UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
Facultad 5

TÍTULO: Comparación de algoritmos de clasificación y agrupamiento aplicando técnicas de Minería de datos.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores**:** Karina Silveira Martineaux.
Reidelendy Fernández Pérez.

Tutor: Dr. Ernesto González Díaz.

Ciudad de la Habana, julio 2008
Año 50 de la Revolución.

"Lo que sabemos es una gota de agua; lo que ignoramos es el océano."

Isaac Newton

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____
Karina Silveira Martineaux
(autora)

_____
Reidelendy Fernández Pérez
(autora)

_____
Dr. Ernesto González Díaz.
(Tutor)

DATOS DE CONTACTO

Ernesto González Díaz: Ingeniero Informático graduado en 1995 en la CUJAE, Master en Ciencias en Informática Aplicada graduado en 1999 en la CUJAE, Doctor en Ciencias Matemáticas graduado de la Universidad de la Habana en el 2001. Profesor del Departamento de Inteligencia Artificial y Sistemas Digitales de la Facultad de Ingeniería Informática de la CUJAE. Ha impartido clases de varias asignaturas de la carrera Ingeniería Informática en la CUJAE y en la UCI, ha impartido clases en la Maestría de Informática Aplicada y en la Maestría en Nuevas Tecnologías en la Educación de la que también fue coordinador. Ha sido tutor de más 30 tesis de Ingeniería Informática y más de 40 tesis de maestría.

AGRADECIMIENTOS

Quiero agradecer a mi familia y en especial a mi mamá por apoyarme siempre y servirme de guía procurando en todo momento mi superación educacional para ser en un futuro una persona independiente.

A Lázara, Hassan, Adrián Carlos y otras personas que me ayudaron mucho en mis estudios e hicieron posible realizar este sueño.

Agradecer a mis amigas Magdelis, Marbelis y Reidelendy que es mi compañera de tesis que desde luego siempre me ha apoyado y con su gran ayuda se ha convertido en un logro el sueño de graduarme, les aseguro que las voy a extrañar mucho y gracias por ser tan consecuentes y atentas conmigo.

Agradecer al profesor Millet su incondicional apoyo cuando necesité su ayuda, por sus consejos y ser siempre una persona servicial y atenta conmigo.

Quiero agradecer a nuestra Revolución por darnos la oportunidad de estudiar en una universidad de excelencia y en especial a nuestro Comandante Fidel Castro.

Karina

AGRADECIMIENTOS

Quiero agradecer a mis padres por toda la confianza depositada en mí y por todo el apoyo que me han dado siempre en los momentos más difíciles de mi carrera.

A Lidia y a Titi por acogerme como una hija más, por ayudarme incondicionalmente en mis estudios y por estar siempre a mi lado en las buenas y en las malas.

A Lisy por ser mi amiga y hermana durante todo este tiempo.

A Dulce, Prío, Pérez, Juanito, Mino y demás familiares por su preocupación constante y su eterno apoyo.

Quiero agradecer a mis compañeros de aula y todos mis amigos por los momentos tan lindos que hemos compartido.

Agradecerle a Lázara, Adrián Carlos y a Hassan por brindarme su ayuda incondicional en mis estudios.

Agradecerle al profesor Millet por la ayuda brindada durante el desarrollo de este trabajo.

Agradecerles a todas esas personas que de una forma u otra hicieron realidad este sueño.

Por último, y no por eso la menos importante, agradecerle a la Revolución y a nuestro Comandante Fidel Castro por haberme dado la oportunidad de estudiar en esta Universidad de excelencia.

Rendy

DEDICATORIA

De Karina:

Dedico especialmente este trabajo de diploma a mi madre porque más que mío este es su sueño y por ser mi gran amiga en todos los momentos.

De Rendy:

Dedico este trabajo de diploma a mis padres por ser mis guías a lo largo de toda mi carrera y de mi vida.

RESUMEN

Las técnicas de clasificación tienen una gran importancia en los procesos de descubrimiento de conocimiento porque pueden identificar nuevos grupos o clases de objetos en bases de datos. Así, existen métodos de aprendizaje supervisado y métodos de aprendizaje no supervisado, que han resultado ser muy útiles en grandes bases de datos, ya sean desconocidas, no etiquetadas o poco estructuradas. En este trabajo se analizan y comparan diferentes algoritmos de clasificación y agrupamiento disponibles en el software WEKA, de Minería de datos, con el objetivo de determinar el de mayor eficiencia. Estos se aplican a una muestra de la base de datos de gestión académica AKADEMOS. Los resultados de esta investigación, pueden ayudar a determinar eficientemente el rendimiento académico de los alumnos que ingresan a la carrera y elevar la calidad de la educación en la Universidad de las Ciencias Informáticas (UCI).

Palabra clave: Algoritmos de clasificación, Algoritmos de agrupamiento, Aprendizaje no supervisado, Aprendizaje Supervisado, Minería de datos.

TABLA DE CONTENIDOS

INTRODUCCIÓN

En los últimos años se ha visto un gran crecimiento en la generación de información. Cada día, la gente almacena gran cantidad de información representada como datos para posteriormente realizar un análisis y administración de estos. El principal objetivo de interactuar con estos datos es clasificarlos en pequeños grupos que describan sus características principales, basándose en la similitud o diferencia entre ellos. El análisis de grandes volúmenes de datos no sólo puede brindar información adicional, sino también conocimiento nuevo.

La Minería de datos es considerada uno de los desarrollos más prometedores interdisciplinariamente en la industria de la información. La Minería de datos representa la posibilidad de buscar información dentro de un conjunto de datos con la finalidad de extraer información nueva y útil que se encuentra oculta en grandes volúmenes de datos.

El agrupamiento y la clasificación son dos de las principales tareas en el proceso de Minería de datos que se utilizan para predecir futuras tendencias y comportamientos, que permiten la toma de decisiones en casos determinados. Es de mucha utilidad aplicar estas tareas para encontrar automáticamente patrones de comportamiento y relaciones entre los datos que se van a probar.

Cuando se aprende de ejemplos, casos o datos conocidos, generalmente lo que se intenta es tomar una decisión sobre nuevos casos, es decir se pretende aplicar lo que se ha aprendido en base a varias situaciones anteriores, a una situación particular nueva. Carece de sentido común pensar que ante una nueva situación se deberá actuar como se hizo en situaciones anteriores parecidas o similares, si en estas se tuvo éxito. En consecuencia, parece una regla bastante clara y natural que puede ser aplicada a multitud de tareas de la Minería de datos.

Por ejemplo, en las tareas de clasificación de la Minería de datos, se puede asignar un nuevo elemento a una clase de acuerdo a la similitud de sus características con las de los elementos componentes de esa clase. De la misma manera, en las tareas de agrupamiento, se asignará un nuevo elemento al grupo donde estén los individuos más similares. La diferencia fundamental en estas tareas radica en que en la clasificación se parte de la predeterminación de las clases, teniendo en cuenta las características de los elementos que deben componer la clase, mientras que en la agrupación no se predeterminan clases, sino que las mismas se van obteniendo a partir de la similitud de las características de los elementos a agrupar.

Situación Problémica

En la UCI se manejan grandes volúmenes de información referente al proceso de gestión académica de los estudiantes. La Universidad cuenta con alrededor de 10 000 estudiantes de todas las provincias y municipios del país; los mismos presentan los más diversos orígenes sociales y académicos. Toda la información personal y docente de los estudiantes, desde hace cinco años se encuentra digitalizada y se mantiene en datos históricos que no brindan mayor utilidad que la de los reportes tradicionales.

El proceso de captación de los estudiantes que ingresan a la Universidad no analiza todos los factores que influyen en los resultados de los mismos una vez matriculados. En muchas ocasiones a los estudiantes matriculados no se les da el seguimiento que se debe y por este motivo en ocasiones pueden llegar a causar baja académica; o en otros casos no se captan alumnos con mayor probabilidad de rendimiento docente para formar parte de grupos de investigación; o para que los profesores puedan realizar un mayor trabajo, con vistas a lograr que los mismos se incorporen más tarde a estos grupos investigativos o eleven su preparación.

En esta institución se realizó un estudio de los factores que influyen en los resultados académicos de los estudiantes utilizando técnicas de clasificación y agrupamiento, donde se obtuvieron patrones y reglas que pueden mejorar el proceso académico, pero no se conoce con certeza la eficiencia y resultados de los algoritmos representativos de estas técnicas.

Por lo anteriormente expuesto, la presente investigación parte del siguiente **problema científico**: ¿Cómo realizar una comparación de los algoritmos más representativos de las técnicas de agrupamiento y clasificación para determinar su eficiencia a partir de la supervisión de su ejecución?

Para dar respuesta a esta interrogante este trabajo incluye como **objeto de estudio** el análisis de la eficiencia de algunos de los algoritmos más representativos de las técnicas de clasificación y agrupamiento y como **campo de acción** los algoritmos más representativos de las técnicas de clasificación y agrupamiento que proporciona la herramienta WEKA, aplicados a una muestra de la base de datos de los estudiantes de la UCI.

Este trabajo se plantea el siguiente **objetivo general**: evaluar cualitativamente la eficiencia de algunos de los algoritmos más representativos de las técnicas de clasificación y agrupamiento. Para cumplimentar este objetivo se desarrollan los siguientes **objetivos específicos**:

- Realizar una búsqueda bibliográfica que permita establecer el marco teórico conceptual.
- Analizar las particularidades de los algoritmos de clasificación y agrupamiento seleccionados, así como su implementación en la herramienta WEKA.
- Evaluar los algoritmos seleccionados en la herramienta WEKA.
- Realizar la comparación de los algoritmos.

Como **resultado** de esta investigación se pretende obtener un estudio comparativo entre los algoritmos más representativos de las técnicas de clasificación y agrupamiento que permita determinar con mayor certeza patrones y reglas que contribuyan a mejorar el rendimiento académico de los estudiantes.

La **Estructuración del Contenido** está dada por tres capítulos.

En el Capítulo I se abordarán temas referentes a la Minería de Datos y los proyectos de Descubrimiento de Conocimiento en Bases de Datos en general. Se explicarán las características generales de las técnicas de clasificación y agrupamiento. También se tratarán aspectos sobre las herramientas que existen para realizar proyectos de minería; principalmente WEKA, demostrando sus potencialidades.

En el Capítulo II se explicarán algunos conceptos de distancia utilizados por los algoritmos para encontrar similitudes o diferencias entre los datos. Además, se destacarán las particularidades de los algoritmos de

clasificación y agrupamiento seleccionados para realizar la comparación. También se hará referencia a la implementación que propone la herramienta WEKA para cada uno de los algoritmos seleccionados.

En el Capítulo III se analizará el caso de estudio seleccionado para realizar la comparación. También se hará la evaluación de los algoritmos de clasificación y agrupamiento sobre la herramienta WEKA y se realizará la comparación de los mismos.

CAPÍTULO 1. Fundamentación Teórica.

Introducción

En la actualidad, muchas de las decisiones importantes que se toman alrededor del mundo se basan en observaciones y/o eventos que han sido previamente registrados de alguna forma en una base o modelo de datos.

En la gran mayoría de las situaciones, los datos que se llegan a almacenar pueden contener demasiadas propiedades o atributos que causan que la información sea complicada de visualizar a primera instancia. En otros casos estas bases de datos pueden llegar a almacenar miles o millones de instancias de datos, las cuales pueden llegar a variar después de cientos o miles de muestras. Muchos de estos datos pueden llegar a pasar desapercibidos por el ser humano y estar siempre presentes en las más difíciles y críticas situaciones [6].

1.1 Descubrimiento de Conocimiento en Bases de Datos (KDD)

En los últimos años, ha existido un gran crecimiento en nuestras capacidades de generar y recolectar datos, debido básicamente al gran poder de procesamiento de las máquinas [14].

Sin embargo, dentro de estas enormes masas de datos existe una gran cantidad de información oculta, de gran importancia estratégica, a la que no se puede acceder por las técnicas clásicas de recuperación de la información. El descubrimiento de esta información oculta es posible gracias a la Minería de Datos (Data Mining), que entre otras sofisticadas técnicas aplica la inteligencia artificial para encontrar patrones y relaciones dentro de los datos permitiendo la creación de modelos, es decir, representaciones abstractas de la realidad, pero es el descubrimiento del conocimiento (KDD, por sus siglas en inglés) que se encarga de la preparación de los datos y la interpretación de los resultados obtenidos, los cuales dan un significado a estos patrones encontrados [14].

Así el valor real de los datos reside en la información que se puede extraer de ellos, información que ayude a tomar decisiones o mejorar nuestra comprensión de los fenómenos que nos rodean. Hoy, más que nunca, los métodos analíticos avanzados son el arma secreta de muchos negocios exitosos. Empleando métodos analíticos avanzados para la explotación de datos, los negocios incrementan sus ganancias, maximizan la eficiencia operativa, reducen costos y mejoran la satisfacción del cliente.

1.1.1 Concepto del KDD

El Descubrimiento de Conocimiento en Bases de Datos (KDD) apunta a procesar automáticamente grandes cantidades de datos para encontrar conocimiento útil en ellos, de esta manera permitirá al usuario el uso de esta información valiosa para su conveniencia [14].

El objetivo fundamental del KDD es encontrar conocimiento válido, relevante y nuevo sobre un fenómeno o actividad mediante algoritmos eficientes, dadas las crecientes órdenes de magnitud en los datos. Al mismo tiempo hay un profundo interés por presentar los resultados de manera visual o al menos de manera que su interpretación sea muy clara. Otro aspecto es que la interacción humano-máquina deberá ser flexible, dinámica y colaboradora [14].

El resultado de la exploración deberá ser interesante y su calidad no debe ser afectada por mayores volúmenes de datos o por ruido en los datos. En este sentido, los algoritmos de descubrimiento de información deben ser altamente robustos.

1.1.2 Metas

Las metas del KDD son:
• Procesar automáticamente grandes cantidades de datos crudos.
• Identificar los patrones más significativos y relevantes.
• Presentarlos como conocimiento apropiado para satisfacer las metas del usuario [14].

1.2 Minería de datos. Conceptos e Historia.

La Minería de Datos (DM) por las siglas en inglés Data Mining es el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos. Las herramientas de la Minería de datos predicen futuras tendencias y comportamientos, permitiendo en los negocios la toma de decisiones [13].

Aunque desde un punto de vista académico el término Minería de datos es una etapa dentro de un proceso mayor llamado extracción de conocimiento en bases de datos, en el entorno comercial, ambos términos se usan de manera indistinta. Lo que en verdad hace la Minería de datos es reunir las ventajas de varias áreas como la Estadística, la Inteligencia Artificial, la Computación Gráfica, las Bases de Datos y el Procesamiento Masivo, principalmente usando como materia prima las bases de datos.

Una definición tradicional es la siguiente: Un proceso no trivial de identificación válida, novedosa, potencialmente útil y entendible de patrones comprensibles que se encuentran ocultos en los datos. Desde el punto de vista empresarial, se define como: La integración de un conjunto de áreas que tienen como propósito la identificación de un conocimiento obtenido a partir de las bases de datos que aporten un sesgo hacia la toma de decisiones [14].

La idea de la Minería de datos no es nueva. Ya desde los años sesenta los estadísticos manejaban términos como data fishing, data mining o data archaeology con la idea de encontrar correlaciones sin una hipótesis previa en bases de datos con ruido. A principios de los años ochenta, Rakesh Agrawal, Gio Wiederhold, Robert Blum y Gregory Piatetsky-Shapiro, entre otros, empezaron a consolidar los términos de Minería de datos y KDD. A finales de los años ochenta sólo existían un par de empresas dedicadas a esta tecnología; en el 2002 existían más de 100 empresas en el mundo que ofrecen alrededor de 300 soluciones. Las listas de discusión sobre este tema las forman investigadores de más de ochenta países. Esta tecnología ha sido un buen punto de encuentro entre personas pertenecientes al ámbito académico y al de los negocios [14].

Durante el desarrollo de un proyecto de Minería de datos se usan diferentes aplicaciones software en cada etapa, que pueden ser estadísticas, de visualización de datos o de inteligencia artificial, principalmente. Actualmente existen aplicaciones o herramientas comerciales de Minería de datos muy poderosas que contienen un sinfín de utilidades que facilitan el desarrollo de un proyecto. Sin embargo, casi siempre acaban complementándose con otra herramienta.

1.2.1 Principales características y objetivos de la Minería de Datos.

Explorar los datos que se encuentran en las profundidades de las bases de datos, como los almacenes de datos, que algunas veces contienen información almacenada durante varios años [14].

• En algunos casos, los datos se consolidan en un almacén de datos y en mercados de datos; en otros, se mantienen en servidores de Internet e Intranet.

El entorno de la Minería de datos suele tener una arquitectura cliente-servidor.

• Las herramientas de la Minería de datos ayudan a extraer el mineral de la información enterrado en archivos corporativos o en registros públicos, archivados.

• El minero es, muchas veces, un usuario final con poca o ninguna habilidad de programación, facultado por barrenadoras de datos y otras poderosas herramientas indagatorias para efectuar preguntas ad-hoc y obtener respuestas rápidamente.

• Hurgar y sacudir a menudo implica el descubrimiento de resultados valiosos e inesperados.

• Las herramientas de la Minería de datos se combinan fácilmente y pueden analizarse y procesarse rápidamente.

• Debido a la gran cantidad de datos, algunas veces resulta necesario usar procesamiento en paralelo para la Minería de datos.

• La Minería de datos produce cinco tipos de información:

— Asociaciones.

— Secuencias.

— Clasificaciones.

— Agrupamientos.

— Pronósticos.

• Los mineros de datos usan varias herramientas y técnicas.

1.3 Técnicas de la Minería de Datos.

Las técnicas de Minería de datos pueden dividirse en 2 categorías principales: predictivas (aprendizaje supervisado) y descriptivas (aprendizaje no supervisados) [7].

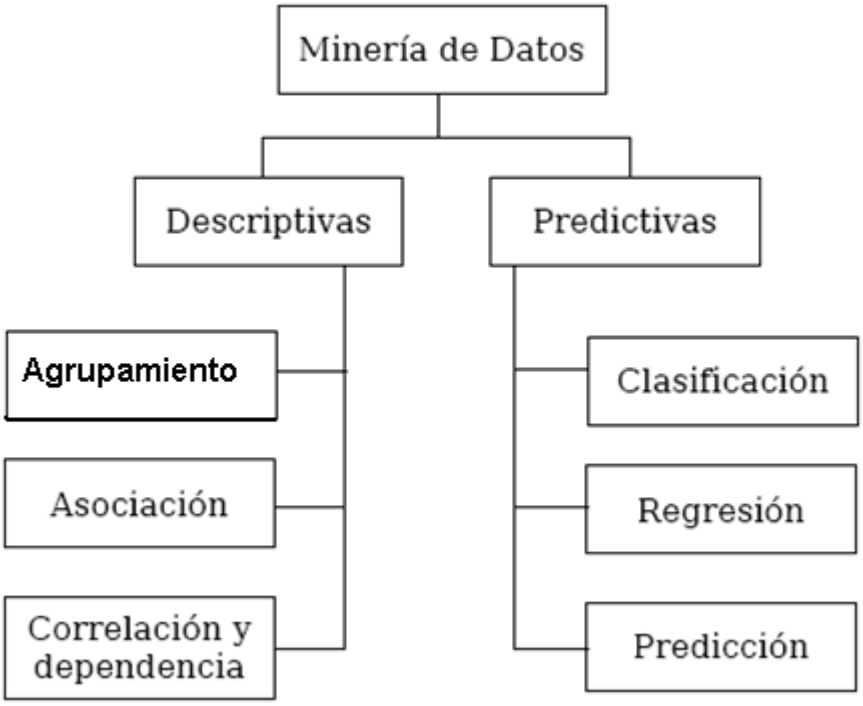


Figura 1.1: Taxonomía de las técnicas de minería de datos.

En las tareas predictivas, cada observación incluye un valor de la clase a la que corresponde. El objetivo de estas tareas es predecir el valor de un atributo particular basado en los valores de otros atributos [7]. Algunas de las tareas predictivas son:

Agrupamiento: El agrupamiento/segmentación es la detección de grupos de individuos. Se diferencia de la clasificación en que en este caso, no se conocen ni las clases ni su número (aprendizaje no supervisado), con lo que el objetivo es determinar grupos o racimos diferenciados del resto.

Asociaciones: Una asociación entre dos atributos ocurre cuando la frecuencia de que se den dos valores determinados de cada uno conjuntamente, es relativamente alta.

Dependencias: Una dependencia funcional (aproximada o absoluta) es un patrón en el que se establece que uno o más atributos determinan el valor de otro.

En las tareas descriptivas, el conjunto de observaciones no tienen clases asociadas. El objetivo es derivar características (correlaciones, grupos, trayectorias, anomalías) que describan las relaciones entre los

datos. Estas tareas son comúnmente de exploración natural y frecuentemente requieren de técnicas de postprocesamiento para explicar los resultados [7]. Algunas de las tareas descriptivas son:

Clasificación: Una clasificación se puede ver como el esclarecimiento de una dependencia, en la que el atributo dependiente puede tomar un valor entre varias clases, ya conocidas.

Regresión: El objetivo es predecir los valores de una variable continua a partir de la evolución sobre otra variable continua, generalmente el tiempo.

Predicción: Encuentra una clasificación de valores faltantes o sin conocimiento previo. Se refiere tanto a la predicción de valores en los datos como a la predicción de clases utilizando la identificación de distribuciones en los datos disponibles.

De estas tareas de Minería de datos, aquellas que son comúnmente utilizadas son: la clasificación, el agrupamiento y la asociación. A continuación se presenta una definición más formal de la clasificación y el agrupamiento [9].

1.4 Clasificación

Permite dividir un conjunto de datos en grupos mutuamente excluyentes de tal manera que cada miembro de un grupo esté lo "más cercano" posible a otro, y grupos diferentes estén lo "más lejos" posible uno del otro, donde la distancia está medida con respecto a variable(s) específica(s), las cuales se están tratando de predecir.

1.4.1 Técnicas de Clasificación

Tabla de Decisión

La tabla de decisión constituye la forma más simple y rudimentaria de representar la salida de un algoritmo de aprendizaje, que es justamente representarlo como la entrada.

Esta técnica consiste en seleccionar subconjuntos de atributos y calcular su precisión para predecir o clasificar los ejemplos. Una vez seleccionado el mejor de los subconjuntos, la tabla de decisión estará formada por los atributos seleccionados (más la clase), en la que se insertarán todos los ejemplos de entrenamiento únicamente con el subconjunto de atributos elegido. Si hay dos ejemplos con exactamente

los mismos pares *atributo-valor* para todos los atributos del subconjunto, la clase que se elija será la media de los ejemplos (en el caso de una clase numérica) o la que mayor probabilidad de aparición tenga (en el caso de una clase simbólica) [12].

Árboles de Decisión

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 [12].
Algunos algoritmos que pertenecen a esta clasificación son: ID3 y C4.5.

Reglas de Clasificación

Las técnicas de Inducción de Reglas surgieron hace más de dos décadas y permiten la generación y contraste de árboles de decisión, o reglas y patrones a partir de los datos de entrada. La información de entrada será un conjunto de casos donde se ha asociado una clasificación o evaluación a un conjunto de variables o atributos. Con esa información estas técnicas obtienen el árbol de decisión o conjunto de reglas que soportan la evaluación o clasificación. En los casos en que la información de entrada posee algún tipo de “ruido" o defecto (insuficientes atributos o datos, atributos irrelevantes o errores u omisiones en los datos) estas técnicas pueden habilitar métodos estadísticos de tipo probabilístico para generar árboles de decisión recortados o podados. Esta técnica suele llevar asociada una alta interacción con el analista de forma que éste pueda intervenir en cada paso de la construcción de las reglas, bien para aceptarlas, bien para modificarlas [12].
Algunos de los algoritmos que pertenecen a esta clasificación son: 1R, PRISM y PART.

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 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.
Uno de los algoritmos que pertenece a esta clasificación es el clasificador de Naive Bayes.

Aprendizaje Basado en Ejemplares

El aprendizaje basado en ejemplares o instancias tiene como principio de funcionamiento, en sus múltiples variantes, el almacenamiento de ejemplos: en unos casos todos los ejemplos de entrenamiento, en otros sólo los más representativos, en otros los incorrectamente clasificados cuando se clasifican por primera vez. La clasificación posterior se realiza por medio de una función que mide la proximidad o parecido. Dado un ejemplo para clasificar se le clasifica de acuerdo al ejemplo o ejemplos más próximos. El bias (sesgo) que rige este método es la proximidad; es decir, la generalización se guía por la proximidad de un ejemplo a otros [12].

Se han enumerado ventajas e inconvenientes del aprendizaje basado en ejemplares, pero se suele considerar no adecuado para el tratamiento de atributos no numéricos y valores desconocidos. Las mismas medidas de proximidad sobre atributos simbólicos suelen proporcionar resultados muy dispares en problemas diferentes [12].
Algunos de los algoritmos que pertenecen a esta clasificación son: K vecinos (KNN) [10] y K estrella (K*).

1.5 Agrupamiento

También llamado clustering, permite la identificación de tipologías o grupos donde los elementos guardan gran similitud entre sí y muchas diferencias con los de otros grupos. Así se puede segmentar el colectivo de clientes, el conjunto de valores e índices financieros, el espectro de observaciones astronómicas, el conjunto de zonas forestales, el conjunto de empleados y de sucursales u oficinas, etc. La segmentación está teniendo mucho interés desde hace tiempo dadas las importantes ventajas que aporta al permitir el

tratamiento de grandes colectivos de forma pseudoparticularizada, en el más idóneo punto de equilibrio entre el tratamiento individualizado y aquel totalmente masificado.

Las herramientas de segmentación se basan en técnicas de carácter estadístico, de empleo de algoritmos matemáticos, de generación de reglas y de redes neuronales para el tratamiento de registros. Para otro tipo de elementos a agrupar o segmentar, como texto y documentos, se usan técnicas de reconocimiento de conceptos. Esta técnica suele servir de punto de partida para después hacer un análisis de clasificación sobre los grupos [12].

La principal característica de esta técnica es la utilización de una medida de similaridad que, en general, está basada en los atributos que describen a los objetos, y se define usualmente por proximidad en un espacio multidimensional. Para datos numéricos, suele ser preciso preparar los datos antes de realizar Minería de datos sobre ellos, de manera que en primer lugar se someten a un proceso de estandarización.

El clustering o agrupamiento es partir el grupo de entrada en pequeños grupos, para este particionamiento se usan distancias. Tiene en cuenta todas las características de los datos. Las distancias más utilizadas son:

- Euclidiana, la más común.
- Manhattan.
- Hamming.

A través del cálculo de distancias se agrupan los elementos de acuerdo a los más cercanos según el cálculo. El agrupamiento es utilizado en la recuperación y extracción de información para realizar agrupamientos de conceptos cercanos de manera que tratando el grupo el resultado sea similar [11].

Características de los algoritmos de agrupamiento

Las características deseables de la mayoría de los algoritmos de agrupamiento son las siguientes [9]:
Escalabilidad. La mayoría de los algoritmos de agrupamiento trabajan de manera apropiada con un número pequeño de observaciones (hasta 200 aproximadamente), mientras que se necesita una gran escalabilidad para realizar agrupamiento de datos en bases con millones de observaciones. Habilidad para trabajar con distintos tipos de atributos. Muchos algoritmos se han diseñado para trabajar sólo con

datos numéricos, mientras que en una gran cantidad de ocasiones, es necesario trabajar con atributos asociados a tipos numéricos, binarios, discretos y alfanuméricos.

Descubrimiento de grupos con formas arbitrarias. La mayoría de los algoritmos de agrupamiento se basan en la distancia Euclidiana, lo que tiende a encontrar grupos todos con forma (circular) y densidad similares. Es importante diseñar algoritmos que puedan establecer grupos de formas arbitrarias.

Requerimientos mínimos en el conocimiento del dominio para determinar los parámetros de entrada. La herramienta no debería solicitarle al usuario que introduzca la cantidad de clases que quiere considerar, ya que dichos parámetros en muchas ocasiones no son fáciles de determinar, y esto haría que sea difícil controlar la calidad del algoritmo.

Habilidad para tratar con datos ruidosos. La mayoría de las BD contienen datos con comportamiento extraño, datos faltantes, desconocidos o erróneos. Algunos algoritmos de agrupamiento son sensibles a tales datos y pueden derivarlos a grupos de baja calidad.

Insensibilidad al orden de las observaciones de entrada. Algunos algoritmos son sensibles al orden en que se consideran las observaciones. Por ejemplo, para un mismo conjunto de datos, dependiendo del orden en que se analicen, los grupos devueltos pueden ser diferentes. Es importante entonces que el algoritmo sea insensible al orden de los datos, y que el conjunto de grupos devuelto sea siempre el mismo.

Alta dimensionalidad. Una BD o DW (DataWarehouse) puede contener varias dimensiones o atributos, por lo que es bueno que un algoritmo de agrupamiento pueda trabajar de manera eficiente y correcta no sólo en repositorios con pocos atributos, sino también en repositorios con un alto espacio dimensional, o gran cantidad de atributos.

Agrupamiento basado en restricciones. Es un gran desafío el agrupar los datos teniendo en cuenta no sólo el comportamiento, sino también que satisfagan ciertas restricciones [9].

Interpretación y uso. Los usuarios esperan que los resultados del agrupamiento sean comprensibles, fáciles de interpretar y de utilizar.

1.5.1 Algoritmos básicos de agrupamiento

Los algoritmos de agrupamiento varían entre sí por las reglas heurísticas que utilizan y el tipo de aplicación para el cual fueron diseñados. La mayoría de ellos se basa en el empleo sistemático de distancias entre vectores (objetos a agrupar) así como entre grupos que se van formando a lo largo del proceso de agrupamiento. Las características básicas por las que los algoritmos de agrupamiento pueden ser clasificados son en función de:

1. El tipo de dato que manejan (numérico, categórico y/o mixto).
2. El criterio utilizado para medir la similitud entre los puntos.
3. Los conceptos y técnicas de agrupamiento empleadas (ej. lógica difusa, estadísticas).

En la literatura existe una gran cantidad de técnicas de agrupamiento que varían de acuerdo a la arquitectura que utilizan. Una clasificación general divide los algoritmos en: agrupamiento particional, agrupamiento jerárquico, agrupamiento basado en densidad y agrupamiento basado en grid.

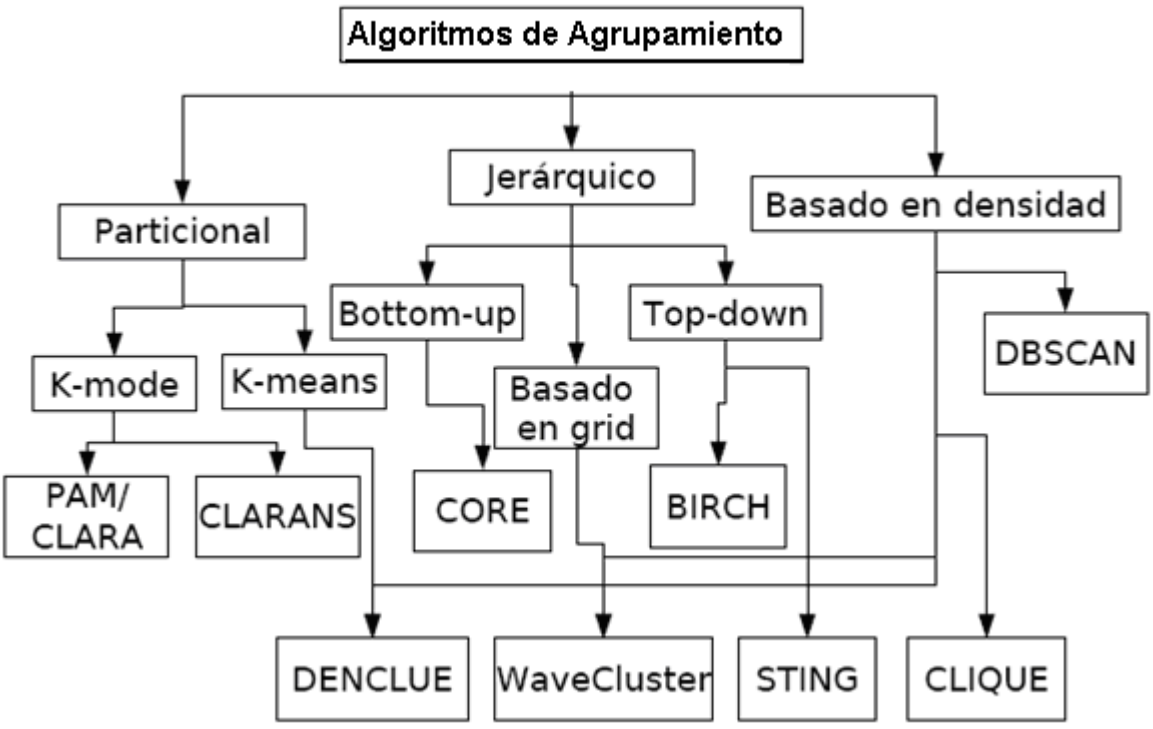


Figura 1.2: Algoritmos básicos de agrupamiento.

Agrupamiento jerárquico

Un método jerárquico crea una descomposición jerárquica de un conjunto de datos, formando un dendograma (árbol) que divide recursivamente el conjunto de datos en conjuntos cada vez más pequeños [3].

El árbol puede ser creado de dos formas: de abajo hacia arriba (bottom-up) o de arriba hacia abajo (top-down). En el caso bottom-up, también llamado aglomerativo, se comienza con cada objeto formando un grupo por separado. Los objetos o grupos se combinan sucesivamente según determinadas medidas, hasta que todos los grupos se hayan unido en uno solo, o hasta que se cumpla alguna condición de terminación.

En el caso top-down, también llamado divisivo, se comienza con todos los objetos en el mismo cluster, y a medida que se va iterando, se dividen los grupos en subconjuntos más pequeños según determinadas medidas, hasta que cada objeto se encuentre en un cluster individual o hasta que se cumplan las condiciones de terminación [3].

Algunos algoritmos de agrupamiento que pertenecen a esta clasificación son: CURE (Clustering Using Representatives), CHAMALEON, BIRCH (Balanced Iterative Reducing and Clustering using Hierarchical) y ROCK (RObust Clustering algorithm using linKs).

Agrupamiento particional

Un algoritmo de agrupamiento particional obtiene una partición simple de los datos en vez de la obtención de la estructura del grupo tal como se produce con los dendogramas de la técnica jerárquica.

El agrupamiento particional organiza los objetos dentro de k grupos de tal forma que sea minimizada la desviación total de cada objeto desde el centro de su grupo o desde una distribución de grupos. La desviación de un punto puede ser evaluada en forma diferente según el algoritmo, y es llamada generalmente función de similitud.

Los métodos particionales tienen ventajas en aplicaciones que involucran gran cantidad de datos para los cuales la construcción de un dendograma resultaría complicada.

El problema que se presenta al utilizar algoritmos particionales es la decisión del número deseado de grupos de salida. Las técnicas particionales usualmente producen grupos que optimizan el criterio de función definido local o globalmente. En la práctica, el algoritmo se ejecuta múltiples veces con diferentes estados de inicio y la mejor configuración que se obtenga es la que se utiliza como el agrupamiento de salida [12].

Algunos algoritmos de agrupamiento que pertenecen a esta clasificación son: CLARA (Clustering Large Applications), CLARANS (Clustering Large Applications based on Randomized Search), K-prototypes, K-mode, K-Means.

Agrupamiento basado en densidad

Los algoritmos basados en densidad obtienen grupos basados en regiones densas de objetos en el espacio de datos que están separados por regiones de baja densidad (estos elementos aislados representan ruido).

Este tipo de métodos es muy útil para filtrar ruido y encontrar grupos de diversas formas. La mayoría de los métodos de particionamiento, realizan el proceso de agrupamiento con base en la distancia entre dos objetos. Estos métodos pueden encontrar sólo grupos esféricos y se les dificulta hallar grupos de formas diversas.

Algunos algoritmos de agrupamiento que pertenecen a esta clasificación son: DBSCAN (Density-Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points To Identify the Clustering Structure) y DENCLUE (DENsity-based CLUstEring)

Agrupamiento basado en grid

Recientemente un número de algoritmos de agrupamiento han sido presentados para datos espaciales, estos son conocidos como algoritmos basados en grid. Estos algoritmos cuantifican el espacio en un número finito de celdas y aplican operaciones sobre dicho espacio. La mayor ventaja de este método es su veloz procesamiento del tiempo, el cual generalmente es independiente de la cantidad de objetos a procesar.

Algunos algoritmos de agrupamiento que pertenecen a esta clasificación son: STING (Statistical Information Grid-based method), CLIQUE y Wavecluster.

Aplicaciones de la Minería de datos.

La Minería de datos se utiliza actualmente tanto en empresas, con finalidades relacionadas con la gestión de negocios, como en organizaciones públicas, con objetivos que vayan en beneficio de la sociedad.

Entre otras aplicaciones, la Minería de datos se destaca por su uso en marketing e iniciativas de CRM (Customer Relationship Management), como, por ejemplo, en la identificación de clientes que eventualmente pudieran interesarse en una oferta específica. También se utiliza en técnicas de venta para realizar "cross selling" o para determinar el mejor mix de productos y su óptima distribución espacial en los locales de venta de las grandes tiendas [15].

En ese sentido, esta tecnología permite establecer perfiles muy bien definidos de los clientes así como registrar las tendencias en sus deserciones con la finalidad de prevenirlas y buscar su fidelización, a partir del conocimiento profundo de sus inclinaciones y necesidades.

La Minería de datos se conoce por su uso para el análisis de riesgo en operaciones financieras e inversiones, lo cual también puede hacerse con los clientes de un banco para decidir la aprobación o negación de un crédito o para determinar el monto apropiado para cada uno de ellos [15].

Esta aplicación es utilizada también en investigaciones científicas, por ejemplo, en la medicina, para analizar un tratamiento en el tiempo, o bien en meteorología, para predecir el estado del tiempo o los niveles de polución en ciudades o zonas geográficas [15].

Sin embargo, uno de las aplicaciones más reconocidas es la detección de fraudes. Gracias a esta tecnología analítica es posible detectar patrones de comportamientos anómalos de clientes de bancos, acciones inusuales en el pago de impuestos u otras operaciones similares, lo cual permite detectar a tiempo acciones fraudulentas.

Por último, en el ámbito de la manufactura y la industria, la Minería de datos permite eventuales fallas. En todos los casos mencionados, se trata de analizar un alto volumen de datos y obtener información que de cualquier otra manera sería imposible lograr debido al tiempo que tomaría su revisión completa [15].

La Minería de datos posee un gran ROI (Retorno sobre la Inversión) ya que ahorra mucho dinero a las empresas y les ayuda a descubrir nuevas oportunidades de negocios, mientras contribuye a que la alta dirección pueda tomar mejores decisiones, a partir de modelos descriptivos y predictivos que establecen relaciones no conocidas previamente y que ofrecen una visión global de la gestión de la organización, teniendo un impacto muy positivo en los resultados finales [9].

1.6 Herramientas Comerciales de Análisis de Datos

KnowledgeSeeker de Angoss Software International, Toronto, Canada

Herramienta interactiva de clasificación, basada en los algoritmos de árboles de decisión CHAID y XAID. Se ejecuta sobre plataformas Windows y UNIX. Realiza una representación flexible de árboles de decisión, provee características para permitir la identificación de la relevancia de los resultados en los negocios. El API permite usar los resultados del análisis en aplicaciones personalizadas.

Aspectos a tener en cuenta:
Esta herramienta sólo soporta árboles de decisión, tiene poco soporte para la transformación de datos y el soporte para predicción se limita a la exportación de las reglas generadas.

DataCruncher de DataMind, San Mateo, CA, USA

Herramienta de la Minería de datos para clasificación y agrupamiento, basada en Tecnología de agentes de redes (ANT Agent Network Technology). La aplicación servidor se ejecuta sobre UNIX y Windows NT; la aplicación cliente en todas las plataformas Windows. Es fácil de usar, ya que los modelos necesitan pocas adaptaciones. Agent Network Technology puede ser utilizada para clasificación, predicción y agrupamiento no supervisado. Permite obtener resultados versátiles, que permiten una minuciosa valoración de los modelos y de sus resultados.

Aspectos a tener en cuenta:
Para trabajar con esta herramienta se necesita familiarizarse con la tecnología para comprender los resultados. Además, está basada en una técnica propietaria y tiene soporte limitado para la transformación de datos.

Intelligent Miner de IBM, Armonk, NY, USA

Soporta múltiples operaciones de la Minería de datos en un entorno cliente-servidor, utiliza redes de neuronas, árboles de inducción y varias técnicas estadísticas. Trabaja sobre clientes Windows, OS/2 y X-Windows, y servidores AIX (incluyendo SP2), OS/400 y OS/390. Tiene buen soporte para análisis de asociaciones y agrupamiento (incluyendo visualización de agrupamiento), además de clasificación y predicción. Optimizada para la Minería de datos en grandes bases de datos (del orden de gigabytes) ya que se aprovecha de la plataforma de procesamiento paralelo PS2 de IBM y tiene un entorno de trabajo integrado con características muy interesantes tanto para usuarios expertos como no especialistas.

Aspectos a tener en cuenta:
Algunos problemas que tenía han sido resueltos con la nueva interfaz que ha sido desarrollada completamente en Java. Además, sólo trabaja sobre plataformas IBM, y el acceso a los datos se limita a las bases de datos DB2 y a ficheros planos por lo que inicialmente la mayoría de los proyectos requerirán entradas importantes desde los servicios de soporte y consultoría de IBM

Clamentine de Integral Solutions, Basingstoks, UK

Herramienta con un entorno de trabajo que soporta todo el proceso de Minería de datos. También ofrece árboles de decisión, redes de neuronas, generación de reglas de asociación y características de visualización y se ejecuta sobre VMS, UNIX o Windows NT. Presenta una interfaz gráfica intuitiva para programación visual, las técnicas de la Minería de datos pueden complementarse combinándose entre sí y presenta una visión interactiva de las relaciones entre las variables a través de grafos de red.

Aspectos a tener en cuenta:
No soporta Windows nativo, es necesario familiarizarse con la herramienta para conseguir una óptima utilización de sus funcionalidades y no está optimizada para arquitecturas en paralelo.

Alice de Isoft SA, Gif sur Yvette, Francia

Herramienta de escritorio para Minería de datos interactiva. Se basa en tecnología de árboles de decisión y se ejecuta sobre plataformas Windows. La representación altamente interactiva permite guiar el análisis y la opción de generar gráficos provee una visión general de los datos en todas las etapas del proceso de

Minería de datos. Se trata de una herramienta económica válida para usuarios que comienzan a realizar Minería de datos.

Aspectos a tener en cuenta:
No tiene opciones para desarrollar modelos, pequeño soporte para transformación de datos y no genera conjuntos de reglas optimizadas desde los árboles de decisión.

Decisión Series, de NeoVista Software Cupertino CA, USA.

Herramientas para múltiples operaciones de Minería de datos para el desarrollo de modelos basados en servidores, proporciona algoritmos de redes de neuronas, árboles y reglas de inducción, agrupamiento y análisis de asociaciones. Trabaja sobre sistemas UNIX mono o multi-procesadores de HP y Sun y accede sólo a ficheros planos, aunque posiblemente las últimas versiones ya trabajarán contra bases de datos relacionales. También soporta un gran rango de operaciones y algoritmos de Minería de datos, la mayoría de los cuales han sido altamente optimizados para obtener altos rendimientos.

Aspectos a tener en cuenta:
Las herramientas de desarrollo gráfico son bastante básicas y tiene poco soporte para la exploración de datos. La mayoría de los clientes necesitarán un considerable soporte de consultas para generar aplicaciones y ejecutarlas. Es necesario tener conocimientos de análisis de datos y de utilización de UNIX para desarrollar las aplicaciones.

Pilot Discovery Server de Pilot Software, Cambridge MA, USA

Herramienta para clasificación y predicción. Está basada en la tecnología de árboles de decisión CART y trabaja sobre UNIX y Windows NT. Presenta buena representación del análisis de resultados, es fácil de usar y de entender y está muy integrada con sistemas gestores de bases de datos relacionales.

Aspectos a tener en cuenta:
Solamente indicada para clientes de los programas para soporte a la toma de decisiones de Pilot. Solamente cubre un específico sector del espectro de la Minería de datos y sólo trabaja con datos almacenados en bases de datos relacionales.

SAS Solution for Data Mining de SAS Institute, Cary, NC, USA

Un gran número de herramientas de selección, exploración y análisis de datos para entornos cliente-servidor. Las opciones de Minería de datos incluyen: aplicaciones de redes de neuronas, de árboles de decisión y herramientas de estadística. Aplicaciones portables para un gran número de entornos PC, UNIX y mainframes.

Aspectos a tener en cuenta:
La oferta para hacer Minería de datos es una mezcolanza de todas las técnicas SAS existentes. Está integrada con la programación en 4GL y no soporta el análisis de asociaciones.

MineSet, de Silicon Graphics, Mountain View, CA, USA

Presenta un paquete de herramientas para Minería de datos y visualización. Además, proporciona algoritmos para la generación de reglas para clasificación y asociaciones y trabaja sobre plataformas SGI bajo IRIS. También ofrece herramientas de visualización para los datos y los modelos generados.

Aspectos a considerar:
Requiere un servidor SGI. Además, la gran cantidad de opciones y parámetros puede provocar confusión en usuarios noveles. Las herramientas de visualización necesitan mucha preparación y personalización de los datos para producir buenos resultados.

SPSS, de SPSS, Chicago IL, USA

Herramientas de escritorio para clasificación y predicción, agrupamiento, y un gran rango de operaciones estadísticas. Proporciona una herramienta de redes de neuronas además de productos de análisis estadístico. SPSS para Windows y Neural Connection son productos que trabajan en modo monopuesto en plataformas Windows. Las funciones de análisis estadístico complejo son accesibles a través de una interfaz de usuario muy bien diseñada.

Aspectos a considerar:
Está diseñada para analistas de datos y estadísticos, más que para usuarios finales, carece de la funcionalidad de otros productos de escritorio de árboles de decisión y se limita a la importación de 32.000 registros.

Syllogic Data Mining Tool, de Syllogic, Houten, The Netherlands

Herramienta con entorno de trabajo multi-estratégico con interfaz visual. Soporta análisis de árboles de decisión, clasificación k-vecino más próximo, y análisis de agrupamiento y asociaciones por k-means. Además, trabaja sobre Windows NT y en estaciones UNIX con uno o varios procesadores. La interfaz visual permite a los usuarios construir proyectos de Minería de datos enlazando objetos.

Aspectos a considerar:
La interfaz y la presentación de resultados necesitan algunos refinamientos para ser utilizada por usuarios finales. DMT/MP no soporta el mismo rango de operaciones que DMT.

Darwin de Thinking Machines, Bedford MA, USA

Herramientas de desarrollo de Minería de datos de tipo cliente-servidor para la construcción de modelos de clasificación y predicción. La construcción de modelos utiliza algoritmos de redes de neuronas, árboles de inducción y k-vecino más próximo. Trabaja sobre plataformas Sun de Solaris, AIX de IBM y SP2, con clientes Motif.

También existen versiones cliente que trabajan sobre Windows. Ofrecen buena cobertura al proceso completo de descubrimiento del conocimiento y pone el énfasis en el desarrollo de modelos predictivos de alto rendimiento.

Aspectos a considerar:
Mejor para analistas de datos y desarrolladores de aplicaciones que para los usuarios de negocio. Es preciso familiarizarse con las diferentes opciones de Darwin para cada tipo de modelo si se quiere obtener el mejor resultado de la herramienta y no soporta análisis no supervisado de agrupamiento o de asociaciones.

WEKA

Del inglés *Waikato Environment for Knowledge Analysis*. Este software es el que se utilizará en este trabajo para realizar las comparaciones de clasificación y agrupamiento. Se decidió emplearlo, porque contiene una extensa colección de algoritmos de aprendizaje por computadora o ML (del inglés M*achine Learning*) para realizar tareas de Minería de datos.

Los algoritmos pueden ser aplicados directamente a un conjunto de datos o *dataset* desde la interfaz gráfica del programa (Java Swing), mandándolos llamar desde el shell o utilizar los códigos independientes que se proporcionan mandándolos llamar desde nuestro propio programa en Java, utilizándolos de la forma en que indica la documentación. Es posible agregar códigos para extender la funcionalidad del paquete ya que proporcionan el código fuente completo.

Este software fue desarrollado en la Universidad de Waikato y es un paquete de fuente abierta bajo la licencia GPL [1]. Los autores de este paquete, son autores también del libro "*Data Mining: Practical Machine Learning Tools and Techniques*", en donde además de explicarse la mayoría de los algoritmos utilizados para realizar Minería de datos y cómo utilizarlos para conseguir la información que se desea, se explica cómo utilizar la implementación de Weka para conseguir los fines del usuario, tanto con la interfaz gráfica que proporciona el paquete, como a través de los códigos fuente [6].

Este paquete tiene una interfaz para la importación de datos que puede importar la información ya sea de archivos en su formato ARFF (**.arff,* del inglés *Attribute Relation File Format*), de instancias binarias serializadas, de archivos C45, de archivos separados por comas (**.csv*), de archivos separados por tabulaciones utilizando el conversor que ofrece, de una URL o de una base de datos PostgreSQL, MySQL u Oracle mediante su JDBC. Su última versión, la 3.4.10, es del 25 de enero de 2007 [6].

Weka es el proyecto de este tipo más antiguo (se inició alrededor del año 1993) y es de los más difundidos hasta la fecha. Este paquete se distribuye como un archivo comprimido, en donde se incluye la documentación, un tutorial, el *log* de cambios, el icono, los códigos fuente, entre otras muchas cosas. Para correr el programa se requiere tener el JRE (del inglés *Java Runtime Environment*) de Sun instalado, descomprimir el archivo (de forma que el archivo weka.jar sea accesible), y correr desde el shell *java -jar weka.jar* [6].

CAPÍTULO 2. Análisis de los algoritmos.

Introducción

Este capítulo se centrará en el análisis de métodos de aprendizaje automático supervisados y no supervisados. Este tipo de métodos trata de resolver un problema a partir de información extraída de un conjunto de ejemplos existentes previamente. Los métodos basados en vecindad reciben su nombre del hecho de que la predicción se basa fundamentalmente en la utilización del conjunto de ejemplos "vecinos" al dato que hay que procesar, o en el caso más general, porque la distancia entre cada ejemplo y el dato en cuestión es esencial en el proceso.

El capítulo se va a dividir en métodos para la tarea de clasificación, en la que los ejemplos tienen información de la clase a la que pertenecen (aprendizaje supervisado), y métodos para la tarea de agrupamiento, en la que los ejemplos no contienen información de la clase a la que pertenecen (aprendizaje no supervisado). El capítulo incluye técnicas diversas: vecinos más próximos, Naive Bayes y K* para clasificación y agrupamiento, K medias y COBWEB. La característica conductora de todas estas técnicas es que se basan en los conceptos de distancia o en conceptos derivados de ella, como es el de cercanía o vecindad.

Se describe además, la implementación de estos algoritmos en la herramienta WEKA, seleccionada para realizar la ejecución de los mismos.

2.1 Distancia

En la tarea de clasificación, podremos asignar una clase a un nuevo caso, observando las clases de casos similares. Igualmente, en la tarea de agrupamiento, asignaremos un nuevo ejemplo al grupo donde estén los individuos más similares.

A partir de esta idea surgen dos nociones muy importantes. La primera, y la más obvia, es determinar qué se entiende por "similitud", dando lugar al concepto matemático equivalente (o más bien, inverso) de "distancia". La segunda es determinar cuándo se va a explotar dicha similitud: si se va a realizar con un preprocesamiento anticipativo o no retardado *(eager)* respecto a un preprocesamiento demorado o retardado *(lazy) [9].*

A continuación se describe en más detalle el concepto de distancia como función inversa de la similitud y la separación entre métodos de aprendizaje supervisado y no supervisado.

Como se ha mencionado, la manera de formalizar el concepto de similitud es a través de métricas o medidas de distancia. En realidad, si se quiere saber la similitud entre dos instancias o individuos, es necesario elegir una función de distancia y calcular con ella la distancia entre los dos individuos. Y éste es precisamente uno de los aspectos más interesantes de los métodos que se describen en este capítulo: el mismo método puede funcionar de maneras diferentes, variando la métrica de distancia. Dicho de otra manera, se pueden adaptar distintos métodos de aprendizaje que tienen una función de distancia como parámetro para trabajar con distintos tipos de problemas [9].

Una pregunta lógica ante esta situación es la siguiente: ¿cuántas funciones de distancia hay? Como se verá a continuación, además de la distancia Euclidiana ó Clásica, existen otras muchas funciones que cumplen los requisitos de una función de distancia (llamada entonces métrica) y que pueden funcionar mejor en ciertos contextos. Más aún, dependiendo de la aplicación, se pueden definir funciones *ad-hoc* que actúen como métrica de distancia.

Las medidas de distancias más tradicionales son aquellas que se aplican sobre dos instancias o ejemplos, tales que todos los atributos son numéricos. Por ejemplo, se pueden definir las siguientes distancias entre dos vectores, puntos o instancias *x* e *y* de dimensión *n* de muy distintas formas.

Distancia Euclidiana. [9]Es la distancia clásica, como la longitud de la recta que une dos puntos en el espacio euclídeo:

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$ Ec. 1

Distancia de Manhattan o distancia por cuadras *(city-block).* [9]Como su nombre indica, hace referencia a recorrer un camino no en diagonal (por el camino más corto), sino zigzagueando, como se haría en Manhattan:

$$d(x,y)=\sum_{i=1}^{n}|x_i-y_i|$$ Ec. 2

Distancia de Chebychev. [9]Simplemente calcula la discrepancia más grande en alguna de las dimensiones:

$$d(x,y) = \max_{i=1..n} |x_i - y_i|$$ Ec.3

Distancia del coseno. [9]Si se considera que cada ejemplo es un vector, la distancia sería el coseno del ángulo que forman:

$$d(x,y) = \arccos\left(\frac{x^T y}{\|x\| \cdot \|y\|}\right)$$ Ec. 4

Distancia de Mahalanobis. [9]Todas las distancias anteriores asumen, en cierto modo, que los atributos son independientes (es decir, consideran cada atributo una dimensión ortogonal a las demás). Una distancia más robusta es ésta, que utiliza la matriz de covarianzas S:

$$d(x,y) = \sqrt{(x-y)^T S^{-1} (x-y)}$$ Ec. 5

El concepto de distancia no sólo existe para valores numéricos, también se puede utilizar cuando los ejemplos están formados por atributos nominales, cosa que es habitual en minería de datos. Por ejemplo, para atributos nominales se suele utilizar la función delta, es decir *8(a, b)=0* si y sólo si *a=b* y *8(a, b)=1* en caso contrario. Con esta función, podemos definir la distancia de manera sencilla:

$$d(x,y) = \omega \sum_{i=1}^{n} \delta(x_i, y_i)$$ Ec. 6

En la ecuación anterior ω es simplemente un factor de reducción (por ejemplo l/n es un valor usual). Este factor se puede elegir convenientemente cuando los atributos son nominales y numéricos, se puede utilizar esta función de distancia para el subconjunto de atributos nominales, utilizar una función de distancia de las anteriores para el subconjunto de atributos numéricos y combinar ambos valores utilizando un factor adecuado.

Del mismo modo se pueden definir distancias para otros tipos de datos más complejos. Por ejemplo, cuando se trabaja con tiras de caracteres, una distancia muy habitual es la *distancia de edición,* en la que se ponderan inserciones, borrados y sustituciones. Por esta razón la palabra "jamón" está más cerca de "amor" que de "monja".

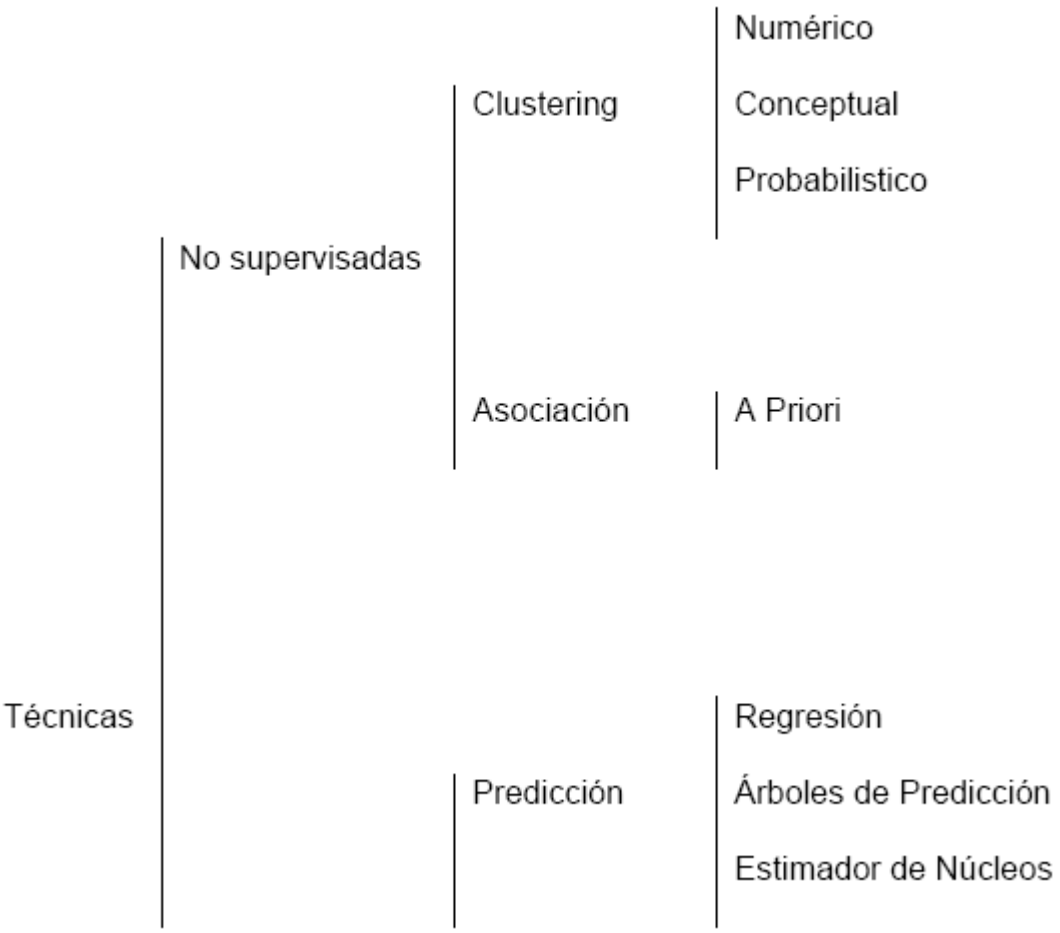
Si los datos son conjuntos más que vectores, entonces se puede definir la distancia entre conjuntos de la siguiente manera:

$$d(x,y) = \frac{|x \cup y| - |x \cap y|}{|x \cup y|}$$ Ec. 7

Finalmente, se pueden definir distancias específicas para documentos de texto o hipertexto, grafos, árboles y cualquier otra estructura de datos que represente los ejemplos. La relevancia de todas estas medidas de distancia es que la mayoría de métodos que se describen en este capítulo se pueden "parametrizar" para trabajar con distintos tipos de datos y actuando de diferentes formas, con sólo cambiar la función de distancia que utiliza el algoritmo.[9]

2.2 Técnicas Supervisadas y Técnicas no supervisadas

Las técnicas de Minería de Datos se clasifican en dos grandes categorías: supervisadas o predictivas y no supervisadas o descriptivas.


Técnicas
No supervisadas
Clustering
Numérico
Conceptual
Probabilistico
Asociación
A Priori
Predicción
Regresión
Árboles de Predicción
Estimador de Núcleos

2.3 Técnica supervisada

En el aprendizaje inductivo supervisado existe un atributo especial, normalmente denominado *clase,* presente en todos los ejemplos que especifica si el ejemplo pertenece o no a un cierto concepto, que será el objetivo del aprendizaje. El atributo clase normalmente toma los valores + y -, que significan la pertenencia o no del ejemplo al concepto que se trata de aprender; es decir, que el ejemplo ejemplifica positivamente al concepto -pertenece al concepto- o bien lo ejemplifica negativamente -que no pertenece al concepto. Mediante una generalización del papel del atributo clase, cualquier atributo puede desempeñar ese papel, convirtiéndose la *clasificación* de los ejemplos según los valores del atributo en cuestión, en el objeto del aprendizaje. Expresado en una forma breve, el objetivo del aprendizaje supervisado es: a partir de un conjunto de ejemplos, denominados de entrenamiento, de un cierto dominio *D* de ellos, construir criterios para determinar el valor del atributo clase en un ejemplo cualquiera del dominio. Esos criterios están basados en los valores de uno o varios de los otros pares (atributo; valor) que intervienen en la definición de los ejemplos. Es sencillo transmitir esa idea al caso en el que el atributo que juega el papel de la clase sea uno cualquiera o con más de dos valores. Dentro de este tipo de aprendizaje se pueden distinguir dos grandes grupos de técnicas: la predicción y la clasificación.

La extracción y recuperación de la información de manera supervisada se puede realizar utilizando los siguientes métodos:

- Árboles de Decisión: Un árbol de decisión es un modelo de predicción utilizado en el ámbito de la inteligencia artificial, dada una base de datos se construyen estos diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que suceden de forma sucesiva, para la resolución de un problema [12].

- Naive Bayes: El método de Naive Bayes es una de las técnicas más populares para clasificar texto. Bayes es un clasificador empleado para representar distribuciones conjuntas de modo que permitan calcular la probabilidad a posteriori de un conjunto de clases dado un conjunto de características obtenidas de la información, y así clasificar los datos en la clase mas probable [12].

- Modelos Lineales de redes neuronales: El método de aprendizaje conocido como Redes Neuronales fue desarrollado simultáneamente en los ámbitos del análisis estadístico y de la inteligencia artificial. La idea central consiste en extraer combinaciones lineales de los atributos presentes en los objetos obteniendo una serie de características y modelizar las clases como funciones no lineales de dichas características.

- El Perceptron Simple y Perceptron Multicapa: son dos tipos de redes neuronales con aprendizaje de manera supervisada. Las redes de Neuronas son unos excelentes clasificadores y está demostrado que clasifican mejor que otros clasificadores como: LVQ, árboles de decisión, vecinos próximos, etc. [12].

- Modelos ocultos de Markov: Un modelo oculto de Markov (HMM en inglés) es un modelo estadístico en el que se asume que el sistema a modelar es un proceso de Markov de parámetros desconocidos. El objetivo es determinar los parámetros ocultos a partir de los parámetros observables. Los parámetros extraídos se pueden emplear para llevar a cabo sucesivos análisis, por ejemplo en aplicaciones de reconocimiento de formas. Un HMM se puede considerar como la red bayesiana dinámica más simple [12].

- Redes de Cuantización Vectorial (LVQ): Una propiedad importante de estas redes, es que el proceso de combinar subclases para formar clases, permite a la red LVQ crear clases más complejas. Una capa competitiva estándar tiene la limitación de que puede crear sólo regiones de decisión convexas; la red LVQ soluciona esta limitación. La red LVQ combina aprendizaje competitivo con aprendizaje supervisado, razón por lo cual necesita un set de entrenamiento que describa el comportamiento propio de la red.

- Vecinos próximos (IB1-IBK): Consiste en métodos de aprendizaje basados en ejemplos (instance-based learning) los ejemplos de entrenamiento se almacenan tal cual se usa una función de distancia para determinar que patrón del conjunto de entrenamiento está más cerca del patrón a clasificar.

A continuación se exponen los algoritmos de clasificación más conocidos que se seleccionaron para realizar la comparación.

2.3.1 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 8.

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)} \quad \text{Ec. 8}$$

Para estimar la hipótesis más probable (MAP, [maximum a posteriori hipótesis]) se busca el mayor *P(h|D)* como se muestra en la ecuación 9.

$$\begin{aligned} h_{MAP} &= argmax_{h \in H}\left(P(h \mid D)\right) \\ &= argmax_{h \in H}\left(\frac{P(D \mid h)P(h)}{P(D)}\right) \\ &= argmax_{h \in H}\left(P(D \mid h)P(h)\right) \end{aligned} \quad \text{Ec. 9}$$

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, [maximum likelihood]) de la ecuación 10.

$$h_{ML} = argmax_{h \in H}\left(P(D \mid h)\right)$$ Ec.10

El clasificador Bayesiano se utiliza cuando se quiere clasificar un ejemplo descrito por un conjunto de atributos (a_i'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 la ecuación 6 al problema de la clasificación se obtendrá la ecuación 11.

$$\begin{aligned} v_{MAP} &= argmax_{v_j \in V}\left(P(v_j \mid a_1,...,a_n)\right) \\ &= argmax_{v_j \in V}\left(\frac{P(a_1,...,a_n \mid v_j)P(v_j)}{P(a_1,...,a_n)}\right) \\ &= argmax_{v_j \in V}\left(P(a_1,...,a_n \mid v_j)P(v_j)\right) \end{aligned}$$ Ec.11

Además, el clasificador Bayesiano asume que los valores de los atributos son condicionalmente independientes dado el valor de la clase. Los clasificadores 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”. Esta 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 Bayes 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.

2.3.2 Algoritmo de los k-vecinos más próximos

El método de los *k*-vecinos más próximos (KNN, [k-Nearest Neighbor]) está considerado como un buen representante de este tipo de aprendizaje, y es de gran sencillez conceptual. Se suele denominar método porque es el esqueleto de un algoritmo que admite el intercambio de la función de proximidad dando lugar a múltiples variantes. La función de proximidad puede decidir la clasificación de un nuevo ejemplo atendiendo a la clasificación del ejemplo o de la mayoría de los *k* ejemplos más cercanos. Admite también funciones de proximidad que consideren el peso o coste de los atributos que intervienen, lo que permite, entre otras cosas, eliminar los atributos irrelevantes. Una función de proximidad clásica entre dos instancias x_i y x_j, si suponemos que un ejemplo viene representado por una *n* tupla de la forma ($a_1(x)$,

$a_2(x),..., a_n(x)$) en la que $a_r(x)$ es el valor de la instancia para el atributo a_r, es la distancia euclídea, que se muestra en la ecuación 12.

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^{n} (x_{il} - x_{jl})^2}$$ Ec.12

COMIENZO
Entrada: $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)\}$
$\mathbf{x} = (x_1, \ldots, x_n)$ nuevo caso a clasificar
PARA todo objeto ya clasificado (x_i, c_i)
calcular $d_i = d(\mathbf{x}_i, \mathbf{x})$
Ordenar $d_i (i = 1, \ldots, N)$ en orden ascendente
Quedarnos con los K casos $D_{\mathbf{x}}^K$ ya clasificados más cercanos a $\mathbf{x}$
Asignar a $\mathbf{x}$ la clase más frecuente en $D_{\mathbf{x}}^K$
FIN

Figura 2.1: Pseudocódigo del algoritmo KNN.

Además de los distintos tipos de atributos hay que tener en cuenta también, en el caso de los atributos numéricos, los rangos en los que se mueven sus valores. Para evitar que atributos con valores muy altos tengan mucho mayor peso que atributos con valores bajos, se normalizarán dichos valores con la ecuación 13.

$$\frac{x_{il} - min_l}{Max_l - min_l}$$ Ec.13

2.3.3 Algoritmo k-estrella

El algoritmo K* es una técnica de Minería de datos basada en ejemplares en la que la medida de la distancia entre ejemplares se basa en la teoría de la información. Una forma intuitiva de verlo es que la distancia entre dos ejemplares se define como la complejidad de transformar un ejemplar en el otro. El cálculo de la complejidad se basa en primer lugar en definir un conjunto de transformaciones $T = \{t_1, t_2, ..., t_n, \sigma\}$ para pasar de un ejemplo (valor de atributo) *a* a uno *b*. La transformación σ es la de parada y es la transformación identidad ($\sigma(a)=a$). El conjunto *P* es el conjunto de todas las posibles secuencias de transformaciones descritos en *T** que terminan en σ, y t(a) es una de estas secuencias concretas sobre el

ejemplo *a*. Esta secuencia de transformaciones tendrá una probabilidad determinada p(t), definiéndose la función de probabilidad *P*(b|a)* como la probabilidad de pasar del ejemplo a al ejemplo b a través de cualquier secuencia de transformaciones, tal y como se muestra en la ecuación 14.

$$P^*(b \mid a) = \sum_{\bar{t} \in P : \bar{t}(a) = b} p(\bar{t})$$ Ec.14.

Esta función de probabilidad cumplirá las propiedades que se muestran en 15.

$$\sum_b P^*(b \mid a) = 1; \quad 0 \le P^*(b \mid a) \le 1$$ Ec.15.

La función de distancia *K** se define entonces tomando logaritmos, tal y como se muestra en la ecuación 16.

$$K^*(b \mid a) = -\log_2 P^*(b \mid a)$$ Ec.16.

2.4 Técnica no supervisada

El aprendizaje inductivo no supervisado estudia el aprendizaje sin la ayuda del *maestro*; es decir, se aborda el aprendizaje sin supervisión, que trata de ordenar los ejemplos en una jerarquía según las regularidades en la distribución de los pares atributo-valor sin la *guía* del atributo especial *clase*. Este es el proceder de los sistemas que realizan *agrupamiento* conceptual y de los que se dice también que adquieren nuevos conceptos. Otra posibilidad contemplada para estos sistemas es la de sintetizar conocimiento cualitativo o cuantitativo, objetivo de los sistemas que llevan a cabo tareas de descubrimiento [11].

Este tipo de clasificación también se le denomina auto-adaptativa ya que se va modificando de forma que se adecua al conjunto de datos de entrada del que parte. Los modos de esta clasificación son búsqueda de:

• Nivel de semejanza entre conceptos buscando la cercanía de los mismos. Aplicada a la recuperación de información significa buscar comparación entre los conceptos para estructurar la propia información.

• Extracción de características y sus relaciones, indagar en las características que existen en los conceptos y sus relaciones. Dichos conceptos parecidos deberían dar los mismos resultados.

• Analizar y extraer principales características, escoger las peculiaridades necesarias y fundamentales que nos den la suficiente información, para la recuperación es importante ya que se restringe los datos a tener en cuenta para realizar la recuperación.

• Modelización, tratar de agrupar y buscar el prototipo que represente a todo un grupo.
A continuación se exponen los algoritmos de agrupamiento más conocidos que se seleccionaron para realizar la comparación.

2.4.1 Agrupamiento Numérico (k-medias)

Uno de los algoritmos más utilizados para hacer agrupamiento es el *k*-medias (kmeans), que se caracteriza por su sencillez. En primer lugar se debe especificar por adelantado cuantos grupos se van a crear, este es el parámetro k, para lo cual se seleccionan *k* elementos aleatoriamente, que representarán el centro o media de cada grupo. A continuación cada una de las instancias, ejemplos, es asignada al centro del grupo más cercano de acuerdo con la distancia Euclidiana que le separa de él. Para cada uno de los grupos así construidos se calcula el centroide de todas sus instancias. Estos centroides son tomados como los nuevos centros de sus respectivos grupos. Finalmente se repite el proceso completo con los nuevos centros de los grupos. La iteración continúa hasta que se repite la asignación de los mismos ejemplos a los mismos grupos, ya que los puntos centrales de los grupos se han estabilizado y permanecerán invariables después de cada iteración [5]. El algoritmo de *k*-medias es el siguiente:

```
1.  Elegir k ejemplos que actúan como semillas (k número de
    clusters).

2.  Para cada ejemplo, añadir ejemplo a la clase más similar.

3.  Calcular el centroide de cada clase, que pasan a ser las nuevas
    semillas

4.  Si no se llega a un criterio de convergencia (por ejemplo, dos
    iteraciones no cambian las clasificaciones de los ejemplos), volver
    a 2.
```

Figura 2.2: Pseudocódigo el algoritmo de k-medias.

Para obtener los centroides, se calcula la media (mean) o la moda (mode) según se trate de atributos numéricos o simbólicos.

2.4.2 Agrupamiento Conceptual (COBWEB)

El algoritmo de *k*-medias se encuentra con un problema cuando los atributos no son numéricos, ya que en ese caso la distancia entre ejemplares no está tan clara. Para resolver este problema Michalski presenta la noción de agrupamiento conceptual, que utiliza para justificar la necesidad de un agrupamiento cualitativo frente al agrupamiento cuantitativo, basado en la vecindad entre los elementos de la población.

En este tipo de agrupamiento una partición de los datos es buena si cada clase tiene una buena interpretación conceptual (modelo cognitivo de jerarquías). Una de las principales motivaciones de la categorización de un conjunto de ejemplos, que básicamente supone la formación de conceptos, es la predicción de características de las categorías que heredarán sus subcategorías. Esta conjetura es la base de COBWEB. A semejanza de los humanos, COBWEB forma los conceptos por agrupación de ejemplos con atributos similares. Representa los grupos como una distribución de probabilidad sobre el espacio de los valores de los atributos, generando un árbol de clasificación jerárquica en el que los nodos intermedios definen subconceptos. El objetivo de COBWEB es hallar un conjunto de clases o grupos (subconjuntos de ejemplos) que maximice la utilidad de la categoría (partición del conjunto de ejemplos cuyos miembros son clases) [12]. La descripción probabilística se basa en dos conceptos:

• Predicibilidad: Probabilidad condicional de que un suceso tenga un cierto atributo dada la clase, $P(A_i=V_{ij}|C_k)$. El mayor de estos valores corresponde al valor del atributo más predecible y es el de los miembros de la clase (alta similaridad entre los elementos de la clase).

• Previsibilidad: Probabilidad condicional de que un ejemplo sea una instancia de una cierta clase, dado el valor de un atributo particular, $P(C_k|A_i=V_{ij})$. Un valor alto indica que pocos ejemplos de las otras clases comparten este valor del atributo, y el valor del atributo de mayor probabilidad es el de los miembros de la clase (baja similaridad interclase).

Estas dos medidas, combinadas mediante el teorema de Bayes, proporcionan una función que evalúa la utilidad de una categoría (CU), que se muestra en la ecuación siguiente:

$$CU = \frac{\sum_{k=1}^{n} P(C_k)\left[\sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2\right]}{n}$$

Ec. 17

En esta ecuación *n* es el número de clases y las sumas se extienden a todos los atributos A_i y sus valores V_{ij} en cada una de las *n* clases C_k. La división por *n* sirve para incentivar tener grupos con más de un elemento. La utilidad de la categoría mide el valor esperado de valores de atributos que pueden ser adivinados a partir de la partición sobre los valores que se pueden adivinar sin esa partición. Si la partición no ayuda en esto, entonces no es una buena partición. El árbol resultante de este algoritmo cabe denominarse organización probabilística o jerárquica de conceptos.

El algoritmo se puede extender a valores numéricos usando distribuciones gaussianas, ecuación 18. De esta forma, la sumatoria de probabilidades es ahora como se muestra en la ecuación 19.

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

Ec. 18

$$\sum_j P(A_i = V_{ij})^2 \leftrightarrow \int_{-\infty}^{+\infty} f(x_i)^2 \, dx_i = \frac{1}{2\sqrt{\pi}\sigma_i}$$ Ec. 19

Por lo que la ecuación de la utilidad de la categoría quedaría como se muestra en la ecuación 20.

$$CU = \frac{1}{k} \sum_{k=1}^{n} P(C_k) \frac{1}{2\sqrt{\pi}} \sum_i \left(\frac{1}{\sigma_{ik}} - \frac{1}{\sigma_i} \right)$$ Ec. 20

1. Nuevo Ejemplo: Lee un ejemplo *e*. Si no hay más ejemplos, terminar.

2. Actualiza raíz. Actualiza el *cálculo* de la raíz.

3. Si la raíz es hoja, entonces: Expandir en dos nodos hijos y acomodar en cada uno de ellos un ejemplo; volver a 1.

4. Avanzar hasta el siguiente nivel: Aplicar la función de evaluación a varias opciones para determinar, mediante la fórmula de utilidad de una categoría, el *mejor* (máxima *CU*) lugar donde incorporar el ejemplo en el nivel siguiente de la jerarquía. En las opciones que se evaluarán se considerará únicamente el nodo actual y sus hijos y se elegirá la *mejor* opción de las siguientes:

   a. Añadir *e* a un nodo que existe (al mejor hijo) y, si esta opción resulta ganadora, comenzar de nuevo el proceso de avance hacia el siguiente nivel en ese nodo hijo.

   b. Crear un nuevo nodo conteniendo únicamente a *e* y, si esta opción resulta ganadora, volver a 1.

   c. Juntar los dos *mejores* nodos hijos con *e* incorporado al nuevo nodo combinado y, si esta opción resulta ganadora, comenzar el nuevo proceso de avanzar hacia el siguiente nivel en ese nuevo nodo.

   d. Dividir el *mejor* nodo, reemplazando este nodo con sus hijos y, si esta opción resulta ganadora, aplicar la función de evaluación para incorporar *e* en los nodos originados por la división.

Figura 2.3: Pseudocódigo del algoritmo de COBWEB.

2.5 Implementación de los algoritmos en WEKA

A continuación se describe la implementación en la herramienta WEKA, de los algoritmos de clasificación y agrupamiento analizados anteriormente.

2.5.1 Naive Bayes en WEKA

El algoritmo *Naive* Bayes se encuentra implementado en la clase *weka.classifiers.NaiveBayesSimple.java*. No dispone de ninguna opción de configuración. El algoritmo que implementa esta clase se corresponde completamente con el expuesto anteriormente. En este caso no se usa el *estimador de Laplace*, sino que la aplicación muestra un error si hay menos de dos ejemplos de entrenamiento para una terna *atributo-valor-clase* o si la desviación típica de un atributo numérico es igual a *0*[12].

Una alternativa a esta clase que también implementa un clasificador Bayesiano es la clase *weka.classifiers.NaiveBayes.java*. Las opciones de configuración de que disponen son las mostradas en la tabla 2.1.

Opción	Descripción
useKernelEstimator(False)	Emplear un estimador de densidad de núcleo para modelar los atributos numéricos en lugar de una distribución normal.

Tabla 2.1: Opciones de configuración para el algoritmo Bayes naive en WEKA.

En este caso, sin embargo, en lugar de emplear la frecuencia de aparición como base para obtener las probabilidades se emplean distribuciones de probabilidad. Para los atributos discretos o simbólicos se emplean estimadores discretos, mientras que para los atributos numéricos se emplean bien un estimador basado en la distribución normal o bien un estimador de densidad de núcleo.

Se creará una distribución para cada clase, y una distribución para cada atributo-clase, que será discreta en el caso de que el atributo sea discreto. El estimador se basará en una distribución normal o kernel en el caso de los atributo-clase con atributo numérico según se active o no la opción mostrada en la tabla 1.

En el caso de los atributos numéricos, en primer lugar se obtiene la precisión de los rangos, que por defecto en la implementación será de 0,01[12].

2.5.2 KNN en WEKA (IBk)

En WEKA se implementa el clasificador KNN con el nombre IBk, concretamente en la clase *weka.classifiers.IBk.java*. Además, en la clase *weka.classifiers.IB1.java* hay una versión simplificada del mismo, concretamente un clasificador NN (Nearest Neighbor), sin ningún tipo de opción, en el que, como su propio nombre indica, tiene en cuenta únicamente el voto del vecino más cercano. Por ello, en la tabla 2.2 se muestran las opciones que se permiten con el clasificador IBk [12].

Opción	Descripción
KNN (1)	Número de vecinos más cercanos.
distanceWeighting (No distance weighting)	Los valores posibles son: *No distance weighting*, *Weight by 1-distance* y *Weight by 1/distance*. Permite definir si se deben "pesar" los vecinos a la hora de votar bien según su semejanza o con la inversa de su distancia con respecto al ejemplo a clasificar.
crossValidate (False)	Si se activa esta opción, cuando se vaya a clasificar una instancia se selecciona el número de vecinos (hasta el número especificado en la opción KNN) mediante el proceso *hold-one-out*.
meanSquared (False)	Minimiza el error cuadrático en lugar del error absoluto para el caso de clases numéricas cuando se activa la opción *crossValidate*.
windowSize (0)	Si es *0* el número de ejemplos de entrenamiento es ilimitado. Si es mayor que *0*, únicamente se almacenan los *n* últimos ejemplos de entrenamiento, siendo *n* el número que se ha especificado.

debug (False)	Muestra el proceso de construcción del clasificador.
noNormalization (False)	No normaliza los atributos.

Tabla 2.2: Opciones de configuración para el algoritmo KNN en WEKA.

El algoritmo implementado en la herramienta WEKA consiste en crear el clasificador a partir de los ejemplos de entrenamiento, simplemente almacenando todas las instancias disponibles (a menos que se restrinja con la opción *windowSize*). Posteriormente, se clasificarán los ejemplos de test a partir del clasificador generado, bien con el número de vecinos especificados o comprobando el mejor *k* si se activa la opción *crossValidate*. En cuanto a los tipos de datos permitidos y las propiedades de la implementación, estos son:

• Admite atributos numéricos y simbólicos.
• Admite clase numérica y simbólica. Si la clase es numérica se calculará la media de los valores de la clase para los *k* vecinos más cercanos.
• Permite dar peso a cada ejemplo.
• El proceso de *hold-one-out* consiste en, para cada *k* entre *1* y el valor configurado en KNN (ver tabla 2.2), calcular el error en la clasificación de los ejemplos de entrenamiento. Se escoge el *k* con un menor error obtenido. El error cometido para cada *k* se calcula como el error medio absoluto o el cuadrático (ver tabla 2.2) si se trata de una clase numérica.

El cálculo de estos dos errores se puede ver en las ecuaciones 21 y 22 respectivamente. Si la clase es simbólica se tomará como error el número de ejemplos fallados entre el número total de ejemplos [12].

$$MAE = \frac{\sum_{i=1}^{m} \left| y_i - \hat{y}_i \right|}{m} \quad \text{Ec. 21}$$

$$MSE = \frac{\sum_{i=1}^{m} \left(y_i - \hat{y}_i \right)^2}{m} \quad \text{Ec. 22}$$

En las ecuaciones 21 y 22 y_i es el valor de la clase para el ejemplo *i* e *i* $\hat{y}$ es el valor predicho por el modelo para el ejemplo *i*. El número *m* será el número de ejemplos.

2.5.3 K* en WEKA

La clase en la que se implementa el algoritmo K* en la herramienta WEKA es weka.classifers.kstar.KStar.java. Las opciones que permite este algoritmo son las que se muestran en la tabla 2.3.

Opción	Descripción
entropicAutoBlend (False)	Si se activa esta opción se calcula el valor de los parámetros x_0 (o *s*) basándose en la entropía en lugar del parámetro de mezclado.
globalBlend (20)	Parámetro de mezclado, expresado en tanto por ciento.
missingMode (Average column entropy curves)	Define cómo se tratan los valores desconocidos en los ejemplos de entrenamiento: las opciones posibles son *Ignore the Instance with missing value* (no se tienen en cuenta los ejemplos con atributos desconocidos), *treta missing value as maximally different* (diferencia igual al del vecino más lejano considerado), *Normilize over the attributes* (se ignora el atributo desconocido) y *Average column entropy curves*.

Tabla 2.3: Opciones de configuración para el algoritmo K* en WEKA.

Dado que los autores de la implementación de este algoritmo en WEKA son los autores del propio algoritmo, dicha implementación se corresponde perfectamente con lo visto anteriormente. Simplemente son destacables los siguientes puntos:

• Admite atributos numéricos y simbólicos, así como pesos por cada instancia.

• Permite que la clase sea simbólica o numérica. En el caso de que se trate de una clase numérica se empleará la ecuación 23 para predecir el valor de un ejemplo de *test* [12].

$$v(a) = \frac{\sum_{i=1}^{n} P^*(b \mid a) * v(b)}{\sum_{i=1}^{n} P^*(b \mid a)} \qquad \text{Ec. 23}$$

En la ecuación 23 *v(i)* es el valor (numérico) de la clase para el ejemplo *i*, *n* el número de ejemplos de entrenamiento, y *P*(i|j)* la probabilidad de transformación del ejemplo *j* en el ejemplo *i*.

• Proporciona cuatro modos de actuación frente a pérdidas en los atributos en ejemplos de entrenamiento.

• Para el cálculo de los parámetros x_0 y *s* permite basarse en el parámetro *b* o en el cálculo de la entropía.

2.5.4 K-means en WEKA

El algoritmo de *k*-medias se encuentra implementado en la clase *weka.clusterers.SimpleKMeans.java*. Las opciones de configuración de que disponen son las que vemos en la tabla 2.4.

Opción	Descripción
numGrupos (2)	Número de grupos.
seed (10)	Semilla a partir de la cuál se genera el número aleatorio para inicializar los centros de los grupos.

Tabla 2.4: Opciones de configuración para el algoritmo k-medias en WEKA.

El algoritmo es exactamente el mismo que el descrito anteriormente. A continuación se enumeran los tipos de datos que admite y las propiedades de la implementación:

• Admite atributos simbólicos y numéricos.

• Para obtener los centroides iniciales se emplea un número aleatorio obtenido a partir de la semilla empleada. Los *k* ejemplos correspondientes a los *k* números enteros siguientes al número aleatorio obtenido serán los que conformen dichos centroides.

• En cuanto a la medida de similaridad, se emplea el mismo algoritmo que el que veíamos en el algoritmo KNN anteriormente.
• No se estandarizan los argumentos, sino que se normalizan (ecuación 24).

$$\frac{x_{il} - min_l}{Max_l - min_l}$$ Ec.24

En la ecuación 24, x_{if} será el valor *i* del atributo *f*, siendo min_f el mínimo valor del atributo *f* y Max_f el máximo [12].

2.5.5 COBWEB en WEKA

El algoritmo de COBWEB se encuentra implementado en la clase *weka.clusterers.Cobweb.java*. Las opciones de configuración de que disponen son las que vemos en la tabla 2.5.

Opción	Descripción
Acuity (100)	Indica la mínima varianza permitida en un cluster.
cutoff (0)	Factor de poda. Indica la mejora en utilidad mínima por una subdivisión para que se permita llevar a cabo.

Tabla 2.5: Opciones de configuración para el algoritmo COBWEB en WEKA.

La implementación de COBWEB en WEKA es similar al algoritmo explicado anteriormente. Algunas características de esta implementación son:

• Se permiten atributos numéricos y simbólicos.
• La semilla para obtener números aleatorios es fija e igual a *42*.
• Permite pesos asociados a cada ejemplo.
• Realmente el valor de cutoff es $0.01 \times 1 \ (2\sqrt{\pi})$.
• En el caso de que el ejemplo que se desea clasificar genere, en un nodo determinado, un *CU* menor al *cutoff*, se eliminan los hijos del nodo (poda) [12].

En este capítulo se describió el concepto de distancia y se analizaron algunas de las funciones de distancias tradicionales. También se abordaron los conceptos de las técnicas de aprendizaje supervisado y no supervisado, así como el análisis de los algoritmos de clasificación y agrupamiento más utilizados en cada una de ellas. Además, se abordó la implementación de los mismos en la herramienta WEKA.

CAPÍTULO 3. Evaluación y comparación de los algoritmos.

Introducción

Este capítulo se centrará en la evaluación de los algoritmos de clasificación y agrupamiento que se seleccionaron en el capítulo anterior. También se realizarán las comparaciones de los algoritmos de cada grupo lo que permitirá llegar a conclusiones acerca de su eficiencia. En los primeros epígrafes del capítulo se analiza el caso de estudio seleccionado para realizar las comparaciones, y se describe el proceso de preparación de los datos que se utilizarán para ejecutar dichos algoritmos.

3.1 Caso de Estudio

Los datos utilizados para este análisis fueron obtenidos de la Base de Datos de gestión académica AKADEMOS, de los estudiantes de la Universidad de las Ciencias Informáticas. Los mismos contienen específicamente la información personal y académica de los estudiantes que actualmente cursan desde el segundo hasta el quinto año; tomando de estos solamente la información histórica en su primer año de universidad.

3.1.1 Preparación de los datos

La Base de Datos se encontraba en un servidor SQL Server 2000, a través de consultas SQL se obtuvo el conjunto de datos para este análisis particular, integrando los datos de las tablas Estudiante, Provincia, NE_Padre, NE_Madre, TC_Procedencia, Vía_Ingreso, Sexo, Nota, Asignatura y Hoja_Matrícula_128. Luego se tomaron los resultados académicos de los estudiantes en su primer curso en la universidad; o sea, las notas de las asignaturas del primer año en la carrera. Se seleccionaron las asignaturas de mayor peso en ese curso escolar. Las asignaturas seleccionadas son las siguientes: Matemática I, Matemática Discreta, Introducción a la Programación, Programación I y Algebra Lineal. La consulta resultante se exportó a una planilla de cálculo.

El archivo fue formateado para cumplir con las restricciones del programa WEKA que fue utilizado para el procesamiento de los datos, y contiene 300 registros con las siguientes variables referidas a los estudiantes: Matemática I(Matemática_1), Introducción a la Programación (IP), Matemática Discreta (Matemática _ discreta), Algebra Lineal (Algebra_Lineal), Programación I(Programación_1), nivel de escolaridad de los padres(NE_Padre), nivel de escolaridad de las madres (NE_Madres), tipo de centro de procedencia(TC_Procedencia) y vía de ingreso a la universidad(Vía_Ingreso). Luego se procedió a la conversión del archivo en formato ARFF, empleando un editor de textos.

En las figuras 3.1 y 3.2 se observa la distribución de las variables Matematica_1, IP, Matematica_Discreta, Algebra_Lineal, Programacion_1, NE_Padre y NE_Madre. Ver Anexo1, Anexo2 y Anexo3 para las distribuciones de las demás variables.

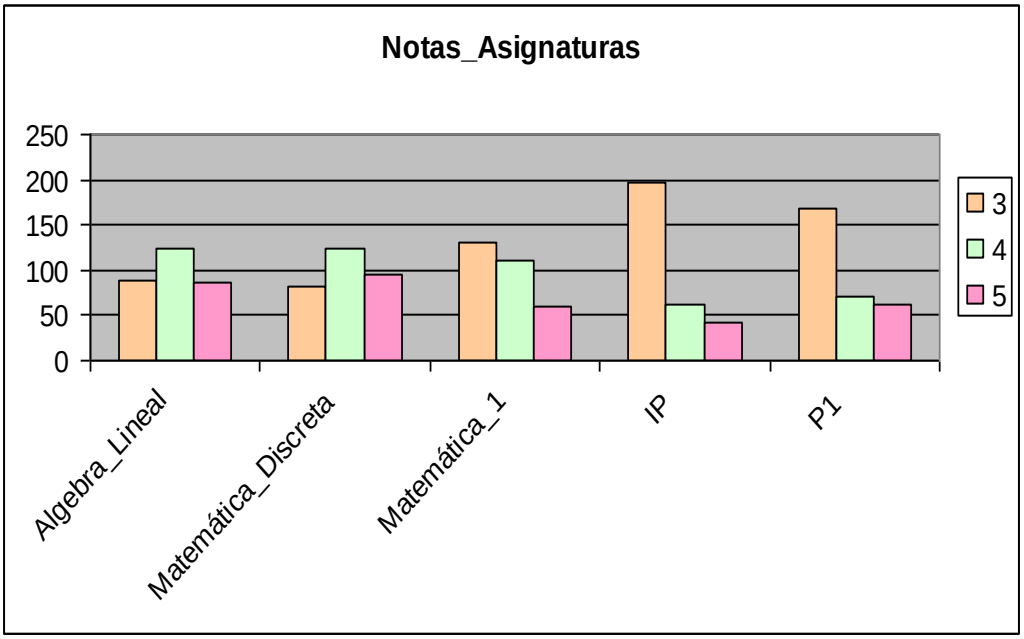


Figura 3.1: Distribución del atributo Notas_Asignaturas.

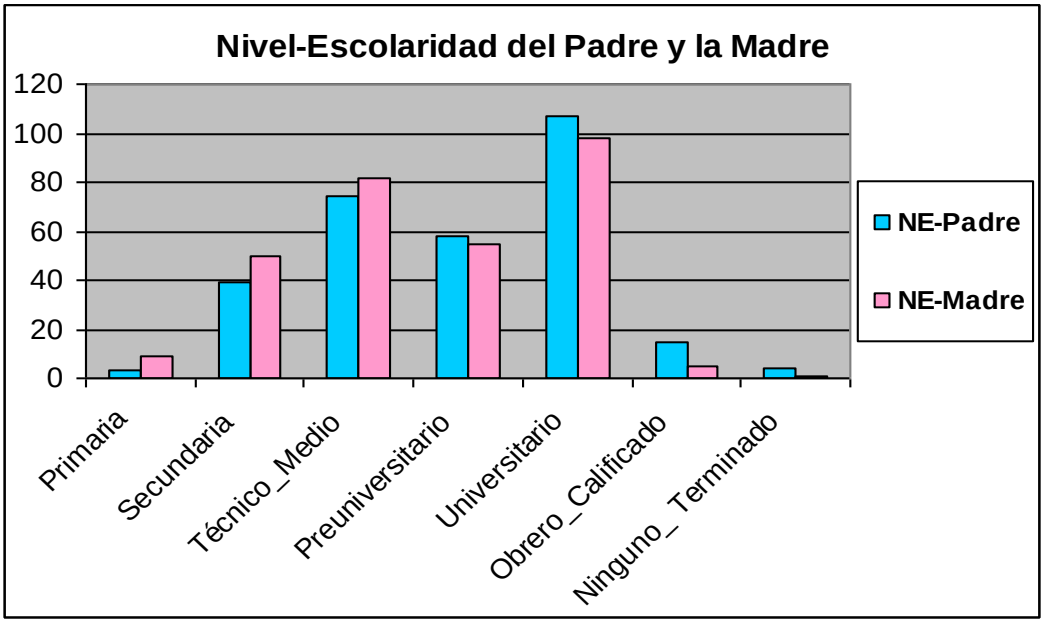


Figura 3.2: Distribución del atributo NE_Padre y NE_Madre.

3.2 Definiciones utilizadas en la evaluación de los algoritmos.

La eficiencia de los algoritmos se mide de acuerdo al tiempo de ejecución que la herramienta utiliza en la corrida de estos y al número de instancias clasificadas correctamente. Este número de instancias depende fundamentalmente de dos aspectos: los Falsos Positivos (FP) y los Verdaderos Positivos (TP).

Los Verdaderos Positivos (TP) constituyen la proporción de elementos de una clase x de entre todos los elementos de entrenamiento y que realmente pertenecen a dicha clase. Además, constituyen la parte de la clase que ha sido capturada. Los Falsos Positivos son aquellos elementos que han sido clasificados en una clase x, pero pertenecen a una clase diferente.

La precisión de estos algoritmos está dada por la proporción de ejemplos que realmente tienen clase x de entre todos los elementos que se han clasificado dentro de la clase x.

En la corrida de estos algoritmos la herramienta arroja una matriz de confusión que es de tamaño n*n, siendo n el número de clases. El número de instancias clasificadas correctamente es la suma de los números que se encuentran en la diagonal de la matriz, los demás están clasificados incorrectamente.

3.3 Evaluación de los métodos de clasificación.

En la investigación, la efectividad es considerada usualmente el criterio de evaluación más importante gracias a su confiabilidad para comparar diferentes metodologías. La forma usual de medir la efectividad de un clasificador es por medio de su *exactitud* (α), y el porcentaje de decisiones correctas.

Durante la experimentación es usual dividir Ω en los tres conjuntos disjuntos T_r (el *conjunto de entrenamiento*), v_a (el *conjunto de validación*) y T_e (el *conjunto de prueba*). El conjunto de entrenamiento es el conjunto de instancias observadas cuando el aprendiz construye el clasificador. El conjunto de validación es el conjunto de instancias utilizadas para optimizar parámetros en los clasificadores, o para seleccionar uno en particular. Entonces, el conjunto de prueba es el conjunto de instancias sobre las cuales se evalúa finalmente la efectividad del clasificador. Sin embargo, la partición anterior no es adecuada cuando la cantidad de datos es limitada; en tal caso, el camino estándar para calcular la exactitud de una técnica de aprendizaje es usar una *validación cruzada con 10 pliegues*. Esta técnica consiste en dividir aleatoriamente Ω en diez partes, conservando en cada partición la proporción original de las clases, posteriormente cada parte es mantenida una vez y el esquema de aprendizaje entrena sobre las nueve partes restantes, entonces la exactitud es calculada sobre la parte conservada fuera del proceso de entrenamiento. Así, el proceso es ejecutado un total de diez veces sobre diferentes conjuntos de entrenamiento. Finalmente, los diez estimados de exactitud son promediados para producir una completa estimación de la misma [8].

3.3.1 Evaluación del algoritmo Naive Bayes.

Se modelaron los atributos numéricos empleando un estimador de densidad de núcleo y se obtuvieron los siguientes resultados.

A continuación aparece el error que tiene este clasificador, según el conjunto de datos que se ha introducido. Como se puede observar sólo clasifica mal un 24.6667% de las instancias y tiene un error absoluto relativo de un 50.5934%.

Instancias Clasificadas Correctamente	226	75.3333%
Instancias Clasificadas Incorrectamente	74	24.6667%
Estadístico de Kappa	0.5807	
Error absoluto	50.5934%	
Número de instancias	300	

Tabla 3.1: Resultados obtenidos con el algoritmo Naive Bayes.

Verdaderos Positivos	Falsos Positivos	Precisión	Clases
0.845	0.205	0.84	3
0.535	0.127	0.567	4
0.754	0.075	0.719	5

Tabla 3.2: Precisión por clases del algoritmo Naive Bayes.

a	b	c	Clasificada
142	19	7	a=3
22	38	11	b=4
5	10	46	c=5

Tabla 3.3: Matriz de Confusión obtenida con el algoritmo Naive Bayes.

Se generaron tres clases correspondientes a las notas de cada asignatura. Los elementos que se encuentran en la diagonal de la Matriz de Confusión constituyen los elementos que cada clase clasificó correctamente.

Los resultados siguientes pertenecen al mismo algoritmo pero utilizando atributos nominales.

Instancias Clasificadas Correctamente	159	53%
Instancias Clasificadas Incorrectamente	141	47%
Estadístico de Kappa	0.3663	
Error absoluto	75.1254%	
Número de instancias	300	

Tabla 3.4: Resultados del Naive Bayes utilizando atributos nominales.

Verdaderos Positivos	Falsos Positivos	Precisión	Clases
0	0	0	Primaria
0.333	0.011	0.813	Secundaria
0.446	0.15	0.493	Tecnico_Medio
0.672	0.256	0.386	Preuniversitario
0.664	0.202	0.645	Universitario
0	0.004	0	Obrero_Calificado
0.75	0.007	0.6	Ninguno_Terminado

Tabla 3.5: Precisión por clases del Naive Bayes utilizando atributos nominales.

Analizando los resultados de este modelo se puede observar que disminuyó la precisión del clasificador pues sólo clasificó correctamente el 53% de las instancias. Además, la precisión del clasificador ahora oscila entre 0 y 0.813.

a	b	c	d	e	f	g	Clasificada como:
0	0	1	2	0	0	0	a = Primaria
0	13	7	13	6	0	0	b = Secundaria
0	1	33	20	20	0	0	c = Tecnico_Medio
0	2	9	39	7	0	1	d = Preuniversitario
0	0	14	20	71	1	1	e = Universitario
0	0	3	7	5	0	0	f= Obrero_Calificado
0	0	0	0	1	0	3	g = Ninguno_Terminado

Tabla 3.6: Matriz de Confusión del Naive Bayes utilizando atributos nominales.

Como se puede observar en la diagonal de la Matriz de Confusión obtenida se obtuvieron clases en las que no se agrupó elemento alguno.

3.3.2 Evaluación del algoritmo KNN.

Este algoritmo se ejecutó para distintos valores de KNN (desde 1 a 10), obteniéndose los mejores resultados para KNN=2.

Los siguientes resultados se obtuvieron utilizando los atributos numéricos de la base de datos. Se puede observar que se clasificó el 18.6667% de las instancias incorrectamente y la precisión en las clases es bastante alta.

Instancias Clasificadas Correctamente	244	81.3333%
Instancias Clasificadas Incorrectamente	56	18.6667%
Estadístico de Kappa	0.5926	
Error absoluto	43.9604%	
Número de instancias	300	

Tabla 3.7: Resultados obtenidos con el algoritmo KNN.

Verdaderos Positivos	Falsos Positivos	Precisión	Clases
0.98	0.356	0.838	3
0.525	0.059	0.696	4
0.465	0.019	0.8	5

Tabla 3.8: Precisión por clases del algoritmo KNN.

a	b	c	Clasificada
192	3	1	a=3
25	32	4	b=4
12	11	20	c=5

Tabla 3.9: Matriz de Confusión del algoritmo KNN.

En la Matriz de Confusión se puede observar que muy pocos elementos no fueron clasificados en la clase a la que pertenecían.

Realizando la evaluación con atributos nominales, al igual que en el algoritmo Naive Bayes, no se obtuvieron mejores resultados en cuanto a la cantidad de instancias correctamente clasificadas y a la precisión de las clases.

Instancias Clasificadas Correctamente	179	59.6667%
Instancias Clasificadas Incorrectamente	121	40.3333%
Estadístico de Kappa	0.4641	
Error absoluto	63.5643%	
Número de instancias	300	

Tabla 3.10: Resultados del KNN utilizando atributos nominales.

Se puede observar que aumentó considerablemente el número de instancias incorrectamente clasificadas. Esto se puede analizar mejor en la Matriz de Confusión obtenida. Además, la precisión del clasificador al utilizar atributos nominales resultó ser muy baja.

Verdaderos Positivos	Falsos Positivos	Precisión	Clases
1	0	0	Primaria
0.692	0.119	0.466	Secundaria
0.649	0.212	0.5	Tecnico_Medio
0.448	0.062	0.634	Preuniversitario
0.701	0.119	0.765	Universitario
0	0.004	0	Obrero_Calificado
0	0	0	Ninguno_Terminado

Tabla 3.11: Precisión por clases del KNN utilizando atributos nominales.

a	b	c	d	e	f	g	Clasificada como:
3	0	0	0	0	0	0	a = Primaria
1	27	4	2	5	0	0	b = Secundaria
2	7	48	4	12	0	0	c = Tecnico_Medio
0	11	18	26	3	0	0	d = Preuniversitario
0	8	17	7	75	1	0	e = Universitario
0	1	9	2	3	0	0	f = Obrero_Calificado
0	4	0	0	0	0	0	g = Ninguno_Terminado

Tabla 3.12: Matriz de Confusión del KNN utilizando atributos nominales.

3.3.3 Evaluación del algoritmo K*.

Primeramente se realizó la evaluación del algoritmo utilizando atributos numéricos y se obtuvo que el 99.6667% de las instancias se clasificaron correctamente con un error absoluto relativo de 0.01.

Instancias Clasificadas Correctamente	299	99.6667%
Instancias Clasificadas Incorrectamente	1	0.3333%
Estadístico de Kappa	0.9935	
Error absoluto	2.9377%	
Número de instancias	300	

Tabla 3.13: Resultados obtenidos con el algoritmo K*.

Verdaderos Positivos	Falsos Positivos	Precisión	Clases
1	0.01	0.995	3
0.984	0	1	4
1	0	1	5

Tabla 3.14: Precisión por clases del algoritmo K*.

Como se puede observar en la Tabla 3.14 la precisión de este clasificador es muy alta a la hora de trabajar con atributos numéricos.

a	b	c	Clasificada
196	0	0	a=3
1	60	0	b=4
0	0	43	c=5

Tabla 3.15: Matriz de Confusión del algoritmo K*.

En la Matriz de Confusión casi todos los elementos se encuentran en la diagonal, o sea, casi todos los elementos fueron clasificados correctamente en la clase a la que pertenecían.

Al evaluar el algoritmo con atributos nominales, se puede observar que el porcentaje de instancias bien clasificadas disminuyó levemente.

Instancias Clasificadas Correctamente	291	97%
Instancias Clasificadas Incorrectamente	9	3%
Estadístico de Kappa	0.96	
Error absolute	11.0294%	
Número de instancias	300	

Tabla 3.16: Resultados del K* utilizando atributos nominales.

Verdaderos Positivos	Falsos Positivos	Precisión	Clases
0.667	0	1	Primaria
0.974	0	1	Secundaria
0.973	0.018	0.947	Tecnico_Medio
0.966	0.008	0.966	Preuniversitario
1	0.016	0.973	Universitario
0.8	0	1	Obrero_Calificado
1	0	1	Ninguno_Terminado

Tabla 3.17: Precisión por clases del K* utilizando atributos nominales.

Se puede observar una disminución en la precisión del clasificador, no obstante la misma oscila entre 0.947 y 1.

a	b	c	d	e	f	g	Clasificada como:
2	0	1	0	0	0	0	a = Primaria
0	38	0	0	1	0	0	b = Secundaria
0	0	72	2	0	0	0	c = Tecnico_Medio
0	0	0	56	2	0	0	d = Preuniversitario
0	0	0	0	107	0	0	e = Universitario
0	0	3	0	0	12	0	f = Obrero_Calificado
0	0	0	0	0	0	4	g = Ninguno_Terminado

Tabla 3.18: Matriz de Confusión del K* utilizando atributos nominales.

En la Matriz de Confusión la mayoría de los elementos se encuentran en la diagonal, o sea, fueron clasificados correctamente.

3.3.4 Comparación de los algoritmos de clasificación

Para el caso de estudio analizado anteriormente siempre será mejor utilizar el último algoritmo, aunque seguirá sin ser suficiente para modelarlo.
Los resultados que se obtuvieron son los siguientes (entre paréntesis se indica si se ha modificado algún parámetro que mejore el resultado):

Método	Instancias correctamente clasificadas para atributos nominales.	Instancias correctamente clasificadas para atributos numéricos.
Naive Bayes	53%	75.3333%
KNN	59.6667%	81.3333%
K*	97%	99.6667%

Tabla 3.19: Resultados de los algoritmos de clasificación.

Resalta el algoritmo K*, pues realizó una mejor clasificación de las instancias en relación con los restantes algoritmos. Además en la matriz de confusión, la diagonal aportó más elementos que los demás, lo que significa que clasificó mejor los datos proporcionados por la base de datos.

3.4 Evaluación de modelos de agrupamiento

Este tipo de modelos son más difíciles de evaluar, ya que no existe una clase o un valor numérico donde medir las veces que el modelo aprendido predice correctamente.

Una opción que se ha considerado es utilizar la distancia entre los grupos como medida de calidad del modelo de agrupamiento. Cuanto mayor sea la distancia entre los grupos, significa que se ha efectuado una mejor separación, y por tanto el modelo se considera mejor. Falta determinar qué se considera como distancia entre grupos; se han estudiado varias opciones, podemos destacar las siguientes:

. Distancia media: la distancia entre dos grupos se calcula como la media de las distancias entre todos los componentes de ambos grupos.
. Distancia simple: se considera en este caso la distancia entre los vecinos más próximos de los dos grupos.
. Distancia completa: similar a la anterior, pero utilizando los vecinos más lejanos, es decir la distancia entre los componentes que se encuentren a más distancia.

Otra alternativa para conocer si un determinado modelo de agrupamiento es adecuado para un conjunto de datos consiste en aprender varios modelos desde ese mismo conjunto utilizando diversas técnicas de aprendizaje. Si los comparamos entre sí y coinciden, podremos pensar que esa agrupación es acertada. Para evaluar la similitud entre dos modelos de agrupamiento podemos utilizar una estrategia, similar a la aproximación entrenamiento/ test de clasificación y regresión, basada en la partición de los datos en dos partes. La primera de ellas se utiliza para construir modelos de agrupamiento, y la segunda para comprobar si los modelos construidos son similares. Para comprobar la similitud entre dos modelos *a* y *b,* con n_a y n_b grupos respectivamente, generamos una matriz con n_a filas y n_b columnas, y se inicializa a O. Por cada dato perteneciente al conjunto de test utilizamos los modelos *a* y *b* para que asignen un grupo cada uno. Los grupos se utilizan para incrementar una celda de la matriz de comparación. Por ejemplo, si *a* retorna 1, y *b* retorna 2, incrementaremos la posición (1, 2). De esta manera, tras evaluar todo el conjunto de test, si los modelos son similares, la matriz estará concentrada en unas pocas celdas. Si son diferentes, la matriz estará bastante dispersa.

Como conclusión, destacamos que los modelos descriptivos en general, son complicados de evaluar debido a la ausencia de una clase determinada donde medir el grado de acierto del modelo. La mejor evaluación de este tipo de modelos es saber si el modelo resultado de la fase de aprendizaje tiene un comportamiento útil cuando se utilice en su área de aplicación.

3.4.1 Evaluación del algoritmo COBWEB

Durante la ejecución del algoritmo se forma un árbol (*árbol de clasificación*) donde las hojas representan los segmentos y el nodo raíz engloba por completo el conjunto de datos de entrada. Al principio, el árbol consiste en un único nodo raíz. Las instancias se van añadiendo una a una y el árbol se va actualizando en cada paso.

Como analizamos en el capítulo anterior el algoritmo es muy sensible a dos parámetros:
Acuity: Este parámetro es muy necesario, ya que la utilidad de categoría se basa en una estimación de la media y la desviación estándar del valor de los atributos, pero cuando se estima la desviación estándar del valor de un atributo para un nodo en particular, el resultado es cero si dicho nodo sólo contiene una instancia. Así pues, el parámetro acuity representa la medida de error de un nodo con una sola instancia, es decir, establece la varianza mínima de un atributo [5].

Cut-off: Este valor se utiliza para evitar el crecimiento desmesurado del número de segmentos. Indica el grado de mejoría que se debe producir en la utilidad de categoría para que la instancia sea tenida en cuenta de manera individual. En otras palabras: cuando no es suficiente el incremento de la utilidad de categoría en el momento en el que se añade un nuevo nodo, ese nodo se corta, conteniendo la instancia otro nodo ya existente [5].

Al algoritmo COBWEB no hay que proporcionarle el número exacto de grupos que queremos, sino que en base a los parámetros anteriormente mencionados encuentra el número óptimo.
Primeramente se ejecutó el algoritmo con valor de 0.0028209479177387815 para el parámetro Cut-off, este es el valor que brinda WEKA por defecto, utilizando atributos nominales, en este caso Nivel_Escolaridad_Padre, obteniendo los siguientes resultados:

Número de fusiones	73
Número de divisiones	70
Número de grupos	20

Tabla 3.20: Resultados del algoritmo COBWEB.

1	46(15%)
2	39(13%)
4	24(8%)
5	22(7%)
6	22(7%)
8	19(6%)
9	16(5%)
10	14(5%)
11	17(6%)
13	13(4%)
15	8(3%)
16	2(1%)
17	11(4%)
18	19(6%)
19	28(9%)

Tabla 3.21: Instancias por grupos.

1	2	4	5	6	8	9	10	11	13	15	16	17	18	19	Asignados a los grupos:
0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	Primaria
2	4	4	5	0	5	4	1	1	2	2	0	1	5	3	Secundaria
18	13	3	5	4	2	0	1	6	4	4	0	4	4	6	Tecnico_Medio
5	6	3	2	7	6	5	3	2	2	1	1	3	5	7	Preuniversitario
18	11	11	8	10	6	7	7	7	5	1	1	2	4	9	Universitario
3	2	0	2	1	0	0	2	1	0	0	0	1	0	3	Obrero_Calificado

0	1	3	0	0	0	0	0	0	0	0	0	0	0	0	Ninguno_Terminado

Tabla 3.21: Asignación de clases a los grupos.

Instancias incorrectamente agrupadas: 250.0 83.3333 %

De los 20 grupos obtenidos 15 contienen de 2 a 46 elementos y los restantes por ninguno. El número tan elevado de grupos hace pensar que el valor de cut-off elegido no es el más adecuado, es necesario incrementarlo. Se realizaron varios intentos donde se le dieron diferentes valores al parámetro *cut-off* en busca de menor cantidad de clústeres y de un menor por ciento de instancias incorrectamente clasificadas, pero no se obtuvieron resultados aceptables.

Nuevamente se ejecutó el algoritmo en busca de resultados favorables, esta vez con atributos numéricos. En el mejor de los casos se obtuvieron 82 fusiones, 65 divisiones y 386 grupos, de ellos 220 contenían de 1 a 3 elementos cada uno, y el 97,6667% de las instancias fueron clasificadas incorrectamente.

3.4.2 Evaluación del algoritmo K-medias

Al algoritmo de las k-medias hay que proporcionarle de antemano el número de grupos en los que se quiere segmentar la base de datos. Por lo tanto el algoritmo de k-medias se aplica sobre la base de datos seleccionada, para un número de grupos de N=3, utilizando para ello atributos nominales, en este caso Nivel_Escolaridad_Padre. Este algoritmo esta concebido para trabajar solamente sobre atributos numéricos, pero para realizar una buena comparación en relación con el algoritmo COBWEB, se decidió analizar los resultados que se obtenían al ejecutarlo sobre atributos nominales. Se obtuvieron los siguientes resultados:

0	171(57%)
1	78(26%)
2	51(17%)

Tabla 3.22: Instancias por grupos.

0	1	2	Asignación
2	0	1	Primaria
24	12	3	Secundaria
38	24	12	Tecnico_Medio
44	10	4	Preuniversitario
53	28	26	Universitario
9	2	4	Obrero_Calificado
1	2	1	Ninguno_Terminado

Tabla 3.22: Asignación de clases a los grupos.

Grupo 0	Preuniversitario
Grupo1	Tecnico_Medio
Grupo 2	Universitario

Tabla 3.23: Grupos creados por el algoritmo K-medias utilizando atributos nominales.

Instancias incorrectamente agrupadas: 206.0 68.6667 %

En este caso el número de elementos por grupos es de 171, 78, y 51, respectivamente. Se realizaron nuevos intentos aumentando el número de grupos con el objetivo de obtener mejores resultados, pero cada vez aumentaba el número de instancias clasificadas incorrectamente.

Se ejecutó el algoritmo con atributos numéricos, obteniéndose mejores resultados, comprobando su utilidad sobre los mismos. A continuación se muestran los resultados con los atributos Matemática_1 y P1:

Atributo: Matematica_1

0	172(57%)
1	75(25%)
2	53(18%)

Tabla 3.24: Instancias por grupos para el atributo Matematica_1.

0	1	2	Asignación
109	14	7	3
47	46	17	4
16	15	29	5

Tabla 3.25: Asignación de clases a los grupos.

Grupo 0	3
Grupo1	4
Grupo 2	5

Tabla 3.26: Grupos creados para el atributo Matematica_1.

Instancias incorrectamente agrupadas: 116.0 38.6667 %

Atributo: IP

0	131(44%)
1	108(36%)
2	61(20%)

Tabla 3.27: Instancias por grupos para el atributo IP.

0	1	2	Asignación
122	61	13	3
6	34	21	4
3	13	27	5

Tabla 3.28: Asignación de clases a los grupos.

Grupo 0	3
Grupo1	4
Grupo 2	5

Tabla 3.29: Grupos creados para el atributo IP.

Instancias incorrectamente agrupadas: 117.0 39 %

En cada uno de los atributos anteriores se obtuvieron 3 grupos, todos compuestos por elementos significativos. En cuanto a la cantidad de instancias incorrectamente clasificadas no sobrepasan el 39%.

3.4.3 Comparación de la evaluación de los algoritmos de agrupamiento

Si se comparan los valores de los grupos obtenidos por los algoritmos bajo estudio, se observa que el método COBWEB, perteneciente a la familia de agrupamiento jerárquico, ofrece 20 grupos, de los cuales 15 contienen elementos y 5 son significativos. Además clasifica incorrectamente 250 instancias que constituyen el 83,3% de los datos nominales. En cuanto a atributos numéricos los resultados no fueron favorables para este algoritmo. El algoritmo K-medias, perteneciendo a la familia de particionado y recolocación ofrece mejores resultados que COBWEB, pues cada cluster obtenido es significativo y no existen grupos a los que no se les haya asignado clase. Además, cuando se ejecutó sobre atributos nominales clasificó incorrectamente 206 instancias que constituyen el 68.6667% de los datos, observándose una notable diferencia en este aspecto en relación con el método anterior. También, realizó una mejor clasificación sobre los atributos numéricos que se le proporcionaron obteniéndose como máximo un 39% de las instancias incorrectamente clasificadas. En la tabla siguiente se recogen los resultados a modo de resumen:

Algoritmos	Nominales Grupos- Instancias	Numéricos Grupos-Instancias
COBWEB	20- 83.3%	386- 97.6%
K-Medias	5- 71.3%	4- 51%

Tabla 3.30: Resultados algoritmos de agrupamiento.

En este capítulo se han evaluado los algoritmos de agrupamiento y clasificación seleccionados en el capítulo anterior para realizar la comparación, en la herramienta WEKA.

En los algoritmos de clasificación resalta el K* cuyos resultados en este caso de estudio sobrepasaron los de los algoritmos KNN y Naive Bayes. En el grupo de los algoritmos de agrupamientos se destaca el K-Medias superando los resultados obtenidos en la evaluación realizada por el algoritmo COBWEB.

CONCLUSIONES

Con el desarrollo de este trabajo se cumplieron los objetivos propuestos al inicio, pues se realizó una valoración del estado del arte de la Minería de datos, así como un estudio de las características generales de las técnicas de clasificación y agrupamiento. También se analizaron algunos de los algoritmos más representativos de estas técnicas con la herramienta WEKA, lo que permitió analizar la eficiencia de los mismos y arrojó interesantes resultados durante la comparación.

Con los resultados obtenidos en este trabajo se pueden obtener reglas y patrones más certeros que contribuyan al mejoramiento del rendimiento académico de los estudiantes.

RECOMENDACIONES

Se recomienda para futuros trabajos:

- Continuar trabajando en esta línea de investigación debido a las potencialidades que tiene la misma, en la búsqueda de conocimientos en diversas áreas de la economía y de las ciencias sociales
- Aplicar los resultados de esta investigación en proyectos de Minería de datos.
- Implementar en la herramienta WEKA extensiones de estos algoritmos con vistas a elevar su eficiencia tanto en tiempo de ejecución como en los parámetros de clasificación.

REFERENCIAS BILIOGRÁFICAS

1. **Hernández, J. y Ferri, C.** *Introducción al Weka.Curso de Doctorado Extracción Automática de Conocimiento en Bases de Datos e Ingeniería del Software.* Valencia : Universitat Politècnica de Valencia, 2006.

2. **María García, Aránzazu Álvarez Sierra.** *Análisis de Datos en WEKA – Pruebas de Selectividad.* Valencia : Ingeniería de Telecomunicación. Universidad Carlos III , 2002.

3. **Suárez-, Ianisse Quinzán.** *Comparación entre métodos de agrupamiento en la selección de características.* Santiago de Cuba : Departamento de Computación, Universidad de Oriente, Cuba, 2005.

4. **Miguel Garre, Juán José Cuadrado, Miguel Ángel Sicilia.** *Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software.* Madrid : Ingeniería Informática Universidad de Alcalá, 2002.

5. **Barreno, Julio Escribano.** *Análisis de Datos mediante WEKA.* Valladolid : Universidad de Valladolid, 2001.

6. **Broitman, Erika Vilches González e Iván A. Escobar.** *Minería de Datos.* Campus, Mexico : Ciencias Computacionales ITESM, 2007.

7. **Edna, Hernandez Valadez.** *Algoritmo de clustering basado en entropia para descubrir grupos en atributos de tipo mixto.* Mexico : Departamento de Ingenieria Electrica del Instituto Politecnico Nacional de Mexico, 2006.

8. **Valero, Alberto Tellez.** *Extracción de Información con Algoritmos de Clasificación.* Tonantzintla, Pue. : Instituto Nacional de Astrofísica, Óptica y Electrónica., 2005.

9. **Jose Henandez Orallo, Jose Ramirez Quintana, Cesar Ferri.** *Introducción a la Minería de Datos.* Madrid : Pearson, 2001.

10. **Carlos G. Figuerola, José L. Alonso Berrocal,Angel F. Zazo Rodríguez, Emilio Rodríguez.** *Algunas Técnicas de Clasificación Automática de Documentos.* Salamanca : s.n., 1999.

11. **Suils, Pilar Sopena.** *Extracción y Recuperacion de información no supervisada.* Valladolid : s.n., 2008.

12. **José Manuel Molina López, Jesús García Herrero.** *Técnicas de Análisis de Datos:Aplicaciones practicas utlizando Microsoft Excel y WEKA.* Madrid : s.n., 2004.

13. **Jorge Humberto Jaramillo Monsalve, John Alexander López Noreña.** *Datamining .* Medellín : s.n., 2000.

14. **Vallejos, Sofia J.** *Minería de Datos.* Corrientes, Argentina : s.n., 2006.

15. **Aguilera, Manuel Ernesto Acosta.** *Minería de Datos y Descubrimiento de Conocimiento.* Vedado, La Habana : s.n., 2004.

BIBLIOGRAFÍA

E. Castillo, J. G. *Expert system and probabilistic network models.* New York : Press, 2002.

Hernandez Orallo, J. R. *Introduccion a la mineria de datos .* Madrid : Pearson, 2004.

J. Han, M. K. *Data Minnig: Concept and Tecniques .* San Francisco : Academic Press, 2001.

IBM Press, I. D. *Utilización del Visualizador de Asociaciones .* USA : IBM Press, 1999.

Jhon Wiley, A. S. *Data Warehouse, Data Mining and OLAP .* USA : Press, 1997.

Mario José Ramirez Quintana, J. H. *Extraccion Automática de Conocimiento en Bases de Datos e Ingeniería del Sofware.* España : Pressman, 2003.

White, C. J. *IBM Enterprise Analiytics for the Intelligent e-Business.* USA : IBM Press., 2001.

D., Tomas. *Controlling Software Projects.* USA : Yourdon Press., 1982.

T., DeMarco. *Controling Software Projects.* USA : Yourdon Press, 1982.

Allen, K. T. *Modeling Software Quality with Clasification Trees.* Singapore : Hoang Pam Editor., 1999.

Miguel Garre, Juán José Cuadrado, Miguel Ángel Sicilia. *Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software.* Madrid : Ingeniería Informática Universidad de Alcalá, 2002.

Suárez-, Ianisse Quinzán. *Comparación entre métodos de agrupamiento en la selección de características.* Santiago de Cuba : Departamento de Computación, Universidad de Oriente, Cuba, 2005.

Hernández, J. y Ferri, C. *Introducción al Weka.Curso de Doctorado Extracción Automática de Conocimiento en Bases de Datos e Ingeniería del Software.* Valencia : Universitat Politècnica de Valencia, 2006.

María García, Aránzazu Álvarez Sierra. *Análisis de Datos en WEKA – Pruebas de Selectividad.* Valencia : Ingeniería de Telecomunicación. Universidad Carlos III , 2002.

Suárez-, Ianisse Quinzán. *Comparación entre métodos de agrupamiento en la selección de características.* Santiago de Cuba : Departamento de Computación, Universidad de Oriente, Cuba, 2005.

Miguel Garre, Juán José Cuadrado, Miguel Ángel Sicilia. *Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software.* Madrid : Ingeniería Informática Universidad de Alcalá, 2002.

Acosta Franco, Javier. *"Aplicación de los Sistemas Clasificadores tradicionales al análisis de datos. Adquisición automática de reglas.* Madrid : s.n., 2002.

R. Agrawal, T. Imielinski, and A. Swami. *Mining association rules between sets of items in large databases.* Washington, DC, : s.n., 1993.

Srikant, R. Agrawal and R. *Fast algorithms for mining association rules in large databases.* San Jose, CA : s.n., 1994.

M.Berry, and G.Linoff. *Data Mining Techniques for Marketing,Sales, and Customer Support.* NY : s.n., 1997.

S. Brin, R. Motwani, and C. Silverstein. *Beyond market basket:Generalizing association rules to correlations.* Tucson, AZ : s.n., 1997.

G. Briscoe, and T. Caelli. *A Compendium of Machine Learning.Symbolic Machine Learning.* New Jersey : s.n., 1996.

Hart., R. Duda and P. *Pattern Classification and Scene Analysis.* New York : s.n., 1973.

Dobson, A. J. *An Introduction to Generalized Linear Models.* New York : s.n., 1990.

Fayyad, U. *Advanced in Knowledge Discovery and Data Mining.* L.A. : MIT Press, 1996.

Kowalski, Gerald. *Information Retrieval Systems. Theory and Implementation.* s.l. : Kluwer Academic Publishers, 1998.

H. Schutze, David A. Hull, and Jan Q. Pedersen. *A comparasion of classifers and document representations for the routing problem.* SIGIR : s.n., 1995.

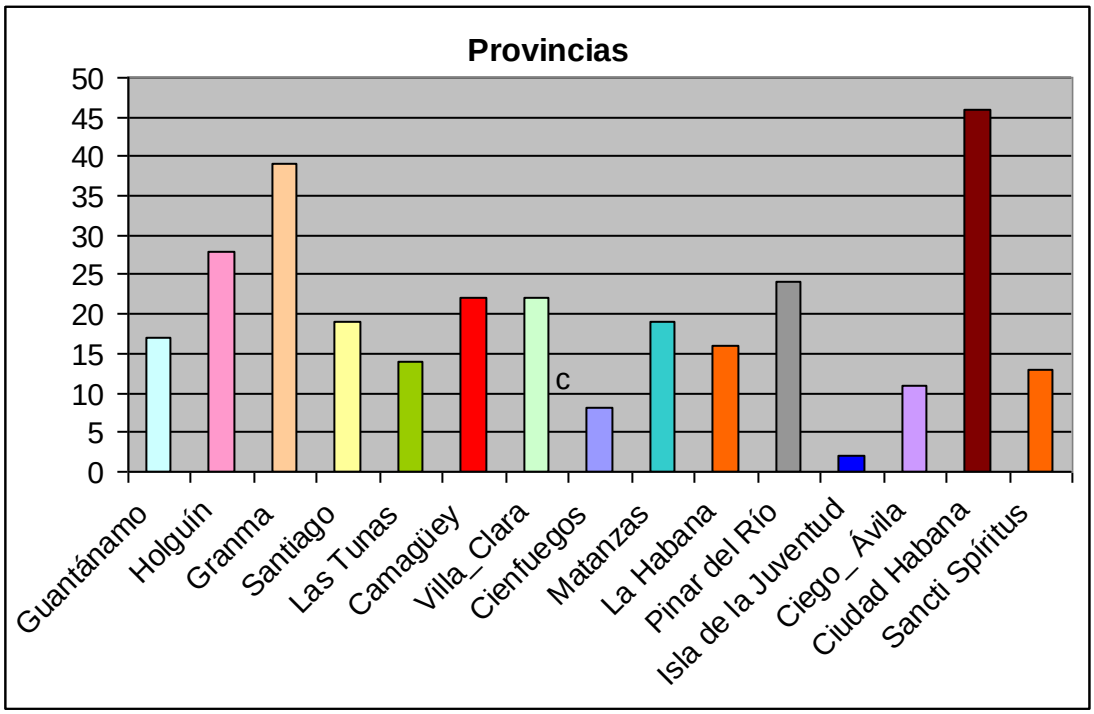
Weili Wu, Hui Xiong, and Shashi Shekhar. *Clustering and Information Retrieval.* Kluwer : s.n., 2003.

Chakrabarti, Soumen. *Mining the Web : discovering knowledge from hypertext data.* San Francisco, CA : s.n., 2003.

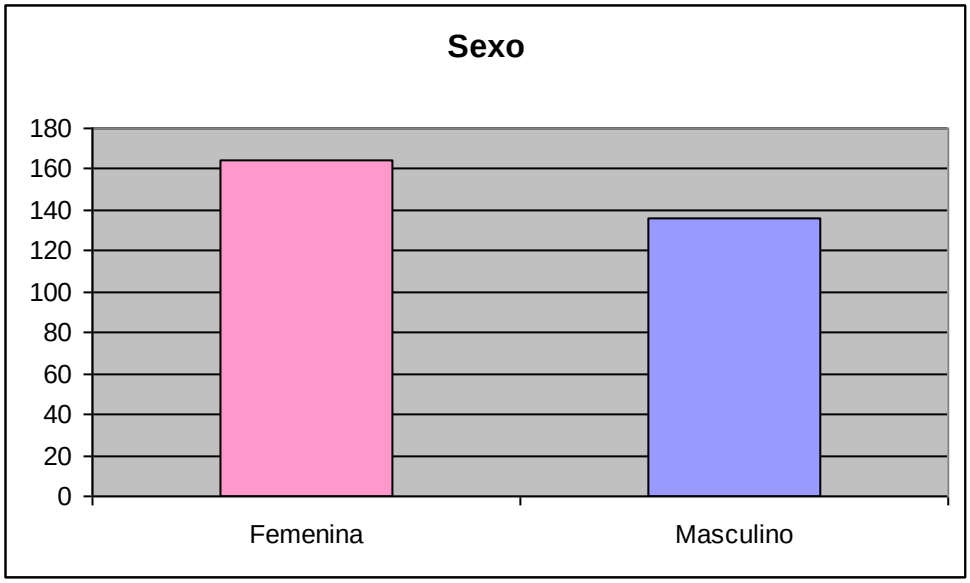
Kowalski, Gerald. *Information Retrieval Systems. Theory and Implementation.* s.l. : Kluwer Academic Publishers, 1998.

ANEXOS

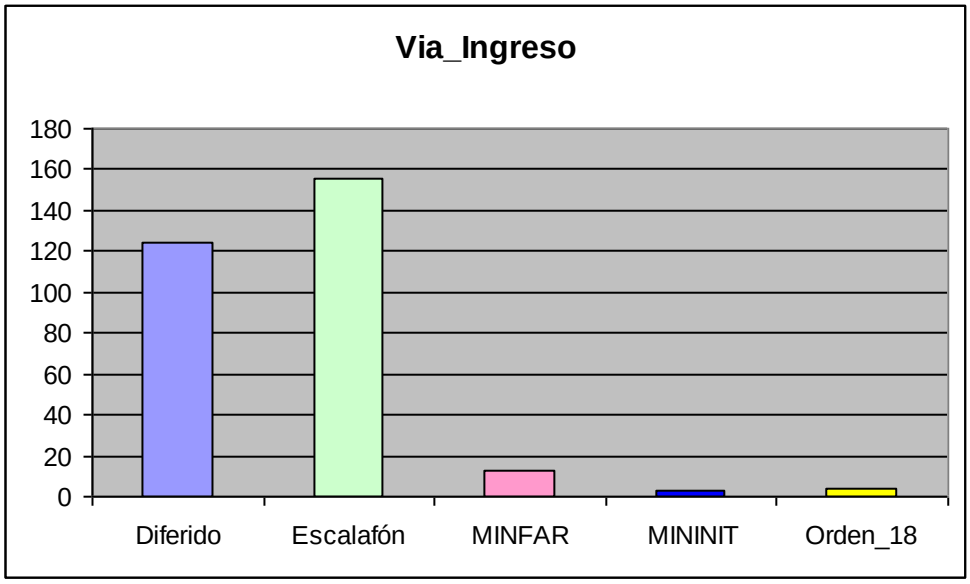
Anexo1: Distribución de la variable Provincia.



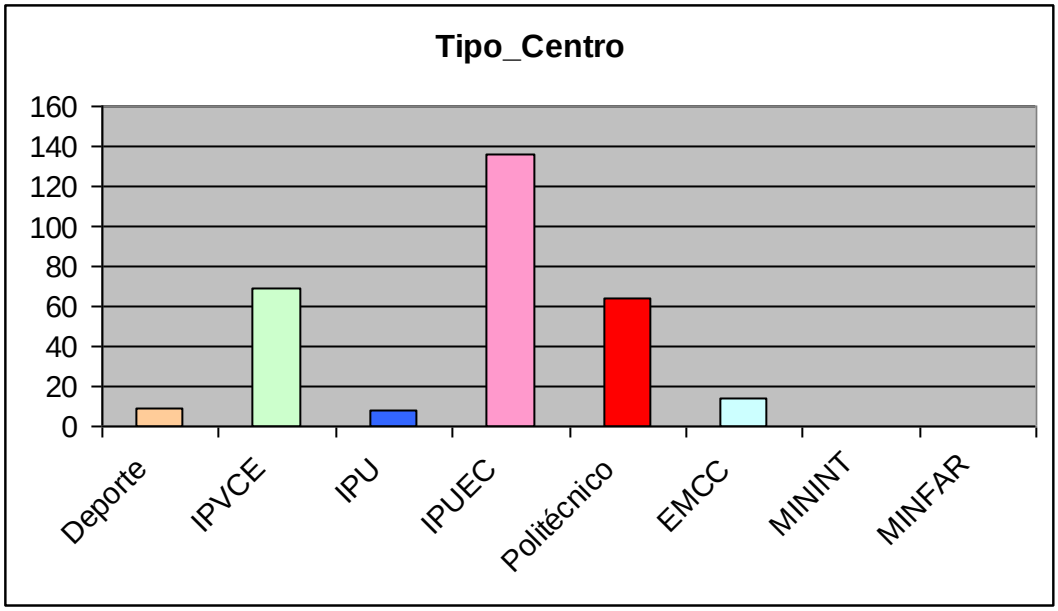
Anexo2: Distribución de la variable Sexo.



Anexo3: Distribución de la variable Vía_Ingreso.



Anexo4: Distribución de la variable Tipo_Centro.



GLOSARIO

Asociación: Toda agrupación de elementos, realizada con un cierto propósito de permanencia para el cumplimiento de una finalidad cualquiera, de un interés común para los asociados, siempre que sea lícito.

Bias (sesgo): Es un error que aparece en los resultados de un estudio debido a factores que dependen de la recogida, análisis, interpretación, publicación o revisión de los datos que pueden conducir a conclusiones que son sistemáticamente diferentes de la verdad o incorrectas acerca de los objetivos de una investigación.

Correctly Classified Instances: Porcentaje de clasificaciones correctas sobre los datos de entrenamiento o en un experimento de validación cruzada.

Cross-validation: Pulsando el botón *Cross-validation* Weka realizará una validación cruzada estratificada del número de particiones dado (*Folds*). La validación cruzada consiste en: dado un número *n* se divide los datos en *n* partes y, por cada parte, se construye el clasificador con las *n*−1 partes restantes y se prueba con esa.

DataWarehouse: Un almacén de datos, es una colección de datos orientada a un determinado ámbito (empresa, organización, etc.), integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza. Se trata, sobre todo, de un expediente completo de una organización, más allá de la información transaccional y operacional, almacenado en una base de datos diseñada para favorecer el análisis y la divulgación eficiente de datos (especialmente OLAP, *procesamiento analítico en línea*).

Discriminación: La discriminación es la descubierta de características o propiedades que diferencian las clases examinadas de otras clases.

Evolución: La evolución es la detección y/o evaluación del comportamiento de ciertos objetos en un período de tiempo. Esto puede incluir caracterización, clasificación, asociación, o agrupamiento de datos relacionados al tiempo.

False Positive (FP) Rate: La proporción de ejemplos que han sido clasificados dentro de la clase *x*, pero pertenecen a una clase diferente. En la matriz de confusión es la suma de la columna de la clase *x* menos el elemento diagonal menos la suma de las filas del resto de las clases.

Incorrectly Classified Instances: Porcentaje de clasificaciones incorrectas sobre los datos de entrenamiento o en un experimento de validación cruzada.

Kappa statistic: Un índice que compara el acuerdo en contra de lo que cabría esperar por azar. Kappa puede ser pensado como la posibilidad de un acuerdo proporcional corregido, y los valores posibles van desde 1 (perfecto acuerdo) a través de 0 (ningún acuerdo por encima de lo esperado por azar) a -1 (total desacuerdo).

Matriz de confusión: También se llama tabla de contingencia. Es de tamaño n*n, siendo n el número de clases. El número de instancias clasificadas correctamente es la suma de los números en la diagonal de la matriz; los demás están clasificados incorrectamente.

Polución: Es la introducción por causas antrópicas de determinadas sustancias de formas de energía que producen efectos biológicos adversos para los seres humanos, las actividades económicas o para el ecosistema.

Precisión: Proporción de ejemplos que realmente tienen clase *x* de entre todos los elementos que se han clasificado dentro de la clase *x*. En la matriz de confusión es el elemento diagonal dividido por la suma de la columna en la que estamos.

Taxonomía: La ciencia de ordenar a los organismos en un sistema de clasificación compuesto por una jerarquía de taxones anidados.

True Positive (TP) Rate: Es la proporción de elementos que están clasificados dentro de la clase *x*, de entre todos los elementos que realmente son de la clase *x*. Es la parte de la clase que ha sido capturada. En la matriz de confusión es el elemento diagonal dividido por la suma de todos los elementos de la fila.

Percentage split: Se define un porcentaje con el que se construirá el clasificador y con la parte restante se probará.

Predicción: La predicción es la estimativa futura o pronóstico de un posible valor de un dado ausente o la probable distribución de un valor con cierto atributo dentro de un conjunto de datos históricos dos datos analizados.