiento del original, además estos métodos mejoran la precisión de las predicciones ya que se construyen varios modelos obteniendo la clasificación de mayoria de votos.
Para estos experimentos ocupé como algoritmo base al C4.5 que fue el que mejor resultados me proporcionó en los experimentos anteriores con 20 iteraciones, es decir se generarán 20 modelos diferentes.

3.5.1 Bagging utilizando validación cruzada a 10 Folds

El modelo construido con este método se generó en 2.64 segundos.

Evaluación de los resultados

En el cuadro 3.21 se muestra la exactitud del modelo respecto al total de instancias evaluadas con un porcentaje de 67.6429% y un errror del 32.3571% estos resultados fueron obtenidos de aplicar la formula 3.1 de igual forma que como se ha hecho con los demás experimentos. Para tener una mejor idea del comportamiento del modelo en la matriz de confusión (cuadro 3.22) se presenta la cantidad de instancias que fueron clasificadas correctamente así como los errores de dicha clasificación para cada una de las clases y su probabilidad de clasificar correctamente una nueva instancia.

Cuadro 3.21: Evaluación bagging sobre los datos de prueba (10folds)

	Numero de instancias	Porcentaje
Correctos	1894	67.6429%
Incorrectos	906	32.3571%

Cuadro 3.22: Matriz de confusión Bagging (10folds)

Clases	Correctos	Errores	Probabilidad
Yes	1000	471	0.6798
No	894	435	0.6726
Totales	1894	906	0.676429

En los resultados de este experimento se visualiza que las clases se clasificaron en la misma cantidad con un 67% pero esto no fue suficiente para decir que el modelo es bueno ya que como he mencionado anteriormente se requiere de un 90% o más de efectividad.

3.5.2 Bagging utilizando Split Sample 66%

Este experimento nos proporcionó 20 árboles diferentes que son los que se ocuparán para la evaluación de las nuevas instancias contenidas en archivo 2 el tiempo en que se construyó fue de 2.59 segundos.

Evaluación de los resultados

Los resultados que se obtuvieron del experimento se muestran en los cuadros 3.23 y 3.24 cada de las estimaciones fue calculada como se ha venido haciendo en los experimentos anteriores ocupando la ecuación 3.1.

El cuadro 3.3 nos muestra que el modelo cuenta con un error demasiado grande del 32.3571% tomando en cuenta que para ser eficiente este debe de ser no

mayor al 10%. Por otra parte el cuadro 3.24 es la matriz de confusión en el que se observa que las dos clases fueron igualmente clasificadas con tan solo una diferencia de 0.0072% lo cual es mínimo en comparación del total de los datos aunque esta clasificación no haya podido ser la deseada al contar con tan solo un 67% de efectividad.

Cuadro 3.23: Evaluación Bagging sobre los datos de prueba (split 66%)

	Numero de instancias	Porcentaje error
Correctos	1894	67.6429%
Incorrectos	906	32.3571%

Cuadro 3.24: Matriz de confusión Bagging (split 66%)

Clases	Correctos	Errores	Probabilidad
Yes	1000	471	0.6798
No	894	435	0.6726
Totales	1894	906	0.676429

Los resultados obtenidos de este experimento no cumplieron con los requisitos para poder considerar el modelo como bueno ya que solo obtuvo un 67.6429% de efectividad cuando el que se busca debe ser 90% o más.

3.5.3 Bagging utilizando conjunto de entrenamiento

La construcción de este modelo duró 2.58 segundos los resultados se pueden ver en los cuadros 3.25 y 3.26 que son los mismos que en las secciones 3.5.2 y 3.5.1 ya que los árboles con los que se formó el modelo son los mismos. Por lo que se puede concluir que nuevamente en esta prueba no se pudo encontrar el resultado deseado de una clasificación efectiva superior del 90%.

Cuadro 3.25: Evaluación Bagging sobre los datos de prueba (conjunto de entrenamiento)

	Numero de instancias	Porcentaje error
Correctos	1894	67.6429%
Incorrectos	906	32.3571%

Cuadro 3.26: Matriz de confusión Bagging (conjunto de entrenamiento)

Clases	Correctos	Errores	Probabilidad
Yes	1000	471	0.6798
No	894	435	0.6726
Totales	2146	906	0.676429

3.5.4 Comparación de Resultados Bagging

En este apartado comentaré los aspectos más relevantes de los experimentos realizados con el algoritmo Bagging destacando las diferencias de cada uno para poder seleccionar aquel que mejor se adaptó a los resultados.

Los resultados de cada experimento se presentan en el cuadro 3.27 en el que se muestra la estimación de instancias correctamente clasificadas y el tiempo que se tardo en construir cada modelo en base al tipo de validación que se ocupó.

Cuadro 3.27: Resultados obtenidos de los experimentos con Bagging

	Instancias clasificadas correctamente (%)	Tiempo para construir el modelo (s)
Validación cruzada 10 folds	67.6429	2.64
División 66%	67.6429	2.59
Suplied test	67.6429	2.58

Lo primero que cabe resaltar de estos algoritmos es que las tres pruebas produjeron el mismo porcentaje de efectividad que no fue el buscado la diferencia de estos está en el tiempo que tardaron en la construcción del modelo aunque la diferencia no fue muy grande dado la cantidad de datos pero si esta aumentara considerablemente el tiempo igual lo haría por lo que se puede tomar la decisión del que mejor se adapto fue el de suplied test con solo 2.58 segundos.

Por último y basado en que ningún experimento pudo superar el 90% de efectividad se puede decir que este algoritmo no es bueno para la realización de predicciones.

3.6 Pruebas con RandomCommittee

Al observar que cuando se ocupó el método bagging para la construcción de los modelos hubo una considerable mejora en los resultados en comparación a los de la sección 3.4 y 3.3 se decidió ocupar otro multiclasificador que construyera pero con una diferente forma de manejar los datos de ahí que se decidió ocupar este algoritmo.Dado que este algoritmo funciona con algoritmos que manejen aleatoreidad se ocupó el que la herramienta weka maneja por default que es el Randomtree el cuál crea un árbol considerando un número aleatorio de caracterizticas para cada nodo.

3.6.1 RandomCommittee utilizando validación cruzada a 10 Folds

Evaluación de los resultados con el árbol generado

El modelo se construyó en 2.66 segundos. Los resultados del experimento se muestran en el cuadro 3.28 en la que se puede ver que se tuvo un buen rendimiento en cuanto a la clasificación ya que el modelo construido tuvo una efectividad del 85.9643%, clasificando de mejor manera la clase Yes como se puede ver en la matriz de confusión del cuadro 3.29 con una efectividad del 90.55% que cumplió con lo que se estaba buscando a diferencia de la clase No que solo logró alcanzar un 80.89%.

Cada una de las estimaciones de efectividad se realizó en base a la ecuación 3.1 en base al total de instancias o las que pertenecen a una clase según requerí.

Cuadro 3.28: Evaluación RandomCommittee (10 folds)

	Numero de instancias	Porcentaje error
Correctos	2407	85.9643%
Incorrectos	393	14.0357%

Cuadro 3.29: Matriz de confusión RandomCommittee (10 folds)

Clases	Correctos	Errores	Probabilidad
Yes	1332	139	0.9055
No	1075	254	0.8089
Totales	2407	393	0.859643

Los resultados de este experimento fueron consoladores ya que se pudo obtener la efectividad que se requería aunque solo para una de las clases; la estimación de efectividad del experimento fue de 85.9643% nada despreciable pero no cumpliendo aún con lo que se espera llegar a obtener que es el una superior al 90%

3.6.2 RandomCommittee utilizando split sample 66%

Evaluación de los resultados con el árbol generado

El modelo resultante en este experimento tardó en construirse 1.33 segundos. En el cuadro 3.30 se puede observar la estimación de instancias correctamente clasificadas por el modelo creado con un total de 2407 de las 2800 que se utilizaron para probarlo y que aplicando la ecuación 3.1 nos da un porcentaje de 85.9643%.

Otra forma de ver los resultados es con la matriz de confusión del cuadro 3.31 donde se puede observar que la clase No se clasificó en un 80.89% cubriendo un total de 1075 instancias correctamente clasificadas que fue superado por la clase Yes que logró obtener un 90.55% del total de las instancias pertenecientes a esta clase.

Cuadro 3.30: Evaluación RandomCommittee sobre los datos de prueba (split 66%)

	Numero de instancias	Porcentaje error
Correctos	2407	85.9643%
Incorrectos	393	14.0357%

Cuadro 3.31: Matriz de confusión RandomCommittee (split 66%)

Clases	Correctos	Errores	Probabilidad
Yes	1332	139	0.9055
No	1075	254	0.8089
Totales	2407	393	0.859646

3.6.3 RandomCommittee utilizando conjunto de entrenamiento

Evaluación de los resultados de las proporciones de clases

Los resultados que se obtuvieron una vez que se probó el modelo con los archivos de prueba encontrados en el archivo 2 se muestran en el cuadro 3.32 en el que se observa que el error del modelo es del 14.0357% con 393 instancias mal clasificadas.

El cuadro 3.33 es la matriz de confusión donde se aprecia que la clase Yes se logro clasificar de mejor forma que la clase No con un 90.55% y 80.89% respectivamente.

Cuadro 3.32: Evaluación RandomCommittee (conjunto de entrenamiento)

	Numero de instancias	Porcentaje error
Correctos	2407	85.9643%
Incorrectos	393	14.0357%

Cuadro 3.33: Matriz de confusión RandomCommittee (conjunto de entrenamiento)

Clases	Correctos	Errores	Probabilidad
Yes	1332	139	0.9055
No	1075	254	0.8089
Totales	2407	393	0.859643

El modelo no cumplió con los requerimientos necesarios para poder catalogarlo como bueno ya que se desea que el error sea menor del 10% y fue de 14.0357%, algo relevante de este modelo es que si se deseará clasificar una nueva instancia perteneciente a la clase Yes tendríamos un alto grado de seguridad de tomar una buena decisión.

3.6.4 Comparación de Resultados RandomCommittee

En el cuadro 3.34 se muestran los resultados obtenidos de este experimento donde se puede observar que los tres algoritmos tuvieron la misma cantidad de instancias evaluadas correctamente por eso se toma en cuenta el tiempo para poder decir que el mejor fue el obtenido con suplied test con 1.16 segundos el que se puede descartar dado su tiempo de construcción es el de 10 folds.

Cuadro 3.34: Resultados obtenidos de los experimentos con RandomCommittee

	Instancias clasificadas correctamente (%)	Tiempo para construir el modelo (s)
Validación cruzada 10 folds	85.9643	2.66
División 66%	85.9643	1.33
Suplied test	85.9643	1.16

La probabilidad de efectividad de este modelo es del 85.9643% el cual es un muy buen resultado por su cercanía con el que se quiere llegar a tener faltándole menos del 5%.

3.7 Selección de atributos

Al observar el comportamiento de los datos con los experimentos realizados en las secciones 3.3 y 3.4 en comparación de los de las secciones 3.5 y 3.6 se pudo observar que los segundos arrojaron mejores resultados dado que la selección de los datos que ocupan los algoritmos Bagging y RandomCommittee se seleccionan aleatoriamente generando nuevos conjuntos a partir del principal, de ahí que pensé que podría haber atributos que aportaban más a los modelos que otros y apoyados en la herramienta weka que nos ofrece un apartado para distinguir aquellos que son mas relevantes procedí hacer una selección.

Para este caso se ocuparon el evaluador CfsSubsetEval y el método de búsqueda RandomSearch obteniendo que los atributos más importantes eran: PMV_50, TRB_50 y Filter_63 que son el promedio del precio de cierre de las acciones en los 50 días anteriores, el promedio máximo de los últimos 50 días y el precio mínimo de los 63 días anteriores respectivamente.

Para los experimentos que se presentarán en las secciones 3.8, 3.9, 3.10 y 3.11 se ocuparon los mismos algoritmos bajo los mismos términos como se hizo en los apartados 3.3, 3.4, 3.5 y 3.6.

3.8 Pruebas C4.5 con selección de atributos

3.8.1 C4.5 utilizando validación cruzada a 10 Folds

El modelo se construyó en un tiempo de 0.14 segundos el cual está formado por 8 reglas que se muestran el la figura 3.7 donde se puede ver que si se quiere clasificar una instancia en el que su atributo TRB_50 es menor o igual a -73.22 será clasificada como Yes.

Por otro lado es posible comprender mejor estas reglas cuando tenemos el árbol (figura 3.8) en la que se puede ver de forma más clara las 8 reglas ya que cada una concluye en las hojas para poder evaluar este modelo no hay más que en base a los atributos de una nueva instancia seguir el camino según las cantidades que tengamos.

```
TRB_50 <= -73.22: yes (532.0/159.0)
TRB_50 > -73.22
|   Filter_63 <= 71.03
|   |   TRB_50 <= -52.75
|   |   |   PMV_50 <= -9.868: no (329.0/154.0)
|   |   |   PMV_50 > -9.868: yes (48.0/6.0)
|   |   TRB_50 > -52.75: no (1228.0/476.0)
|   Filter_63 > 71.03
|   |   TRB_50 <= -9.14: yes (264.0/75.0)
|   |   TRB_50 > -9.14
|   |   |   PMV_50 <= 73.151
|   |   |   |   PMV_50 <= 26.496: yes (29.0/6.0)
|   |   |   |   PMV_50 > 26.496: no (334.0/158.0)
|   |   |   PMV_50 > 73.151: yes (36.0/4.0)
```

Figura 3.7: Reglas que forman el árbol de decisión (C4.5 10 Folds)

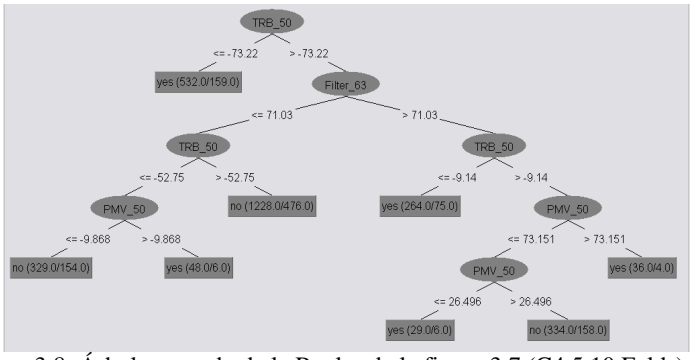

Figura 3.8: Árbol generado de la Reglas de la figura 3.7 (C4.5 10 Folds)

Evaluación de los resultados del árbol de decisión

Los resultados obtenidos al evaluar las 2800 instancias de prueba se muestran en forma general en el cuadro 3.35 en la que se observa que el modelo tiene una estimación de efectividad del 57.8571% el cual se calculó en base a la ecuación 3.1 y un error del 42.1429% calculado del restar del 100% de efectividad ideal requerido la estimación de efectividad.

De forma más detallada se presentan los resultado en la matriz de confusión del cuadro 3.36 en la que se puede ver que clase No pudo clasificar en un 70.43% de 1329 instancias pertenecientes superando en gran número a la clase Yes con solo un 46.49% clasificando 684 instancias de 1471 que había.

Cuadro 3.35: Evaluación C4.5 con selección de atributos sobre los datos de prueba (10 folds)

	Numero de instancias	Porcentaje
Correctos	1620	57.8571%
Incorrectos	1180	42.1429%

Cuadro 3.36: Matriz de confusión C4.5 con selección de atributos (10 folds)

Clases	Correctos	Errores	Probabilidad
Yes	684	787	0.4649
No	936	393	0.7043
Totales	1620	1180	0.578571

Los resultados presentados en este experimento no superaron las espectativas previstas ya que el error de clasificación fue del 42.1429% que es demasiado alto en comparación al deseado que debe ser menor del 10% además de que no nos proporciona ninguna garantía cuando se trata de un nuevo elemento perteneciente a la yes.

3.8.2 C4.5 utilizando Split Sample al 66%

En este experimento el modelo se construyó en 0.06 segundos del cuál se obtuvieron 8 reglas que se muestran en la figura 3.10 donde se puede observar que el árbol tendrá 14 ramas donde cada una será una de las comparaciones que sigue cada una de las reglas. Una forma de comprender esto es viendo el árbol de forma gráfica como se presenta en la figura 3.9 en la que los nodos representan el atributo que será evaluado las ramas serán las condiciones que se deben cumplir para seguir ese camino y por ultimo las hojas nos darán la solución sobre a que clase pertenece una instancia una vez evaluada.

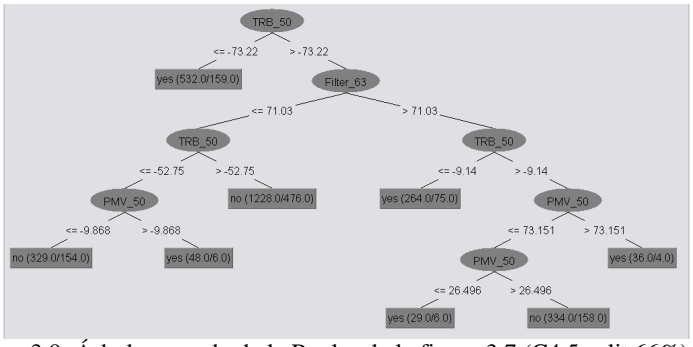

Figura 3.9: Árbol generado de la Reglas de la figura 3.7 (C4.5 split 66%)

```
TRB_50 <= -73.22: yes (532.0/159.0)
TRB_50 > -73.22
|   Filter_63 <= 71.03
|   |   TRB_50 <= -52.75
|   |   |   PMV_50 <= -9.868: no (329.0/154.0)
|   |   |   PMV_50 > -9.868: yes (48.0/6.0)
|   |   TRB_50 > -52.75: no (1228.0/476.0)
|   Filter_63 > 71.03
|   |   TRB_50 <= -9.14: yes (264.0/75.0)
|   |   TRB_50 > -9.14
|   |   |   PMV_50 <= 73.151
|   |   |   |   PMV_50 <= 26.496: yes (29.0/6.0)
|   |   |   |   PMV_50 > 26.496: no (334.0/158.0)
|   |   |   PMV_50 > 73.151: yes (36.0/4.0)
```

Figura 3.10: Reglas que forman el árbol de decisión (C4.5 split 66%)

Evaluación de los resultados del árbol de decisión

En los cuadros 3.37 y 3.38 se muestran los resultados de cómo se clasificaron en general los datos y como se comportaron respecto a cada una de las clases, en el primer cuadro se tiene que el modelo clasificó de forma correcta 1620 de los 2800 datos que se probaron es decir solo un 57.8571%, mientras que en el segundo cuadro se presenta la matriz de confusión que nos dice que la clase que peor se clasificó es la de Yes con un 46.49% que fue superado por la clase No con 936 instancias de las 1329 que hay en total cada uno de estos resultados se obtuvo aplicando la ecuación 3.1.

Cuadro 3.37: Evaluación C4.5 con selección de atributos sobre los datos de prueba (split 66%)

	Numero de instancias	Porcentaje error
Correctos	1620	57.8571%
Incorrectos	1180	42.1429%

Cuadro 3.38: Matriz de confusión C4.5 con selección de atributos (split 66%)

Clases	Correctos	Errores	Probabilidad
Yes	684	787	0.4649
No	936	393	0.7043
Totales	1620	1180	0.578571

Este modelo no pudo superar los requisitos de tener un error menor al 10% en cada una de las clases por lo que no se puede considerar el modelo como bueno.

3.8.3 C4.5 utilizando conjunto de entrenamiento

El árbol que se obtuvo de aplicar el algoritmo se muestra en la figura 3.11 el cual tardó 0.16 segundos en crearse y donde se puede ver que cuenta con 8 hojas o posibles formas de clasificar una instancia tomando como referencia cada una de las ramas otra forma de apreciarlo es a través de las reglas de la figura 3.12 que es otra representación del árbol no gráfica.

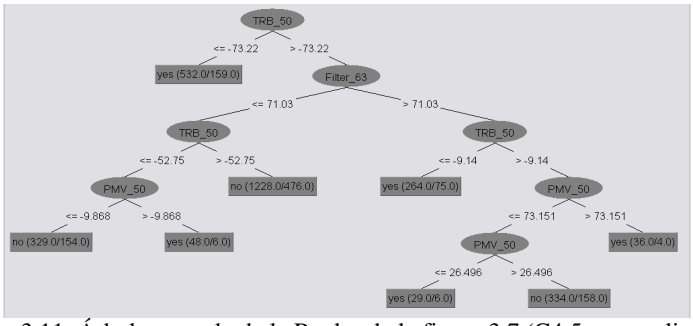

Figura 3.11: Árbol generado de la Reglas de la figura 3.7 (C4.5 con suplied test)

```
TRB_50 <= -73.22: yes (532.0/159.0)
TRB_50 > -73.22
|   Filter_63 <= 71.03
|   |   TRB_50 <= -52.75
|   |   |   PMV_50 <= -9.868: no (329.0/154.0)
|   |   |   PMV_50 > -9.868: yes (48.0/6.0)
|   |   TRB_50 > -52.75: no (1228.0/476.0)
|   Filter_63 > 71.03
|   |   TRB_50 <= -9.14: yes (264.0/75.0)
|   |   TRB_50 > -9.14
|   |   |   PMV_50 <= 73.151
|   |   |   |   PMV_50 <= 26.496: yes (29.0/6.0)
|   |   |   |   PMV_50 > 26.496: no (334.0/158.0)
|   |   |   PMV_50 > 73.151: yes (36.0/4.0)
```

Figura 3.10: Reglas que forman el árbol de decisión (C4.5 con suplied test)

Los resultados proyectados del modelo al probarse con los datos de prueba se muestran en el cuadro 3.39 que muestra que el porcentaje de efectividad es de 57.8571% que esta por de bajo del que se requiere. Por otra parte en la matriz de confusión presentada en el cuadro 3.40 se visualiza que la clase que mejor se clasificó fue la de No con un 70.43% mientras que la clase Yes obtuvo un 46.49% es decir no clasificó bien ni siquiera la mitad de los datos que se ocuparon para probarlo, de ahí que se pueda decir que el algoritmo bajo estos términos no se adaptó de forma adecuada.

Cuadro 3.39: Evaluación C4.5 con selección de atributos (conjunto de entrenamiento)

	Numero de instancias	Porcentaje error
Correctos	1620	57.8571%
Incorrectos	1180	42.1429%

Cuadro 3.40: Matriz de confusión C4.5 con selección de atributos (conjunto de entrenamiento)

Clases	Correctos	Errores	Probabilidad
Yes	684	787	0.4649
No	936	393	0.7043
Totales	1620	1180	0.578571

3.8.4 Comparación de Resultados obtenidos con C4.5

En el siguiente apartado dare una explicación de los resultados obtenidos en los experimentos de las secciones 3.8.3, 3.8.2 y 3.8.1.

En el cuadro 3.41 se presenta una comparativa de los modelos lo primero que se observa es que los tres modelos clasificaron correctamente el mismo número de instancias, entonces para poder decidir cual de ellos se ajusto de mejor forma a los datos se ocupará el parámetro del tiempo de construcción que es de vital importancia ya que si se llegra aumentar considerablemente el numero de instancias para la construcción de dicho modelo el tiempo lo haría proporcionalmente, de ahí que se puede decir que quien lo hizo fue el de división del 66% ya que lo logró hacer en 0.06 segundos.

Cuadro 3.41: Resultados obtenidos de los experimentos con C4.5

	Instancias clasificadas correctamente (%)	Tiempo para construir el modelo (s)
Validación cruzada 10 folds	57.8571	0.14
División 66%	57.8571	0.06
Suplied test	57.8571	0.16

Dado que los tres modelos creados obtuvieron el mismo porcentaje de efectividad y este fue de 57.8571% no superando los requisitos para poder tener un alto grado de confianza en cuanto a la clasificación de nuevas instancias se tiene que decir que el algoritmo no se adaptó con los datos.

3.9 Pruebas Naive Bayes con selección de atributos

3.9.1 Naive Bayes utilizando validación cruzada a 10 Folds

El tiempo que tomó la creación del modelo fue de 0.03 segundos en el cuadro 3.42 se muestran la media y la desviación tipica de cada uno de los atributos que son los datos que se ocupan para tomar una decisión sobre la clase a que pertenece una instancia a través de la ecuación 2.18, pero hay que recordar que estos valores se calculan para cada clase, en nuestro caso son 2.

Cuadro 3.42: Parámetros de distribución Naive Bayes (10 folds)

	Media (Yes)	Desviación Típica (Yes)	Media (No)	Desviación Típica (No)
PMV_50	-3.1634	37.455	1.0424	29.7532
TRB_50	-46.4967	42.1706	-35.1276	32.8494
Filter_63	48.6495	43.2878	45.0047	36.4078

Evaluación de los resultados de las proporciones de clases

En el cuadro 3.43 se muestra la exactitud con la que cuenta el modelo una vez que ha sido probado con los datos del segundo archivo de la misma manera de como se ha realizado en los apartados anteriores usando la ecuación 3.1 para cada una de las estimaciones y restando estas del 100% para calcular el error. Se puede observar que el modelo cuenta con un 55.6071% de efectividad para conocer de forma más detallada el comportamiento del modelo, pasemos al cuadro 3.44 en el cual la matriz de confusión obtenida del modelo se puede observar que la clase No ha sido mejor clasificada con un 65.09% de todas las instancias pertenecientes a este grupo, mientras que la clasificación de Yes fue bastante desfavorable por no superar siquiera el 50%.

Cuadro 3.43: Evaluación Naive Bayes con selección de atributos (10 folds)

	Numero de instancias	Porcentaje error
Correctos	1557	55.6071%
Incorrectos	1243	44.3929%

Cuadro 3.44: Matriz de confusión Naive Bayes con selección de atributos (10 folds)

Clases	Correctos	Errores	Probabilidad
Yes	692	779	0.4704
No	865	464	0.6509
Totales	1557	1243	0.556071

Los resultados que se obtuvieron de esta prueba no fueron favorables ya que se obtuvo un porcentaje de apenas un 55.6071% de efectividad, aunque este resultado esperaba que así fuera ya que este algoritmo alcanza el rendimiento de los árboles de decisión y en las pruebas de este tipo no se superaron los requisitos para poderse considerar como un buen modelo.

3.9.2 Naive Bayes utilizando split sample al 66%

El modelo resultante de este experimento se creo en solo 0.02 reafirmando que este algoritmo es caracterizado por la velocidad aunque desafortunadamente los resultados que más me interesan no fueron los que se requerían, primero mostraré en el cuadro 3.45 los parámetros de distribución con los que se calcula si una nueva instancia pertenece a una clase u otra.

Cuadro 3.45: Parámetros de distribución Naive Bayes (10 folds)

	Media (Yes)	Desviación Típica (Yes)	Media (No)	Desviación Típica (No)
PMV_50	-3.1634	37.455	1.0424	29.7532
TRB_50	-46.4967	42.1706	-35.1276	32.8494
Filter_63	48.6495	43.2878	45.0047	36.4078

Ahora bien en los cuadros 3.46 y 3.47 se presenta como clasificó el modelo con los datos de el cuadro 3.45, obteniendo un 55.671% de efectividad y un error del 44.3929% en cuanto a cada una de las clases la que mejor se clasificó fue la de No con 865 instancias que hacen un 65.09% del total que había en este grupo en comparación de la clase Yes con un 47.04%.

Cuadro 3.46: Evaluación Naive Bayes con selección de atributos (split 66%)

	Numero de instancias	Porcentaje
Correctos	1557	55.6071%
Incorrectos	1243	44.3929%

Cuadro 3.47: Matriz de confusión Naive Bayes con selección de atributos (split 66%)

Clases	Correctos	Errores	Probabilidad
Yes	692	779	0.4704
No	865	464	0.6509
Totales	1557	1243	0.556071

3.9.3 Naive Bayes utilizando conjunto de entrenamiento

El tiempo estimado de construcción del modelo fue de 0.03 segundos con los parámetros de distribución del cuadro 3.48 que nos ayudarán a la clasificación de las nuevas instancias ocupándolos en la ecuación 2.18.

Cuadro 3.48: Parámetros de distribución Naive Bayes (10 folds)

	Media (Yes)	Desviación Típica (Yes)	Media (No)	Desviación Típica (No)
PMV_50	-3.1634	37.455	1.0424	29.7532
TRB_50	-46.4967	42.1706	-35.1276	32.8494
Filter_63	48.6495	43.2878	45.0047	36.4078

Evaluación de los resultados de las proporciones de clases

Los resultados que se muestran en los cuadros 3.49 y 3.50 fueron calculados con la ecuación 3.1, donde se puede observar que el rendimiento de este algoritmo es del 55.6071% con un total de 1557 instancias clasificadas de las 2800 existentes. El otro cuadro es la matriz de confusión donde se realizan los mismos cálculos que con el cuadro anterior pero para cada clase.

Cuadro 3.49: Evaluación Naive Bayes con selección de atributos (conjunto de entrenamiento)

	Numero de instancias	Porcentaje
Correctos	1557	55.6071%
Incorrectos	1243	44.3929%

Cuadro 3.50: Matriz de confusión Naive Bayes con selección de atributos (conjunto de entrenamiento)

Clases	Correctos	Errores	Probabilidad
Yes	692	779	0.4704
No	865	464	0.6509
Totales	1557	1243	0.556071

Los resultados obtenidos de este experimento nos dan a entender que no se puede tener garantía al clasificar nuevas instancias con el modelo generado ya que para clasificar la clase Yes y No solo se tiene una efectividad de 47.04% y 65.09% cuando deberían de ser mayor de 90%.

3.9.4 Comparación de Resultados Naive Bayes

Una de las características más relevantes de los experimentos realizados en las secciones 3.9.3, 3.9.2 y 3.9.1 es el tiempo que tardaron en la creación de los modelos demostrando que en cuestión de velocidad son los mejores. En el cuadro 3.51 se muestran las estimaciones de efectividad de cada uno donde se observa que la de los tres es igual esto se debe a que los parámetros de distribución son iguales. Aún así se puede decir que el que tuvo una mejor solución fue con división del 66% ya que construyó el mismo modelo en 0.02 segundos en comparación de los otros dos que lo hicieron en 0.03 segundos.

Cuadro 3.51: Resultados obtenidos de los experimentos con C4.5

	Instancias clasificadas correctamente (%)	Tiempo para construir el modelo (s)
Validación cruzada 10 folds	55.6071	0.03
División 66%	55.6071	0.02
Suplied test	55.6071	0.03

Al observar que ninguno de los experimentos realizados con esta prueba no lograron clasificar correctamente al menos en un 90% se puede descartar este modelo para la clasificar con este tipo de datos.

3.10 Pruebas Bagging con selección de atributos

Los experimentos que se realizaron con este algoritmo se hicieron bajo las mismas condiciones que en el apartado 3.5, es decir utilizando como algoritmo base el C4.5 con 20 iteraciones pero esta vez solo con tres atributos.

3.10.1 Bagging utilizando validación cruzada a 10 Folds

El tiempo en que se construyó el modelo fue de 1.64 segundos. En el cuadro 3.52 se muestra el rendimiento que tuvo el experimento notando que la efectividad fue de 44.5357% que fue calculado con la ecuación 3.1 mientras que el de los incorrectos se obtuvo de restar del 100% la efectividad obtenida siendo esta de 35.4643%.
Por su parte el cuadro 3.53 es la matriz de confusión obtenida después de probar el modelo con los datos de prueba donde se puede observar que las dos clases se clasificaron de igual forma ya que mientras la clase Yes tuvo una efectividad de 65.6% la clase No obtuvo un 63.2%.

Cuadro 3.52: Evaluación Bagging con selección de atributos sobre los datos de prueba (10 folds)

	Numero de instancias	Porcentaje
Correctos	1807	64.5357%
Incorrectos	993	35.4643%

Cuadro 3.53: Matriz de confusión Bagging con selección de atributos (10 folds)

Clases	Correctos	Errores	Probabilidad
Yes	966	505	0.657
No	841	488	0.633
Totales	1807	993	0.645357

Los resultados que se obtuvieron de esta prueba no cumplieron las expectativas de que pudieran alcanzar el 100% de efectividad en general es decir que las dos clases tuvieran una estimación de exactitud mayor del 90%.

3.10.2 Bagging utilizando split sample del 66%

El modelo que se produjo de este experimento esta formado por 20 árboles y su tiempo de construcción fue de 1.56 segundos.

Evaluación de los resultados con el árbol generado

La estimación de efectividad del modelo fue de 64.5357% como se puede apreciar en el cuadro 3.54 con un error de 35.4643% que es demasiado alto como para poder considerarlo como bueno ya que el ideal seria de 0%.

En el cuadro se presenta la matriz de confusión que se calculó de igual forma como se ha venido haciendo en los apartados anteriores y se puede observar que la clase que mejor fue clasificada fue la de Yes con 966 instancias mientras que la clase No solo lo logró con 841 es nos dice que si se deseara probar una nueva instancia tendríamos una probabilidad de que la decisión sea correcta si clasifica como Yes.

Cuadro 3.54: Evaluación Bagging con selección de atributos (split 66%)

	Numero de instancias	Porcentaje
Correctos	1807	64.5357%
Incorrectos	993	35.4643%

Cuadro 3.55: Matriz de confusión Bagging con selección de atributos (split 66%)

Clases	Correctos	Errores	Probabilidad
Yes	966	505	0.657
No	841	488	0.633
Totales	1807	993	0.645357

3.10.3 Bagging utilizando conjunto de entrenamiento

Evaluación de los resultados de las proporciones de clases

Este modelo se construyó en 1.31 segundos y se tuvo una baja efectividad de exactitud en cuanto la clasificación con solo un 64.5357% de efectividad como lo muestra el cuadro 3.56, pero una forma un poco más detallada de saber como se comportó es viendo la matriz de confusión que se muestra en el cuadro 3.57 ya que se presenta la efectividad que tuvo cada una de las clases como es en el caso de la de Yes cuya estimación de exactitud fue de 65.7% y un error de 63.3%.

Cuadro 3.56: Evaluación Bagging con selección de atributos (conjunto de entrenamiento)

	Numero de instancias	Porcentaje
Correctos	1807	64.5357%
Incorrectos	993	35.4643%

Cuadro 3.57: Matriz de confusión Bagging con selección de atributos (conjunto de entrenamiento)

Clases	Correctos	Errores	Probabilidad
Yes	966	505	0.657
No	841	488	0.633
Totales	1807	993	0.645357

Los resultados que se obtuvieron de este experimento no fueron los buscados para poder cataloga al modelo como bueno dado que su porcentaje de exactitud fue bajo (64.5357%) cuando el que se desea obtener es mayor al 901%.

3.10.4 Comparación de Resultados Bagging

En este apartado se comentaran los resultados obtenidos de los experimentos de las secciones 3.10.3, 3.10.2 y 3.10.1 los cuales se han puesto en el cuadro 3.58 en la que se puede observar que los tres modelos clasificaron de igual forma las instancias de prueba esto se debe a que los tres generaron los mismos árboles pero en el parámetro que difieren es en el del tiempo donde se puede escoger aquel con menor tiempo que fue el de suplied test con solo 1.31 segundos pero esto no quiere decir que el modelo sea el ideal para el tipo de datos que se ocupó ya que todos los modelo clasificaron pobremente con un 64.5357%.

Cuadro 3.58: Resultados obtenidos de los experimentos con Bagging

	Instancias clasificadas correctamente (%)	Tiempo para construir el modelo (s)
Validación cruzada 10 folds	64.5357	1.64
División 66%	64.5357	1.56
Suplied test	64.5357	1.31

3.11 Pruebas RandomCommittee con selección de atributos

3.11.1 RandomCommittee utilizando validación cruzada a 10 Folds

Evaluación de los resultados con el árbol generado

El modelo se construyó en un tiempo de 0.77 segundos proporcionándonos buenos resultados como se puede ver en el cuadro 3.59 en la que se puede observar que el ero fue de solo 14.8571% y una efectividad de 85.1429% estos datos fueron calculados con la ecuación 3.1.

Además, se puede ver en la matriz e confusión del cuadro 3.60 que la clase que mejor se clasificó fue la de yes con un 88.9% que se aproximo de buena forma al que de esta buscando que debe ser superior al 90% aunque por otro lado no se pueda decir lo mismo de la clase No que solo obtuvo un 81% de efectividad.

Cuadro 3.59: Evaluación RandomCommittee con selección de atributos (10 folds)

	Numero de instancias	Porcentaje
Correctos	2384	85.1429%
Incorrectos	416	14.8571%

Cuadro 3.60: Matriz de confusión RandomCommittee con selección de atributos (10 folds)

Clases	Correctos	Errores	Probabilidad
Yes	1308	163	0.889
No	1076	253	0.81
Totales	2384	416	0.851429

Los resultados de este experimento fueron bastante alentadores ya que se está aproximando a lo que se requiere para poder considerarlo como un buen modelo faltandole solo un 4.8571% para poder alcanzar el 90% que se anda buscando.

3.11.2 RandomCommittee utilizando split sample 66%

La construcción de este modelo dilató 0.91 segundos proporcionando muy buenos resultados como se puede apreciar en el cuadro 3.61 que tiene que la efectividad de clasificación fue de 85.1429% con 2384 instancias de las 2800 que se ocuparon para probarlo.

El cuadro 3.71 nos da a conocer como que se clasificaron cada una de las clases por separado observando que para la clase Yes se tuvo una clasificación del 88.92% y un error de 11.08% que aunque no aparece en la matriz de confusión es fácil de calcular teniendo solo que restar del 100% que sería la clasificación ideal la que se tuvo en realidad, por otro lado la clase tuvo una menor estimación de efectividad pero no despreciable de 80.96%.

Cuadro 3.70: Evaluación RandomCommittee con selección de atributos (split 66%)

	Numero de instancias	Porcentaje
Correctos	2384	85.1429%
Incorrectos	416	14.8571%

Resultados con los datos de prueba:

Cuadro 3.71: Matriz de confusión RandomCommittee con selección de atributos (10 split 66%)

Clases	Correctos	Errores	Probabilidad
Yes	1308	163	0.8892
No	1076	253	0.8096
Totales	2384	416	0.851429

Cada uno de los resultados obtenidos de este experimento fueron alentadores aunque no del todo ya que no se a podido obtener el 90% o más de efectividad que requerimos para poder decir que un algoritmo se amoldo a los datos que se tienen.

3.11.3 RandomCommittee utilizando conjunto de entrenamiento

En este experimento se logró obtener la solución ideal de clasificación de las clases el tiempo en que se construyó este modelo fue de 1.31 segundos.

En el cuadro 3.72 se puede observar que el modelo logró clasificar en un 100% todas las instancias lo que quiere decir que no se tuvo ningún error en la clasificación de las 2800 instancias. En la matriz de confusión del cuadro 3.73 también se puede ver que no hubo error alguno de clasificación teniendo una probabilidad de 1 cada vez que se quiera clasificar una nueva instancia.

Cuadro 3.72: Evaluación RandomCommittee con selección de atributos (conjunto de entrenamiento)

	Numero de instancias	Porcentaje error
Correctos	2800	100%
Incorrectos	0	0%

Cuadro 3.73: Matriz de confusión RandomCommittee con selección de atributos (conjunto de entrenamiento)

Clases	Correctos	Errores	Probabilidad
Yes	1447	0	1
No	0	1353	1
Totales	1447	1353	1

Algo que se debe de hacer notar es que aunque se tenga un error del 0% no quiere decir que cuando se realicen predicciones con el modelo construido siempre se va a tener la certidumbre de que es una buena decisión ya que el algoritmo esta basado en probabilidades lo que quiere decir es que las predicciones que se realicen solo tienen mayor probabilidad de suceder.

3.11.4 Comparación de Resultados RandomCommittee

Para los experimentos efectuados en las secciones 3.11.3, 3.11.2 y 3.11.1 se puede concluir que el mejor fue cuando se ocupó la técnica de suplied test ya que nos proporcionó un 100% de efectividad que es la que se esperaba encontrar, el tiempo que tardaron cada uno de estos experimentos no tiene relevancia ya que los otros dos no lograron alcanzar la meta que se requería.

Capítulo 4

RESULTADOS

En este capítulo se pretende dar un panorama más amplio de los resultados obtenidos en cada experimento para esto se presentarán las diferencias y similitudes de los experimentos dependiendo de la técnica de validación ocupada.

Los primeros resultados que se analizarán serán los que se realizaron con validación cruzada a 10 folds para ello me apoyaré en el cuadro 4.1, en la que se muestra el porcentaje de instancias correctamente clasificadas y el tiempo en que se construyó el modelo.

Lo primero que destaca de los experimentos es que fueron los multiclasificadores los que mejores resultados proporcionaron pero con un mayor tiempo de creación mientras que los más rápidos fueron los creados con el algoritmo Naive Bayes pero que fueron los que peor resultado ofrecieron, los resultados de aplicar el algoritmo C4.5 fueron parecidos al anterior de ahí que se descarten estos algoritmos.

Cuadro 4.1: Resultados obtenidos de los experimentos con validación cruzada a 10 folds

	Instancias clasificadas correctamente (%)	Tiempo para construir el modelo (s)
C4.5	60.3029	0.13
C4.5*	57.8571	0.14
Naive Bayes	55.2143	0.03
Naive Bayes*	55.6071	0.03
Bagging	67.6429	2.64
Bagging*	64.5357	1.64
RandomCommittee	85.9643	2.66
RandomCommittee*	85.1429	0.77

Otra forma de poder ver los resultados es a partir de la gráficas de la figura 4.1 en la que se puede observa de mejor forma que los resultados no mejoraron con la selección de atributos y se puede ver el contraste en cuanto a las estimaciones de efectividad entre el que peor resultados nos proporciono que fue Naive Bayes con las de Random Committee que fueron los mejores.

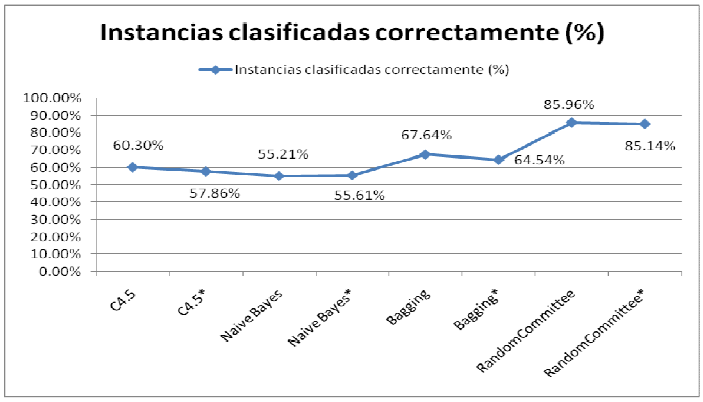

Figura 4.1 Porcentaje de instancias correctamente clasificadas con validación cruzada a 10folds

En la gráfica de la figura 4.2 se observan los tiempos de creación de cada uno de los modelos siendo los multiclasificadores los que más tardaron, otra cosa importante de señalar es la rapidez con la que se llevaron a cabo las pruebas con Naive Bayes.

En lo que se refiere a la selección de atributos se puede observar que para los multiclasificadores una importante diferencia en el tiempo de construcción siendo más eficiente cuando hubo menos atributos mientras que en los demás algoritmos el tiempo no tuvo gran diferencia.

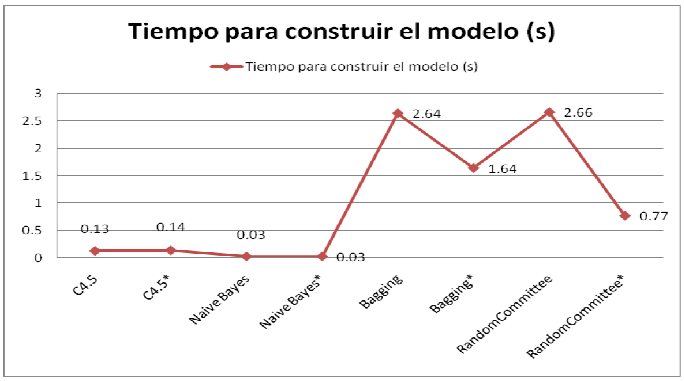

Figura 4.2 comparación de tiempos de creación de los modelos con validación cruzada a 10 folds

Para el método de validación de división del archivo al 66% se obtuvieron los resultados que se muestran en el cuadro 4.2 en la que se puede ver como nuevamente fueron los multiclasificadores los que mejores resultado nos dieron siendo Random Committee el mejor, por el otro lado fue Naive Bayes el que peores resultado proporcionó no variando mucho el algoritmo C4.5 comprobando que ambos algoritmos pueden tener el mismo rendimiento aunque no fue el mejor de ahí que estos se descarten como buenos algoritmos.

Cuadro 4.2: Resultados obtenidos de los experimentos con porcentaje Split 66%

	Instancias clasificadas correctamente (%)	Tiempo para construir el modelo (s)
C4.5	60.3029	0.19
C4.5*	57.8571	0.14
Naive Bayes	55.2143	0.03
Naive Bayes*	55.6071	0.02
Bagging	67.6429	2.59
Bagging*	64.5357	1.56
RandomCommittee	85.9643	1.33
RandomCommittee*	85.1429	0.91

En la gráfica de la figura 4.3 se puede ver el impacto que tuvo la selección de atributos en cuestión del tiempo de construcción siendo más notorio en los algoritmos Bagging y Random Committee reduciéndose cuando el experimento se realizó con tres atributos tal es el caso del primero en el que se logró disminuir el tiempo de construcción en más de un segundo sin variar de manera extrema en los resultados.

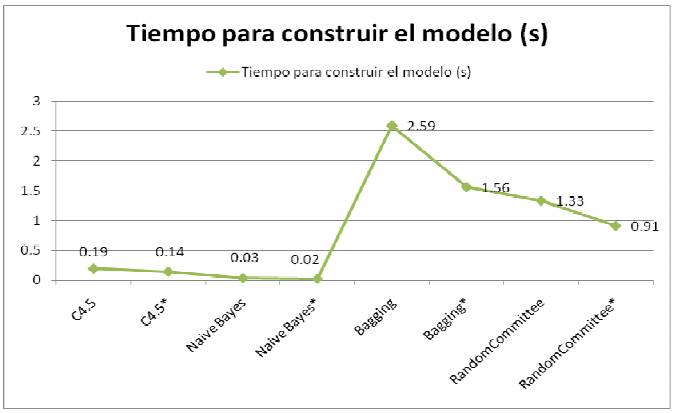

Figura 4.3 comparación de tiempos de creación de los modelos con split sample 66%

Los resultados para los experimentos usando suplied test se presentan en el cuadro 4.3 en la que se ve que el algoritmito RandommCommitte con tres atributos, alcanzó el maximo valor de exactitud en un tiempo de 1.31 segundos el segundo mejor fue nuevamente RandomCommittee pero con 6 atributos por lo que se puede seleccionar este algoritmo con el método de validación suplied test para la costrucción de modelos con datos financieros.

Cuadro 4.3: Resultados obtenidos de los experimentos con porcentaje Split 66%

	Instancias clasificadas correctamente (%)	Tiempo para construir el modelo (s)
C4.5	60.3029	0.245
C4.5*	57.8571	0.14
Naive Bayes	55.2143	0.02
Naive Bayes*	55.6071	0.03
Bagging	67.6429	2.58
Bagging*	64.5357	1.31
RandomCommittee	85.9643	1.16
RandomCommittee*	100	1.31

En la figura 4.4 se muestran los tiempos de construcción de los modelos donde se puede apreciar que el algoritmo más rápido es Naive Bayes aunque los resultados que se obtuvieron para la clasificación fueron malos. Nuevamente la grafica nos hace ver la diferencia entre los tiempos cuando el experimento se realiza con 3 atributos que cuando se utilizan los 6 sobre todo en el algoritmo de bagging donde se redujo el tiempo poco menos de la mitad.

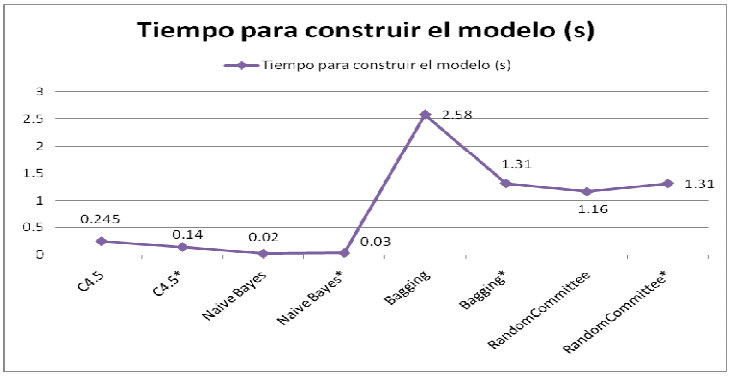

Figura 4.4 Comparación de tiempos de creación de los modelos con suplied test

Otra forma de poder decidir entre cual algoritmo se adaptó de mejor forma a los datos es comparando los errores de clasificación de todos los experimentos para esto me apoyaré en la gráfica de la figura 4.5 donde se puede se puede observar que para cada uno de los algoritmos se tuvieron los mismos errores para cada uno de los métodos de validación que se ocuparon a excepción del método RandomCommitte donde para la validación con suplied test se obtuvo el error de cero.

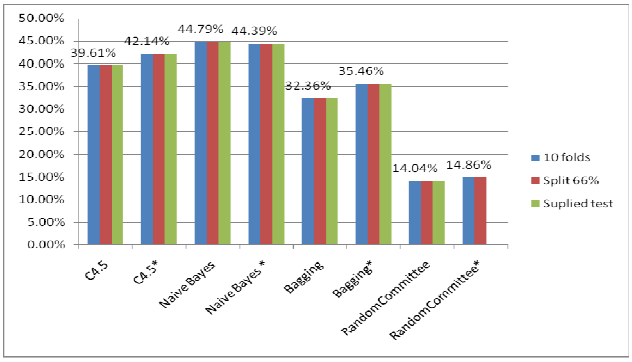

Figura 4.5 Error cometido en cada experimento

Capítulo 5

CONCLUSIONES

El problema de predicción que se planteó en la introducción se logró resolver de forma efectiva, toda vez que al probar el modelo construido con el algoritmo Random Committee, el objetivo de verificar si el índice se elevaba en un 4 % o más dentro de los 63 días comerciales siguientes, se clasificó de forma correcta con un 100% de efectividad como lo muestra la tabla 4.3 del capitulo anterior.

En consecuencia de lo anterior cada día de comercio que se probó fue clasificado en la categoría "de compra" si el objetivo era positivo o la categoría "de no-compra" en el caso que era negativo, sin, error como se verifica en la gráfica de la figura 4.5

De cada uno de los experimentos se observó que las técnicas estadísticas son fundamentales a la hora de validar las hipótesis y analizar los datos, proporcionando herramientas para cuantificar adecuadamente la incertidumbre resultante de la inferencia de patrones, a partir de datos particulares, es decir calcular el error en la clasificación de las nuevas instancias.

Por lo que respecta a la minería de datos y al descubrimiento del conocimiento (KDD) se observó que contribuyeron a la toma de decisiones tácticas y estratégicas; proporcionando un sentido automatizado para la generación del conocimiento y por ende a la toma acertada de decisiones para dar respuestas a las preguntas complejas de Inteligencia de Negocios.

Los multiclasificadores tienen aplicación en diversas áreas, la combinación apropiada de dos o más clasificadores puede proporcionar una predicción más robusta, más confiable y eficiente que el uso de un único clasificador como se visualiza en cada una uno de lo gráficos que se muestran en las figuras 5.1, 5.2 y 5.3, siendo estos los que lograron reducir de forma considerable el error en la clasificación.

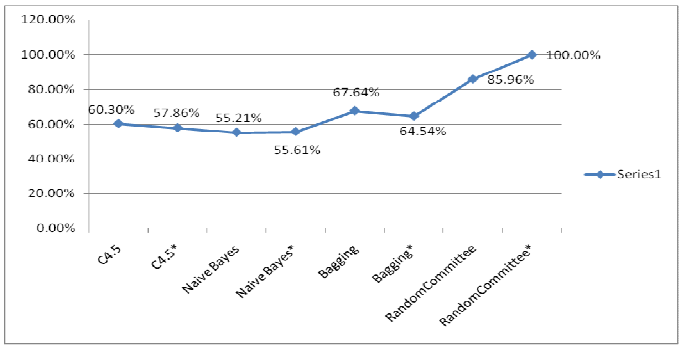

Figura 5.1 Porcentaje de instancias correctamente clasificadas con validación cruzada a 10folds y Split de 66%

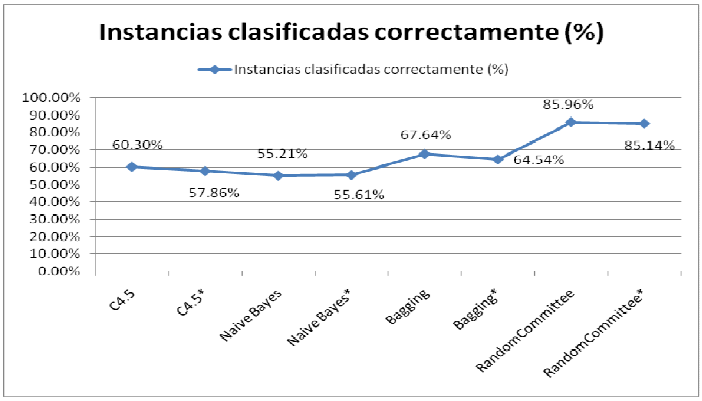

Figura 5.2 Porcentaje de instancias correctamente clasificadas suplied test

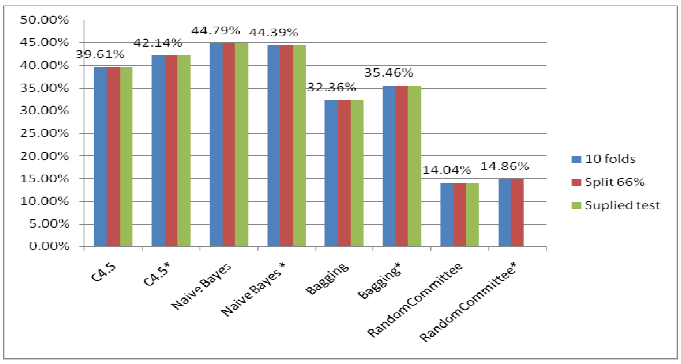

Figura 5.3 Error cometido en cada experimento

Algo importante de señalar en base a los resultados obtenidos es que la naturaleza de cada uno de los algoritmos de aprendizaje influyó en la precisión de la clasificación de un mismo conjunto de datos.

Del tiempo de construcción de cada uno de los modelos se puede decir que aquellos algoritmos que ocupan un solo conjunto de datos para su entrenamiento los métodos más rápidos en comparación con los multiclasificadores, destacando Naive Bayes; en el cual se muestra su independencia de clases en la construcción del modelo no importando como se realizaran los experimentos y siempre se usaron los mismos parámetros de distribución.

Bibliografía

- Morgan Kaufmann, *Data Mining Practical Machine Learning Tools and Techniques,* 2nd edición, USA, Morgan Kaufmann Publishers 2005.

- Mehmed Kantardzic, *Data Mining - Concepts, Models, Methods, and Algorithms*, 1er. Edicion, John Wiley & Sons 2003.

- Fayyad, Usama M. Grinstein, Georges G. Wierse, Andreas., *Information Visualization In Data Mining And Knowledge Discovery*, San Francisco: MK/Morgan Kaufmann Publishers, c2002.

- Dr. Ivo Humberto Pineda Torres, *Minería de datos y almacenes de datos,* Puebla 2007.

Publicaciones:

- María N. Moreno García, Luis A. Miguel Quintales, Francisco J. García Peñalvo y M. José Polo Martín, *Aplicación de técnicas de minería de datos en la construcción y validación de modelos predictivos y asociativos a partir de especificaciones de requisitos de software,* Universidad de Salamanca. Departamento de Informática y Automática 2004

- Saddys Segrera Francia, María N. Moreno García. *Multiclasificadores: Métodos y Arquitecturas.* Universidad de Salamanca. Departamento de Informática y Automática 2006.

- Christian A. Martínez, *Un vistazo a la minería de datos,* Instituto Tecnológico de estudios superiores de Monterrey campus Puebla.