20/2/99 9(nueve)

Trabajo de grado

Data Mining

Generalidades y un enfoque al problema de Reglas de Asociación Cuantitativas

Director:
Prof. Lic. Gustavo Rossi

CoDirector:
Prof. Lic. Juan Ale

Alumnos:
Fontanari, Andrea Lorena
Marin, Carola

ABSTRACT

Con el correr del tiempo, las capacidades de generar y coleccionar datos se han incrementado con rapidez. El extenso uso de códigos de barras en al mayoría de los productos comerciales, la informatización de muchas actividades y de gestiones de gobierno, y los avances en herramientas de colección de datos; proveen enormes cantidades de ellos. La disponibilidad de poderosos sistemas de Bases de Datos y el crecimiento explosivo en los datos, generaron una urgente necesidad de algunas técnicas y herramientas. Estas deben poder, inteligentemente y automáticamente, transformar los datos procesados en información y conocimiento útil. En consecuencia, ***Data Mining*** *se ha transformado en una área de investigación, cuya importancia se incrementa día a día. En este trabajo tratamos de presentar esta área y sus generalidades, y luego nos ubicamos en el problema de encontrar Reglas de Asociación Cuantitativas.*

Introducción

Data Mining, representa un proceso no trivial de extracción de información implícita, previamente desconocida y potencialmente útil. Con este concepto, el primer objetivo de este trabajo es presentar esta área de investigación de creciente importancia. El segundo objetivo, es la investigación de una de las mas recientes técnicas de Data Mining, encontrar Reglas de Asociación Cuantitativas. A partir de la cual implementamos un software que realiza el mining de tales reglas.

El capítulo 1, establece un contexto para Data Mining, ubicándolo dentro del campo de investigación denominado: ***Descubrimiento de Conocimiento en Bases de Datos (K.D.D. Knowledge Discovery in Databases).*** Explicaremos las causas que dieron origen a este campo, daremos una definición del mismo y resumiremos los pasos que componen este proceso, lo cual mostrará que Data Mining es uno de ellos.

El capítulo 2, da una definición formal de Data Mining, y se establecen las diferencias entre ésta última y la de K.D.D, luego se explicarán las tareas que puede realizar, y se ejemplificaran cada una de ellas. Hay dos estilos de Data Mining, los que se desarrollarán y se confrontarán para poder ver sus utilidades. No obstante, muchas veces un modelo es seguido por el otro. Ambas modelizaciones tienen como primer paso la definición de los objetivos del mining, éstos sirven, para poder realizar una medición de efectividad comparándolos con los resultados obtenidos. Este tema es desarrollado, y se le introduce una medida denominada Lift, la cual da un parámetro para extender un modelo y obtener mejores resultados.

El capítulo 3, presenta algunos problemas que enfrenta el análisis de datos y las modelizaciones, luego se resume siete de las técnicas de Data Mining más

conocidas, la tarea o las tareas que involucran cada una de ellas, y por último sus ventajas y desventajas.

El capítulo 4, ilustra una de las técnicas más recientes, el mining de Reglas de Asociación. En principio, éste problema es definido para atributos booleanos, pero es corriente en las bases de datos e investigaciones tener datos categóricos y cuantitativos. Por ésto desarrollamos el problema de obtener Reglas de Asociaciones Cuantitativas, lo cual incluye sus dificultades y posibles soluciones, para una mejor explicación utilizamos un ejemplo que ilustra la problemática. Este capítulo, es el punto inicial, para luego introducir el software implementado, el cual trabaja con atributos categóricos y cuantitativos para generar Reglas de Asociaciones Cuantitativas.

El capítulo 5, presenta el desarrollo que realizamos del software mencionado anteriormente, en el cual se explica el algoritmo seleccionado, la interface y procedimientos que lo componen. Por último, se ilustran los resultados de experimentos de performance y escalabilidad aplicados al software utilizando diferentes bases de datos sintéticas.

1 Descubrir Conocimiento en Bases de Datos

1.1 Introducción

Actualmente los datos están siendo generados y acumulados a pasos agigantados, por esto, urge la necesidad de una nueva generación de técnicas y herramientas computacionales, que nos asistan en la extracción de información útil (conocimiento), a partir del rápido crecimiento del volumen de los datos. Estas técnicas y herramientas son el sujeto del surgimiento del área de descubrimiento de información en bases de datos (KDD - Knowledge Discovery in Databases).

En este capítulo haremos una breve descripción del proceso KDD, incluyendo su definición y los pasos que lo componen, con el fin de ubicarnos en el contexto de esta tesis.

1.2 Definición

Históricamente la idea de encontrar patrones útiles en los datos, ha dado una variedad de nombres, que incluyen términos como data mining, extracción de conocimiento, descubrir información, recolección de información, arqueología de datos y proceso de patrones de datos. El termino KDD fue inventado en el primer taller de KDD en 1989 (Piatetsky - Shapiro 1991) para enfatizar que "conocimiento" es el producto final de un descubrimiento a partir de los datos.

Definiremos KDD (Fayyad, Piatetsky, Shapiro & Smyth 1996):

Descubrir Conocimiento en Bases de Datos es el proceso no trivial de identificación de patrones validos, originales, potencialmente útiles y comprensibles en los datos.

El término *proceso* implica que KDD se compone de muchos pasos, los cuales comprenden: preparación de datos, búsqueda por patrones, evaluación del conocimiento y refinamiento, todos repetidos en múltiples iteraciones.

Por *no trivial* entendemos que, algunas búsquedas o inferencias son involucradas, por ejemplo, no es sólo un cálculo simple de cantidades predefinidas, como es obtener el valor promedio de un conjunto de números, sino que el patrón descubierto deberá ser "válido", sobre los nuevos datos con algún grado de certeza.

También queremos patrones *originales y potencialmente útiles*, por ejemplo, con algún beneficio para la tarea del usuario.

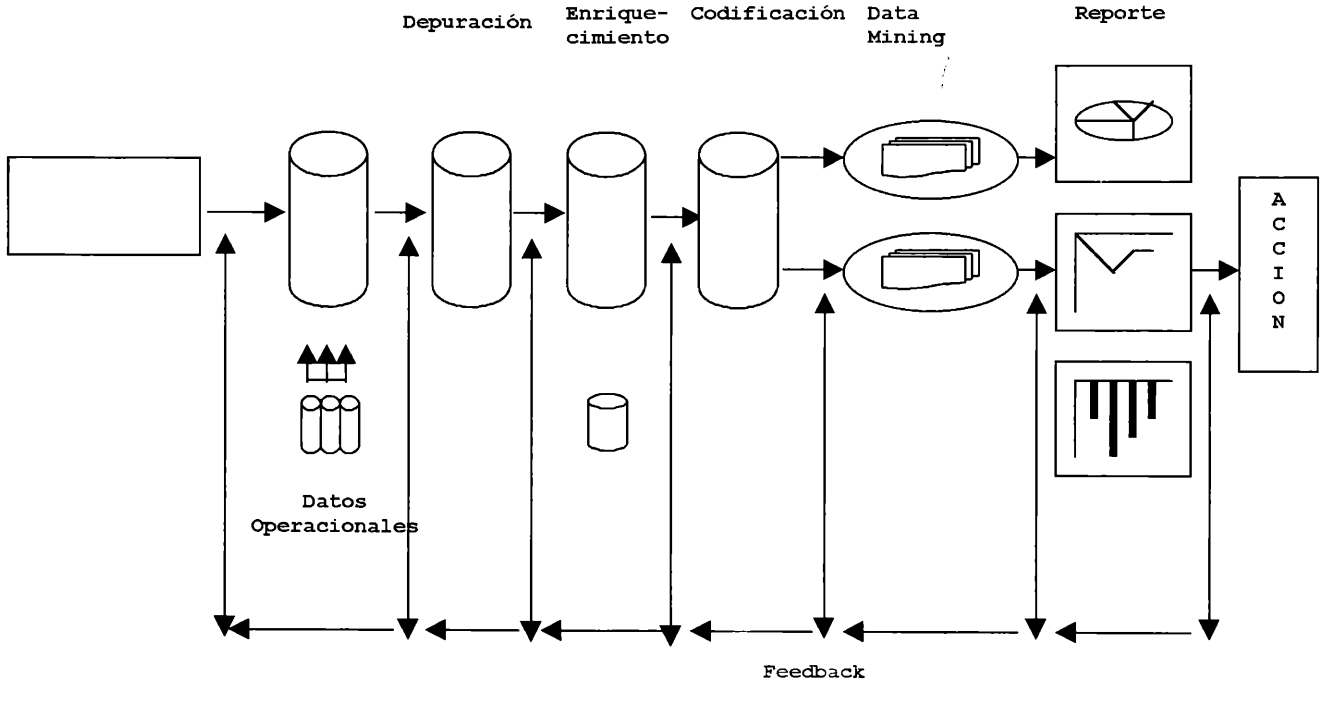
Finalmente, los patrones deberían ser *comprensibles*, si no es inmediatamente, luego de algún procesamiento.

1.3 Pasos del proceso KDD

El proceso KDD es interactivo e iterativo, involucrando numerosos pasos con muchas especificaciones dadas por el usuario.

Los pasos involucrados son:

- Selección de datos
- Depuración
- Enriquecimiento
- Codificación
- Data Mining
- Reporte



El proceso KDD

1.3.1 Selección de los datos

Una vez que tenemos formulados los requerimientos de información por parte del usuario, como: qué quiere saber y qué hará con ese conocimiento, el siguiente paso lógico es recolectar y seleccionar los datos necesarios.

En muchos de los casos los datos son guardados en bases de datos operacionales usadas por los sistemas de información de la organización, no obstante, recaudar esa información desde una base de datos (BD) centralizada, no siempre es una tarea fácil, puede involucrar conversiones de datos a bajo nivel. Podemos encontrar que los datos operacionales usados en diferentes partes de la organización varían en calidad: algunos departamentos mantienen BDs de alta calidad, que contienen información vital para su operatibilidad, mientras otros pueden tener pequeños conjuntos de datos, construidos sin un planeamiento previo, que no tienen valor para nuestra perspectiva. Algunas BDs son actualizadas día a día, otras pueden contener información de hace varios años. BDs diferentes, usadas en varias partes de la organización pueden usar técnicas completamente diferentes para identificar clientes: algunas usan strings y otras números. Estos ejemplos nos muestran que recaudar la información desde BDs operacionales para empezar con el proceso de KDD, no es trivial.

Finalmente la selección de los datos ésta fuertemente relacionada con la información requerida y la definición de las estructuras de las BDs y sus campos.

1.3.2 Depuración

Una vez que recolectamos los datos, el siguiente paso es la depuración, probablemente no se sabe la cantidad de información errónea (polución) que existe en los datos, entonces, es un bueno perder algún tiempo en examinar los datos para obtener una idea del problema. En la práctica puede ser dificultoso con BDs muy grandes. Cuando las BDs son muy grandes, generalmente se seleccionan ejemplos al azar y se analizan para tener una idea de que podemos encontrar. Casi todas las BDs en grandes organizaciones, contienen información errónea, y cuando empezamos a mirar los datos desde una perspectiva de data mining, ideas concernientes a la consistencia de los datos, cambian. Muchas veces, los datos vitales de una organización son almacenados correctamente, por ejemplo, en una compañía de seguros de vida, la fecha de nacimiento de los clientes estará correcta, mientras que no es anormal, que en un banco, el 30% o 40% de los campos de edad contengan blancos o datos incorrectos. Para una compañía de seguros de vida, tener la fecha de nacimiento correcta es de vital importancia, para un banco no lo es. Así, el negocio más importante de una organización es reflejado en la forma en que la consistencia de los datos es manejada. Por lo tanto las BDs contienen mucha polución, hasta hace poco a nadie se preocupaba por ello, porque ésto no afectaba a las operaciones normales. Estos cambios, sin embargo, cuando toman nuestra atención para data mining, las fechas de nacimientos no son almacenadas correctamente en la BD, es imposible descubrir patrones relativos a la edad. Por ésto antes de empezar con el proceso de data mining, tenemos que depurar los datos lo más posible, y puede ser hecho automáticamente en muchos casos. Esto no es real, sin embargo, se espera poder remover toda la polución, algunas anomalías en los datos van a ser solamente descubiertas durante el proceso de data mining mismo. Esto muestra nuevamente que KDD es un proceso iterativo. La polución puede tomar diferentes y sutiles niveles, chequear la consistencia del dominio, necesita ser

llevado fuera por los programas que tengan un profundo conocimiento de la semántica de los atributos que van a ser chequeados. La mayoría de la polución es producida vía, el método en que los datos son recaudados en el campo, remover éste tipo de polución casi siempre involucra procesos de reingeniería.

1.3.3 Enriquecimiento

Luego de que los datos son depurados podemos querer enriquecerlos la forma de agregar mas información a los datos, es mediante el cruzamiento de las bases de datos. En muchos países el acceso a algunas bases de datos adicionales esta disponible sobre bases comerciales, y ésto puede proveer información de diferente tipo, incluyendo: datos demográficos, el promedio de precios de casas y autos, tipos de seguros que tienen las personas, y así siguiendo. Las compañías pueden extraer datos que aportarán más información a los que ya existen para coordinar sus operaciones de marketing.

Actualmente, esta tarea no es posible de realizar en todos los casos, ya que existen dificultades legales para tener BDs adicionales, disponibles que abarquen la mayoría de los temas, por ejemplo, bases de datos sobre las historias clínicas que en ciertos aspectos de la medicina, es información reservada, a la que no se puede acceder. Las BDs adicionales existentes en el mercado pueden no ser útiles para nuestro objetivo en el proceso del descubrimiento.

1.3.4 Codificación

A veces podemos sacar la polución de los datos simplemente filtrando aquellos registros que la contienen. Por ejemplo, alguna información concerniente a dueños de autos o casas, es pobre para algunos individuos en la BD, si esa pobreza de información ésta distribuida al azar sobre la base, remover esos registros, no va a afectar al tipo de grupos que vayamos a encontrar, pudiéndolo realizar exitosamente. Por otro lado, es posible que haya alguna conexión no casual entre la pobreza de cierta información y cierto tipo de clientes, especialmente en situaciones donde el fraude puede estar involucrado, como en registros de seguros. Algunos clientes pueden dar deliberadamente información errónea, para obtener una cobertura de seguro que en otro caso no lo lograrían. Obviamente, en estos casos, remover información ciertamente afectará al tipo de patrones que encontraremos.

La tarea de codificación tiene como meta lograr datos valuables, es decir, categorizar la información de los campos dependiendo del objetivo en particular del proceso de descubrimiento.

1.3.5 Data Mining

Es un paso del proceso KDD, que consiste en aplicar análisis de datos y algoritmos de descubrimiento, que producen una enumeración particular de patrones sobre los datos.

El componente Data Mining (del proceso KDD) frecuentemente involucra repetidas aplicaciones iterativas de métodos particulares de Data Mining.

En éste paso se realiza el verdadero descubrimiento de un proceso KDD.

Esta etapa, consiste en aplicar técnicas de análisis de datos y algoritmos de descubrimiento, que encuentren relaciones intrínsecas en los datos.

Data mining es, en esencia, un conjunto de técnicas que permiten el acceso a datos que están ocultos en las BDs, es apropiado para obtener información segura y útil de grandes BDs la cual, no podría ser encontrada usando herramientas de lenguaje normalizado de consulta (SQL Standard Query Language) estándares.

1.3.6 Reporte

La última fase en el proceso de KDD es el reporte, que combina dos funciones diferentes:

- El análisis de los resultados de los algoritmos de reconocimiento de patrones
- La aplicación de los resultados de los algoritmos de reconocimiento de patrones a nuevos datos

No solo queremos inspeccionar que aprendimos, sino que podemos querer aplicar la información de clasificación y segmentación obtenida. En muchos casos, el reporte es realizado usando una herramienta de query tradicional para BDs . Actualmente, surgieron técnicas nuevas de visualización de datos, desde simples diagramas, bidimensionales hasta complejos ambientes interactivos.

1.4 Conclusión:

KDD se solapa con aprendizaje de máquina y reconocimiento de patrones en el estudio particular de técnicas y algoritmos de Data Mining, propuestos para modelar datos y extraer patrones. Se enfoca sobre aspectos de encontrar patrones comprensibles, que puedan ser interpretados como conocimiento útil o interesante, y pone fuerte énfasis sobre el trabajo con grandes conjuntos de datos del mundo real.

El objetivo unificador, es extraer conocimiento de alto nivel desde datos de bajo nivel en el contexto de grandes conjuntos de datos.

KDD ha evolucionado, y continua desarrollándose, desde la intersección de campos de investigación tales como aprendizaje de maquina, reconocimiento de patrones, bases de datos, estadística, inteligencia artificial, adquisición de conocimiento por sistemas expertos, visualización de datos, y alto desarrollo de cómputos.

2 Aspectos de Data Mining

2.1 Introducción

Existe una confusión acerca del significado exacto de los términos "Data Mining" y "KDD", muchos autores los consideran sinónimos. En la 1era. Conferencia internacional de KDD en Montreal en 1995, fue propuesto que el término KDD se emplee para describir aquellos procesos de extracción de conocimiento desde los datos. En éste contexto, conocimiento significa relaciones y patrones entre los elementos (datos). A partir de ésto se propuso que el término "Data Mining" debería usarse exclusivamente para el estado de descubrimiento de un proceso KDD. Por esto, presentamos en el capítulo anterior a Data Mining como uno de los pasos de un proceso KDD.

Este capítulo muestra que Data Mining no es sólo una simple técnica como lo es la idea de que hay más conocimiento oculto en los datos, de lo que ellos muestran en su superficie.

Cualquier técnica que ayude a extraer información de los datos, es útil, las técnicas de Data Mining forman un pequeño grupo heterogéneo. Diferentes técnicas son usadas para diferentes propósitos.

Un proceso de Data Mining puede ser iterativo, incluir una o más técnicas según el objetivo, es decir, hay metodologías a seguir que se conocerán aquí. También se veran los resultados de un proceso de Data Mining en relación al modelo aplicado y cómo evaluar su efectividad - utilidad en relación al objetivo planteado.

2.2 Data Mining - Qué es y Qué puede hacer ?

Una definición típica de Data Mining es la del Grupo Gartner:

"Data Mining es un proceso de descubrimiento de nuevas y significativas correlaciones, patrones y tendencias, examinando grandes cantidades de datos almacenados y usando técnicas de reconocimiento de patrones, de estadística y matemáticas."

Los objetivos de descubrir conocimiento están definidos por el uso prometido del sistema. Distinguimos dos tipos de objetivos: *Verificación,* donde el sistema es limitado a verificar las hipótesis de los usuarios; *Descubrimiento,* donde le sistema automáticamente encuentra nuevos patrones. Subdividimos nuevamente el objetivo de *Descubrimiento* en *Predicción,* donde el sistema encuentra patrones con el propósito de predecir el comportamiento futuro de algunas entidades; *Descripción,* donde el sistema encuentra patrones con el propósito de presentarle luego al usuario en forma entendible para él.

Aunque los límites, entre la predicción y descripción no están definidos (algunos de los modelos predictivos pueden ser descriptivos, para el grado en que ellos son entendibles y viceversa), la distinción es útil para el entendimiento de todo el objetivo

del descubrimiento. La importancia relativa de la predicción y la descripción para aplicaciones particulares de Data Mining puede variar considerablemente.

Sin embargo, en el contexto de un KDD, la descripción tiende a ser más importante que la predicción. En contraste para algunas aplicaciones de aprendizaje de máquina (Machine Learning) y reconocimiento de patrones donde la predicción es a menudo el objetivo principal.

Los objetivos de predicción, descripción y verificación son logrados mediante las siguientes tareas:

Clasificación

Estimación

Predicción

Agrupamiento por Afinidad

Clustering

Descripción

Ninguna herramienta o técnica de Data Mining es igualmente aplicable a todas las tareas.

Ahora se explicará cada una de las tareas, los modelos de Data Mining que las incluyen y qué técnicas pueden ser aplicadas a cada una.

2.2.1 Clasificación:

La clasificación, es la tarea más común de Data Mining. Clasificar parece ser una necesidad humana, para poder comunicarnos en el mundo, estamos constantemente clasificando, categorizando y separando en niveles

Clasificar consiste en examinar características de los nuevos objetos presentados, y asignarlos en una de las clases predefinidas dentro del conjunto de clases. Aquí los objetos a ser clasificados generalmente son registros en una base de datos y el hecho de clasificarlos consiste en modificar cada registro con un campo que contenga el código de clase de algún tipo.

La tarea de clasificación está caracterizada por una buena definición de clases, y la preparación de un conjunto de ejemplos preclasificados. La tarea es construir un modelo que pueda ser aplicado a un conjunto de datos sin clasificar para clasificarlos.

Ejemplos de tareas de clasificación:

- asignación de claves a artículos
- clasificación de aplicaciones en bajo, medio o alto riesgo
- determinar que números de teléfonos pertenecen a fax
- designación de trabajos sobre las bases de las descripciones de los mismos

En todos los ejemplos anteriores hay un número limitado de clases, y esperamos poder asignar cualquier registro a una u otra clase.

Herramientas que pueden ser bien aplicadas a ésta tarea son: árboles de decisión, razonamiento basado en memoria (vecino más cerca) y el análisis de links, son buenos en algunas circunstancias.

2.2.2 Estimación:

La clasificación trabaja con resultados discretos (distintos): sí o no, sarampión o varicela (resultados opuestos).

La estimación trabaja con resultados valuados continuamente. Dados algunos datos de entrada, usamos la estimación para presentar un valor para alguna variable desconocida continua, como: sueldo, altura, o el balance de una tarjeta de crédito.

En la práctica, la estimación es frecuentemente usada para representar la tarea de clasificación. Una empresa puede construir un modelo de clasificación colocando a todos sus clientes en una de las dos clases: esquiadores y no esquiadores, otra aproximación, es construir un modelo que asigne a cada cliente un puntaje tentativo de ski. Esto puede ser un valor desde 0 a 1, indicando la posibilidad estimada de que el cliente sea esquiador. La tarea de clasificación ahora es la más importante para establecer el puntaje inicial, cualquiera con un puntaje mayor o igual que el inicial es clasificado como esquiador y cualquiera con un puntaje menor es considerado como no esquiador.

La aproximación de la estimación tiene la gran ventaja de que los registros individuales ahora pueden ser ordenados por un puntaje. Para ver la importancia de ésto, imaginemos que la compañía de fabricación de botas de ski planea entregar por correo 500 000 piezas. Si se utiliza la clasificación y 1,5 millones son identificados como esquiadores, se puede simplificar tomando 500 000 clientes al azar del total. Sí por otro lado cada cliente tiene su puntaje se puede seleccionar a los 500 000 clientes con puntaje más alto.

Ejemplos de tareas de estimación:

- estimar el número de hijos en una familia
- estimar la entrada total de sueldos en una familia
- estimar el tiempo de vida de un cliente.

Las redes neuronales es una técnica apropiada para realizar tareas de estimación.

2.2.3 Predicción:

La predicción es como la clasificación o estimación, excepto que los registros son clasificados de acuerdo a un comportamiento futuro pronosticado o un valor futuro estimado. En la tarea de predicción la única forma de chequear la seguridad de la clasificación es esperar y ver.

Cualquiera de las técnicas usadas para la clasificación y estimación puede ser adaptadas para ser usadas en la predicción, usando ejemplos donde el valor de una variable a ser pronosticada ya es conocido, con datos históricos para esos ejemplos. Los datos históricos son usados para construir un modelo que explique el

comportamiento observado. Cuando este modelo es aplicado a entradas actuales, el resultado es una predicción sobre el comportamiento futuro.

La técnica de análisis de canasta de mercado, usada para descubrir cuales ítems son comúnmente compradas juntos, en un almacén, también puede ser adaptada a un modelo de compras o tendencias o acciones futuras que están implícitos en los datos actuales.

Ejemplos de tareas de predicción:

- predicción de cuales clientes no van a venir los siguientes seis meses.
- predicción de qué suscriptores de teléfono ordenarán un servicio agregado, como llamado en conferencia y/o contestador automático

Las siguientes técnicas son apropiadas para la tarea de predicción: análisis de canasta de mercado, razonamiento basado en memoria, árboles de decisión y redes neuronales artificiales. La elección de la técnica depende de la naturaleza de los datos de entrada, el tipo de valor a ser predecido y la importancia que trae aparejada la explicación de la predicción.

2.2.4 Agrupamiento por Afinidad:

La tarea de agrupamiento por afinidad, es para determinar que objetos van juntos. El prototipo de ejemplo es determinar que cosas van juntas en un carrito de supermercado, de aquí que se lo suele denominar análisis de canasta de mercado. Cadenas de minoristas pueden usar agrupamiento por afinidad para planear la ubicación de los ítems en las estanterías de almacén o catálogos, entonces los ítems que frecuentemente se compran juntos van a ser vistos juntos.

Esta tarea puede ser usada para identificar oportunidades de ventas cruzadas y para diseñar paquetes atractivos o grupos de productos y servicios. Si dos ítems, por ejemplo comida y cuchas para gatos ocurren lo suficientemente frecuentes, podemos generar dos reglas de asociación:

"Las personas que compran comidas para gatos, también compran una cucha con probabilidad P1."

"Las personas que compran una cucha para gatos, también van a comprar comida para gatos con probabilidad P2."

La técnica de análisis de canasta de mercado y reglas de asociación es apropiada para la tarea de agrupamiento por afinidad.

2.2.5 Clustering:

La tarea de clustering es la segmentación de una población heterogénea en un numero de subgrupos (o cluster) homogéneos. Lo que la distingue de la clasificación, es que el clustering no confía en las clases predefinidas.

En la clasificación, la población es subdividida por la asignación de cada elemento o registro a una clase predefinida en base al modelo desarrollado a través de ejemplos preclasificados. En el clustering, no hay clases predefinidas ni ejemplos, los registros

son agrupados juntos en base a la similitud. Esto ayuda para determinar que significados, si hay alguno, agregar a los clusters resultantes.

Los clusters de síntomas pueden indicar diferentes enfermedades, los clusters de atributos de hojas y núcleos pueden indicar diferentes especies de plantas.

Frecuentemente la tarea de clustering puede ser el primer paso en la segmentación de un mercado: en vez de tratar de obtener todas las reglas para saber "a que tipos de promoción responden mejor los clientes", podemos primero dividir a los clientes en clusters o personas con hábitos similares de compras, y luego preguntar "que tipo de promisión funciona mejor para cada cluster".

2.2.6 Descripción:

A veces el propósito de Data Mining es simplemente describir que está sucediendo en una base de datos para incrementar el entendimiento de personas, productos, o procesos que producen datos. Frecuentemente, una descripción suficientemente buena de un comportamiento también sugerirá una buena explicación para éste. Además una buena descripción sugiere donde empezar a mirar para una explicación.

La famosa división de la política norteamericana es un ejemplo de como una simple descripción, "las mujeres defienden a los demócratas en mayor número que los hombres", puede provocar un estudio de gran interés para periodistas, sociólogos, economistas y políticos para nombrar candidatos en un puesto político.

La técnica de análisis de canasta de mercado es puramente descriptiva. Otras técnicas, por ejemplo, las redes neuronales proveen poca descripción.

2.3 Modelos de Data Mining

Hay dos estilos básicos de Data Mining. El primero, *Testeo de Hipótesis*, es una aproximación *top-down* que intenta sustentar o desaprobar ideas preconcebidas. El segundo, *Descubrir de Conocimiento*, es una aproximación *bottom-up* que comienza con los datos, y los procesa para expresar algo que no se conozca.

Se pueden resolver problemas de Data Mining desde ambas metodologías, balanceándose entre las dos aproximaciones.

Usando la aproximación *top-down*, se piensa en posibles explicaciones para el comportamiento observado y esas hipótesis tientan los datos a ser analizados Utilizando la otra, *bottom-up*, se deja a los datos sugerir nuevas hipótesis a testear.

2.3.1 Testeo de Hipótesis:

El testeo de hipótesis es en lo que los científicos y estadistas gastan su vida haciendo. Una hipótesis es una explicación propuesta cuya validez puede ser testeada. El testeo de la validez de una hipótesis se realiza con un análisis de los datos que simplemente fueron recolectados por observaciones o generados por un experimento.

El proceso de testeo de hipótesis tienen varios pasos:

a.- Generar buenas ideas (hipótesis).

b.- Determinar que datos permiten que estas hipótesis sean testeadas.

c.- Localizar los datos.

d.- Preparar los datos para el análisis.

e.- Construir modelos de computación basados en los datos.

f.- Evaluar los modelos de computación para confirmar o refutar hipótesis.

Explicación breve de cada paso:

a.- Generar buenas ideas (Hipótesis)

Para ésto se debe tener claramente definido el problema, especialmente si es algo que no fue reconocido previamente como tal. La clave es tomar información desde un gran espectro de la organización, es decir desde varias partes de la empresa.

b.- Determinar que datos permiten que las hipótesis sean testeadas

Una vez que las hipótesis están generadas o se piensan que están correctas, serán evaluadas por su predisposición a ser testeadas. Es decir, algunas hipótesis pueden ser testeadas por simples queries que existan en el soporte de decisiones de bases de datos, otras requerirán información recolectada desde sistemas operacionales, investigaciones, etc.

Para cada hipótesis se debe generar una lista de datos requeridos.

c.- Localizar los datos

Data Mining requiere datos. En el mejor de los casos los datos necesitados están residentes en un Data Warehouse corporativo. Es más común que están disponibles en varios sistemas, en formatos operacionales incompatibles, corriendo en diferentes sistemas operativos y siendo accedidos a través de herramientas de escritorio incompatibles.

El origen de los datos que son útiles y están disponibles varían de problema en problema y de industria en industria.

d.- Preparar los datos para el análisis

No es muy común que los datos disponibles sean recolectados y guardados con un propósito de data mining. Con lo cual se necesitan transformaciones especificas para los datos requeridos por la técnica en particular de data mining a utilizar.

Por ejemplo, algunas técnicas necesitan tener todas las variables continuas acotadas en rangos. Otras requieren todos los valores normalizados en un rango de cero a uno, etc.

e.- Construir modelos de computación basados en los datos

Se debe empezar con un modelo mental. Antes de llevar el modelo mental a computacional, se analiza si los datos disponibles pueden intervenir en el modelo mental

f.- Evaluar el modelo computacional

Finalmente, se aplica el modelo computacional a los datos reales para ver si alguna hipótesis se confirma. Dependiendo de las hipótesis, y la naturaleza del modelo, puede significar la interpretación de un valor simple, u observar una colección de reglas de asociación generadas por un análisis de canasta de mercado, o determinando el significado de correlación encontrado por un modelo de regresión.

2.3.2 Descubrir de conocimiento

Descubrimiento de conocimiento es el estilo de data mining que más interés genera. Es fácil imaginar que ahora podemos simplemente dejar que un programa "inteligente" se pierda en los datos y vuelva con información que se puede aplicar para hacer dinero, ganar clientes, influenciar en la legislación, etc.

El aprendizaje indirecto es el objetivo desde hace tiempo de los investigadores del campo de inteligencia artificial en la disciplina aprendizaje de máquina.

En el mundo real, descubrir patrones significativos es útil, pero es un trabajo difícil.

2.3.2.1 Directo vs. Indirecto

El descubrimiento puede ser directo o indirecto. En el descubrimiento de conocimiento directo, la tarea es explicar el valor de algunos campos en particular (edad, sueldo, etc.) con respecto a todos los otros campos. Se selecciona un campo destino y se dirige a la computadora para que explique cómo estimarlo, clasificarlo, o predecirlo. En el descubrimiento de conocimiento indirecto, no hay campos destino. Simplemente se pregunta a la computadora que patrones identifica en los datos que puedan ser significativos. Este último complementa la experiencia de un analista de negocios.

Usamos el descubrimiento indirecto para reconocer relaciones en los datos, y el directo para explicar esas relaciones una vez que fueron encontradas.

El aprendizaje indirecto puede ser muy útil, especialmente clustering y agrupamiento por afinidad, pero la gran mayoría de trabajos de descubrimiento de conocimiento es mucho más directo.

Los programas de descubrimiento de conocimiento deben poder reconocer lo que están buscando, y es más fácil que una red neuronal o un árbol de decisión trate de reconocer una "transferencia de balance de una cuenta" que trate de reconocer "algo interesante".

2.3.2.2 Descubrimiento de conocimiento directo

El descubrimiento de conocimiento directo está orientado al objetivo. Hay un campo especifico cuyo valor se quiere predecir, o asignar cada registro a un conjunto de clases fijas, o explorar una relación especifica. El data mining directo trata de contestar preguntas como estas:

•Qué productos muestran un crecimiento de venta, cuando la venta del queso crema disminuye ?

•Quién es el que puede adquirir un seguro de créditos ?

•Qué ganancia se predice de nuevos clientes ?

•Qué avisos comerciales van a aparecer en la ventana web, cuando mjab@data-miners.com ejecute una búsqueda por clave ?

•Cuáles de las ciudades potenciales para una nueva cadena de negocios minoristas darán mejores ganancias ?

Los pasos en el proceso de descubrimiento de conocimiento directo son:

a.- Identificar orígenes de datos preclasificados

b.- Preparar los datos para el análisis

c.- Construir y probar el modelo computacional

d.- Evaluar el modelo computacional

a.- Identificar orígenes de datos preclasificados

El descubrimiento de conocimiento se basa en la premisa que la respuesta a preguntas del tipo de las anteriormente mencionadas pueden encontrarse examinando los datos sobre lo sucedido en el pasado. Entonces el primer requerimiento para el éxito del descubrimiento de conocimiento es buenos datos.

El origen ideal para los datos es la existencia de un data warehouse. El problema es que muchas veces no existe y por diversos motivos los datos históricos no son generalmente guardados por mucho tiempo, sólo se conservan los de determinado tiempo atrás.

En el descubrimiento de conocimiento directo, se usa el pasado para construir un modelo del futuro. Si se quiere distinguir personas que comúnmente tienden a no cumplir los pagos de un préstamo de las personas que si lo hacen, se miran millones de ejemplos de personas de cada clase para construir el modelo que distinga una de otra.

Implícito en el modelo está la idea de poder explicar que sucedió en el pasado. Para aprender de errores pasados, primero se debe reconocer que se cometieron. Por ejemplo, si una compañía trata de usar "descubrimiento de conocimiento directo", para construir un modelo de reclamos de seguros fraudulentos, porque sospechan que algunos pueden serlos, y no saben cuales de ellos, sin un conjunto de reclamos de seguros claramente marcados como tales o legítimos, es imposible aplicar esta técnica.

b.- Preparar los datos para el análisis

Los mismos comentarios generales sobre la preparación de los datos hecho en la sección anterior de testeo de hipótesis, son aplicables en éste paso. Además como el descubrimiento de conocimiento confía en herramientas para encontrar patrones automáticamente, muchas veces se necesita aumentar los datos con campos adicionales derivados de los ya existentes, de tal forma que éstos pueden ser obvios para quienes implementan un modelo de testeo de hipótesis, pero difícilmente sean considerados por un software simple. Esto se hace especialmente verdadero cuando por ejemplo se quiere ver relaciones entre muchos campos. Agregar campos que representan relaciones en los datos, que se conocen desde la experiencia, son muy

importantes porque pueden incrementar la oportunidad de que el proceso de descubrimiento de conocimiento, de resultados útiles.

c.- Construir y probar el modelo computacional

Los detalles de éste paso varían de técnica en técnica. En términos generales, aquí es donde esta la mayor parte del trabajo de creación del modelo. El conjunto de prueba se usa para generar una explicación de una variable destino independiente en términos de una variable de entrada independiente. Esta explicación puede tomar forma de neuronal, de árbol de decisión, o alguna otra representación de relaciones entre el campo que se trata de estimar, clasificar, o predecir y los otros campos de la base de datos.

d.- Evaluación del modelo

En este punto, ya se utiliza el conjunto de datos de prueba para optimizar el modelo, pero todavía no sabemos que tan bien lo hemos hecho como para tener expectativas de su trabajo sobre los datos que aún no ha visto. Usando el conjunto de datos de prueba se eliminaron reglas o relaciones que dependen enteramente de la idiosincrasia de este conjunto de datos, pero es posible que se haya usado algunas pequeñas desviaciones del conjunto. Para confiar en la estimación de la ejecución del modelo, se necesita aplicarlo a la colección final de registros preclasificados (reservados), denominada conjunto de evaluación. El grado de error sobre el conjunto de evaluación es una buena predicción del grado de error sobre los datos no vistos.

2.3.2.3 El descubrimiento de conocimiento indirecto

El proceso previamente descripto es llamado descubrimiento de conocimiento directo. Es el tipo de descubrimiento de conocimiento más común. Se caracteriza por la presencia de un único campo destino, cuyo valor va a ser predecido en termino de los otros campos de la base de datos.

El descubrimiento de conocimiento indirecto es diferente, no hay un campo destino. La técnica de data mining simplemente se "pierde" en los datos con la esperanza de descubrir estructuras significativas. Una comúnmente utilizada para este tipo de descubrimiento es la técnica de análisis de canasta de mercado que pregunta: "que ítems se venden juntos" en vez de "que ítems se venden con el café". Otra es el clustering, donde grupos de registros son asignados al mismo cluster, si tienen algo en común.

Un buen ejemplo de descubrimiento de conocimiento indirecto, es el estudio de la segmentación del mercado de clientes que realizan llamadas de larga distancia. Tomados como un todo, los datos mostraron claramente patrones uniformes de uso, con variaciones predecibles entre días de semana, fines de semana y ciertos feriados. Pero, una vez que el grupo de datos fue dividido en segmentos, usando la técnica indirecta clustering, se hizo claro que comportamiento agregado de la población de clientes, ocultaba algunos patrones muy interesantes en el comportamiento de varios subgrupos.

Un grupo, en particular, tiene ambos perfiles: de uso y lucrativo. El uso de llamadas de larga distancia de este grupo permanece alto y constante, la mayor parte del año, pero luego se precipita para algunos meses. Aún más, aunque el grupo como un todo, tiene un nivel bajo en varios modelos de confiabilidad en el pago, ellos son ambas cosas, confiables y no son morosos. Examinando más de cerca este grupo, están fuera de él el grupo formado por estudiantes universitarios, no tienen entrada

fija, ni historial de pagos, pero suelen llamar mucho a los padres, lo que sucede es que llaman por cobrar.

El proceso de descubrimiento de conocimiento indirecto tiene los siguientes pasos:

a.- Identificar orígenes de datos

b.- Preparar los datos para el análisis

c.- Construir y preparar el modelo computacional

d.- Aplicar el modelo a nuevos datos

e.- Identificar destinos potenciales para descubrimiento de conocimiento directo.

f.- Generar nuevas hipótesis para testear.

Los pasos desde (a) a (b) son los mismos que en el descubrimiento de conocimiento directo. Los pasos (e) y (f) reflejan el hecho de que usualmente el descubrimiento de conocimiento indirecto, es el preludio de mas investigaciones a través de técnicas directas.

e.- Identificar destinos potenciales para descubrimiento de conocimiento directo.

El descubrimiento de conocimiento indirecto, es un gran camino para generar ideas que pueden ser verificadas usando más modelos directos. Un reporte de canasta de mercado está relacionado a preguntas sobre: porqué los productos que se venden juntos, están juntos?, quién compra combinaciones particulares de productos?, y cuándo tienden a ser hechas las ventas?. Estas preguntas se dirigen al descubrimiento de conocimiento directo. Los nuevos clusters o agrupamientos de productos descubiertos se transforman en variables destinos en la necesidad de explicaciones.

f.- Generar nuevas hipótesis para testear.

Una vez que un programa de clustering haya hecho su trabajo, se necesita estudiar los cluster resultantes para ver que tienen en común y como se puede dar un buen uso a esa información.

2.4 Efectividad de Data Mining

Data mining es costoso, requiere un importante esfuerzo en la recolección y preparación de los datos, la integración del software, los problemas de formulación, la construcción del modelo y el análisis. Cómo se puede asegurar que los resultados valen todo el tiempo, el dinero, y el esfuerzo invertido?.

Para contestar esta pregunta, se necesita poder responder otras tres:

- Cuál es el objetivo del ejercicio de data mining?
- Cuán bien logrado fue el objetivo?
- Cuál fue el esfuerzo (tiempo, dinero, esfuerzo) invertido para alcanzar el objetivo

Aquí se examinarán cada una de estas preguntas. Data mining puede ser usado en diferentes formas para diferentes propósitos. La selección de la herramienta y el método de data mining está en gran parte determinado por lo que se quiere lograr.

Cada método de data mining tiene su propio vocabulario y su forma de evaluar seguridad y rendimiento. Estos son utilizados por los analistas para evaluar el desarrollo de herramientas de data mining. Puede ser difícil hacer un juicio del grado de error, medida de confianza, y desviaciones estándares usadas para medir los modelos de data mining, especialmente cuando se trata de comparar modelos de diferentes tipos.

Se introduce el concepto de ***Lift*** , una medida de cuánto se debe extender el modelo para lograr mejores resultados.

2.4.1 Objetivos:

En los proyectos de data mining con objetivos extensos, y en términos generales como:

Obtener el comportamiento de los clientes

Descubrir patrones significativos en los datos

Aprender algo interesante

son objetivos importantes, pero aunque se hayan logrado son difíciles de medir.

Cuando sea posible, los objetivos extensos y/o generales deben ser descompuestos en otros más específicos para que sea más fácil probar el progreso en la tarea de lograr los objetivos.

Por ejemplo, el objetivo:

Obtener el comportamiento de los clientes

puede ser especificado en los siguientes objetivos:

Identificar a los clientes que seguramente no renovaran su suscripción

Ordenar a todos los clientes por su predisposición a la práctica de ski

Listar los productos que se vieron adversamente afectados si baja el valor del vino y la cerveza.

2.4.2 Descripción o Predicción

Los objetivos de data mining pueden ser descriptivos o predictivos. Si se consideran dos conjuntos de reglas derivadas de una gran base de datos. El primero incluye cuatro reglas, cada una de ellas es una conjunción de simples testeos, verdadero o falso, en campos individuales. El segundo conjunto incluye cincuenta reglas, cada una es una combinación lineal de docenas de campos. El primer conjunto de reglas clasifica correctamente el 70% de los casos, el segundo conjunto el 72%. Cuál es el mejor?. La respuesta depende enteramente del objetivo del ejercicio de data mining.

En este capítulo identificamos seis tipos de tareas de data mining, cada una de ellas puede ser descriptiva o predictiva. Una tarea descriptiva es aquella cuyo objetivo es entender, explicar, o descubrir conocimiento.

Muchas definiciones de data mining se enfocan en las tareas descriptivas, - encontrar patrones interesantes.

En contraste, muchas de las aplicaciones actuales de data mining son para clasificar y predecir.

Hasta cuando el objetivo de data mining es predictivo, puede ser importante que el modelo usado sea lo suficientemente descriptivo para ser claro el por que una predicción particular fue hecha.

2.4.3 Modelos de medición

Cuán preciso es el modelo?

Cuán bien describe los datos observados?

Cuánta confianza se le puede dar a las predicciones del modelo?

Cuán comprensible es modelo?

Hay formas objetivas de contestar estas preguntas.

2.4.3.1 El todo o las partes

Se necesita distinguir entre el modelo como un todo y toda clasificación o predicción particular. En el campo académico de aprendizaje de máquina - el orígen de muchos de los algoritmos usados para data mining - los investigadores tienen el objetivo de generar modelos que puedan ser entendibles en su totalidad.

Un modelo que es fácil de entender se dice que tiene una buena "adaptación mental". Con el interés de obtener la mejor "adaptación mental", estos investigadores siempre prefieren modelos que consisten de pocas reglas, a los que contienen muchas, aunque luego estos últimos son más precisos. En el contexto de los negocios, se está más interesado en la explicación de cada predicción o clasificación individual que en la comprensión del modelo común todo.

Cuando se mide la certeza de un modelo predictivo, se interesan en ambos: la certeza del modelo como un todo (el porcentaje de registros clasificados correctamente) y la certeza de las predicciones individuales.

Dos modelos con la misma exactitud pueden tener algunos niveles de variación diferentes entre predicciones individuales.

2.4.3.2 Medición de modelos descriptivos

La regla: "si la casa está en La Plata entonces la calefacción es a gas", es más descriptiva que la regla: "si el codigo-área = 207 o el codigo-área = 603 o el codigo-área = 802 o el codigo-área = 413 o el codigo-área = 503 o el codigo-areá- = 617 entonces la calefacción de la casa es a gas". Aunque el resultado de ambas reglas sea equivalente, la primera es más expresiva.

Se puede suponer que el grado de poder de expresión de una regla es puramente subjetivo, pero existe de hecho, una medida teórica que puede ser usada para determinarlo. Esta medida es llamada: ***longitud mínima de descripción*** (MDL-Minimum Description Length).

El MDL para un modelo, es el número de bits que lleva la codificación de ambas: la regla y la lista de todas las excepciones a la regla. La que requiera menor número de bits es la mejor regla. Algunas herramientas de data mining usan MDL para decidir que conjuntos de regla tomar y que conjuntos de reglas dejar.

2.4.3.3 Medición de modelos predictivos

Los modelos predictivos son evaluados sobre la certeza de sus predicciones para los datos que no habían sido vistos previamente. Diferentes tareas de data mining se utilizan para evaluar de diferente forma el desarrollo del modelo como un todo y diferentes formas de juzgar la probabilidad de que el modelo produzca resultados certeros para cualquier registro particular.

2.4.3.4 Medición de clasificación y predicción

Para tareas de clasificación y predicción, la certeza es medida en términos de grado de error. El grado de error es simplemente el porcentaje de registros clasificados incorrectamente.

El grado de error de clasificación que el modelo toma del conjunto de datos preclasificados para la evaluación, es una estimación del grado de error esperado cuando clasifiquemos nuevos registros. Por supuesto, este procedimiento sólo es válido si el conjunto de evaluación es una muestra representativa de la gran población

Un método recomendado para establecer el grado de error para un modelo, es medirlo con un conjunto de evaluación tomado de la misma población como el conjunto de testeo, pero entre ellos disjunto. En el interés de conservar los registros preclasificados, que en algunos ambientes pueden ser escasos, hay algunas aproximaciones para estimar el grado de error de un modelo aún cuando los únicos registros disponibles sean los que están en el conjunto de testeo.

2.4.3.5 Resultados de la medición

Cada técnica de data mining tiene su propio conjunto de criterios de evaluación, lo cual hace difícil comparar modelos distintos. Hay una forma de comparar modelos sin tomar su técnica de trabajo. Los modelos predictivos, ya sea usando redes neuronales, arboles de decisión, etc., todos son creados para cumplir con una tarea, entonces, por que no juzgarlos por sus habilidades para clasificar, estimar y predecir ?

Lift:

La forma más común de comparar el desarrollo de modelos de clasificación es usando la relación llamada ***Lift*** . Esta técnica puede ser adaptada para comparar modelos diseñados para otras tareas. Lo que ***Lift*** mide actualmente, es el cambio en la concentración de una clase en particular cuando el modelo es usado para seleccionar un ejemplo con un fin determinado del total de la población.

Lift = P (clase $_t$ | ejemplo) | P (clase $_t$ | población total)

Para finalizar, la única medida que realmente importa es el resultado de la investigación. Medidas de lift sobre un conjunto de evaluación puede ayudar a elegir el modelo correcto. Modelos útiles basados en Lift ayudan a decidir como aplicar los resultados del modelo. Pero es muy importante medir este tipo de cosas sobre los campos también.

3 Técnicas de Data Mining

3.1 Introducción

Data Mining dentro de un proceso KDD, genera información beneficiosa y procesable.

Lift y las mediciones comentadas en el capítulo anterior dan medidas de efectividad de la información y acciones retornadas por el proceso de Data Mining.

Las tareas descriptas -clasificación, estimación, predicción, agrupamiento por afinidad, clustering y descripción - transforman los datos en información para satisfacer necesidades continuas de una organización.

Aquí veremos el "como", de las técnicas de Data Mining, y se introducirán siete técnicas especificas, las más comúnmente usadas y ejemplos de su uso.

Una única técnica de Data Mining, no resuelve todos los problemas de Data Mining. Por lo general, para proveer la mejor aproximación para resolver problemas de Data Mining son necesarias una variedad de técnicas.

Este capítulo provee información de las técnicas desde la perspectiva de problemas y desafíos que surgen a través de Data Mining.

3.2 Modelos de Análisis de Datos

Las técnicas de Data Mining muestran en su historia la influencia de otras disciplinas. Los algoritmos genéticos y redes neuronales descienden de intentar modelar procesos biológicos sobre computadoras.

Memoria basada en razonamiento es una técnica que desciende directamente del campo de la inteligencia artificial, y el análisis de enlaces proviene de la teoría de grados y su aplicación a estructuras de datos en la ciencia de la computación.

No obstante, como técnica completa, la estadística incluye muchas de estas técnicas, y es la más clásica de ellas.

La estadística fue originada para dar sentido a observaciones hechas sobre el mundo, generalmente en el nombre de las ciencias naturales o políticas. Estadísticas descriptivas dan información general sobre observaciones – cuál es el promedio, el valor medio, cuáles son los valores observados, cuál es la distribución de los valores. El análisis de regresión se refiere a las técnicas usadas para interpolar y extrapolar esas observaciones. La técnica de regresión más familiar es indudablemente la regresión lineal, que intenta unir con una línea los datos observados.

Uno de los mayores problemas con la regresión es que trabaja mejor cuando la forma de la solución es cocida como cuando se espera que la solución se vea, como cuando se espera que la solución se vea como una línea. El análisis de correlación es otra técnica importante que describe cuando diferentes observaciones ocurren juntas. Un ejemplo de correlación puede ser que personas con grandes ingresos tienden a tener grandes prestamos hipotecarios, aunque hay mejores ejemplos. Un

problema con el análisis de correlación es que casi siempre se produce resultados triviales que ya son conocidos.

El muestreo es una técnica muy importante, usada extensamente en estadística (y un poco menos en Data Mining) para reducir el tamaño de los datos sobre los cuales se va a trabajar.

Hoy, la estadística tiene algunas ventajas que no comparten con las técnicas de Data Mining. Hay gran cantidad de personas calificadas, que estudiaron estadística y su aplicación a virtualmente todas las áreas. Los software de estadísticas están disponibles, corren sobre muchas plataformas y tienen un creciente desarrollo sobre plataformas paralelas.

En algunos casos, los resultados estadísticos resultan tan buenos como cualquiera de las técnicas que se van a describir.

La estadística es muy útil pero no resuelve todos los problemas de Data Mining. El muestreo reduce el tamaño del conjunto de datos, lo que puede perder importantes subconjuntos de datos, perdiéndose quizás importantes oportunidades. Regresiones no son de propósito general, como las técnicas de Data Mining, porque pone énfasis en funciones continuas y formas ya conocidas. La complejidad computacional de las aproximaciones estadísticas no crecen bien con tamaños grandes de datos.

3.2.1 Modelos

Un modelo puede producir uno o más valores de salida para un conjunto de entrada dado. El análisis de los datos es siempre el proceso de construcción de un modelo apropiado para los datos. Una regresión lineal construye un modelo que es una línea, que tiene la forma: aX + bY + C = 0; donde a b y c son parámetros del modelo: y X e Y son variables. Para un valor dado X, se puede usar la línea para estimar el valor de Y. Este tipo de modelo es uno de los modelos disponibles más simples.

No solo porque el modelo exista, significa que provee resultados precisos. Para cualquier conjunto de puntos dado, hay alguna línea que une mejor los puntos, aún cuando no haya ninguna relación lineal entre los valores. Hay buenos y malos modelos, medir los resultados de uno es un paso critico, tanto su uso como su desarrollo.

El modelo lineal descripto es un ejemplo de regresión - encontrar la mejor forma de una curva para unir todos los puntos. Un modelo es extenso, y puede ser usado para clustering, clasificación, y análisis de time-series. Los modelos proveen un lenguaje común para hablar sobre Data Mining.

Un modelo de clasificación toma un nuevo registro y le asigna una clasificación existente, puede también darle una nueva clasificación, una probabilidad de correctitud, y otra información, dependiendo de la naturaleza del mismo. Un modelo predictivo es similar a uno de clasificación, excepto por la salida que no está limitada a un número de clases. Un modelo de clustering toma muchos registros y retorna, en el mejor de los casos, a un pequeño grupo de clusters. Estos clusters frecuentemente pueden ser luego aplicados a nuevos registros, para crear un modelo de clasificación. Un modelo "time-series" es como un modelo predictivo o de clasificación, excepto por el dominio que incluye datos por tiempos (over time).

Cuando un modelo es creado, la entrada generalmente está claramente especificada. Los registros de un file que tiene información sobre clientes, es un ejemplo típico del dominio que puede ser útil para un modelo predictivo. En el mundo

real, preparar los datos desde sistemas operacionales dispares para el dominio del modelo, es siempre más exigente que crear el modelo en si mismo. Las estradas al modelo puede afectar en la elección de la técnica: Para problemas de medicina con muchas variables de entrada continuas, las técnicas estadísticas de regresión trabajan muy bien. Cuando las entradas tienen muchas variables categóricas, los árboles de decisión a menudo trabajan bien. Cuando las relaciones entre las entradas y la salida es difícil de representar, las redes neuronales son la técnica frecuentemente elegida.

La salida del modelo es por lo general especificada anticipadamente. Frecuentemente son categorías, como si un cliente responderá a un "mailing", o una variable continua, como el máximo que se le puede extender un crédito a una persona. En los modelos de clustering, la salida especifica un cluster, pero éste puede no tener significado intrínsecamente. Entender el cluster requiere investigarlo y analizarlo, utilizando otras técnicas. Un modelo, generalmente, provee un grado de confianza en sus salidas. Esto es muy útil para determinar cuando aplicar los resultados del modelo.

Cuando creamos modelos para Data Mining, hay algunas cosas útiles que se debe tener en cuenta:

- uno de los peligros de los modelos es el *"underfitting"* o *" overfitting"* de los datos.
- data mining directo o indirecto usan modelos, pero en diferentes formas
- algunos modelos explican lo que realizan mejor que otros
- algunos modelos son más fáciles de usar que otros

Aquí están discutidos en más detalle:

3.2.1.1 Underfitting y Overfitting

Muchas veces, los modelos no trabajan bien. Dos causas comunes son el "underfitting" (escases) y "overfitting" (exceso) de los datos. Los mapas de rutas proveen un ejemplo familiar para ilustrar estos problemas. Algunos mapas de rutas son muy detallados, incluyen cada calle en un área relativamente pequeña, como una ciudad, con el sentido de ellas, muestran edificios y el nombre de los parques. Otros mapas son mas generales, cubren las carreteras o rutas más importantes en un área extensa, como una provincia. Cuál es el mejor mapa para usar?. Si se necesita viajar una gran distancia, puede resultar dificultoso usar el mapa más detallado, éste mapa "overfits" la información, la ruta que se necesita está allí, pero es difícil ubicarla entre todos los detalles, por otro lado, si se quiere buscar una calle particular, puede no estar en el mapa general, este es un ejemplo de "underfitting" de los datos, el mapa es muy general y no contiene la información que se necesita (la calle especifica) o no la describe bien.

Estos conceptos se aplican directamente a los modelos. El "overfitting" de los datos es cuando el modelo memoriza los datos y predice resultados basados en la idiosincrasia de éstos particularmente usados, por una variedad de razones: si el conjunto de datos es muy pequeño, algunas modelizaciones de técnicas realmente memorizan los datos, un ejemplo trivial de esto puede ser correr una técnica de regresión lineal para exactamente dos puntos. La regresión encuentra la línea que conecta los dos puntos, pero no comienza con los datos suficientes para determinar si la misma es útil.

También puede haber "overfitting" cuando un campo pronosticado es redundante, es decir, otro campo o combinaciones de campos tienen la misma información que el campo pronosticado, tal vez de otra forma.

El "underfitting" ocurre cuando el modelo falla en encontrar patrones de interés en los datos, es común cuando aplicamos técnicas estadísticas a los mismos, siendo una de las causa la eliminación de variables que tienen poder predictivo, pero no están incluidas en el modelo, otra que la técnica simplemente no trabaja bien para los datos en cuestión.

3.2.1.2 Directo vs. Indirecto

En el capítulo anterior ya se describieron las diferencias entre data mining directo o indirecto, desde la perspectiva de aplicar técnicas de Data Mining, las diferencias ocurren al crear el modelo. En el directo, las salidas del modelo son especificadas antes de créalo, el mismo se "entrena" con ejemplos donde los resultados son conocidos, en el indirecto, determina la salida, ayudando a los analistas, que es lo interesante de los resultados. En ambos casos, el modelo resultante puede ser aplicado a otros datos.

3.2.1.3 Explicabilidad

Para algunos propósitos, conocer porque un modelo en particular produce cierto resultado no es importante, pero en otros casos puede ser esclarecedor e importante. Algunos tipos de modelos son mas fáciles de entender que otros.

Por ejemplo, árboles de decisión y el análisis de canasta de mercado producen un conjunto claro de reglas. En el otro extremo, las técnicas de redes neuronales y clustering proveen una pequeña idea de porque un modelo en particular hace lo que hace.

3.2.1.4 Fácil de Aplicar

Similar a la explicabilidad es la facilidad de aplicar el modelo a registros nuevos. Si los datos son almacenados en una base de datos relacional, entonces el modelo que es implementado usando sentencias SQL es preferible a otro que requiere exportar datos.

3.3 Técnicas y Tareas

3.3.1 Análisis de Canasta de Mercado

El análisis de Canasta de Mercado es una forma de clustering usada para encontrar grupos de ítems que tienden a ocurrir juntos en una transacción. El modelo que construye, da la posibilidad de que diferentes productos sean comprados juntos y puede ser expresado como reglas, está fuertemente relacionado con la industria al por menor, donde la información sobre productos adquiridos juntos, puede ser los únicos datos disponibles para buscar patrones de clientes.

Como la técnica de clustering, el análisis de canasta de mercado es útil en problemas donde se desea saber que ítems aparecen juntos o en una secuencia en particular. Los resultados son a menudo altamente procesables porque incluyen los ítems específicos. La información resultante puede ser usada para muchos propósitos, como planificar la disposición del almacén, limitar a un producto del conjunto que se adquieren juntos, armar ofertas de grupos de productos, etc.

Esta técnica puede ser adaptada para usar datos históricos y demográficos.

3.3.2 Razonamiento basado en Memoria (MBR)

MBR es una técnica de data mining directa que usa instancias conocidas como un modelo para hacer predicciones sobre instancias no conocidas. MBR mira los vecinos más cercanos de las instancias conocidas y combina sus valores para asignar valores de clasificación o predicción. Por ejemplo, se trata de mantener una base de datos de demandas las cuales se pueden actualizar después de una investigación. Si se quiere determinar si una nueva demanda amerita una investigación, se puede encontrar demandas similares - vecinos - en la bases de datos y tomar la decisión: "realizar más investigación " o "pagar inmediatamente", basada en el estado de sus vecinos. La distancia a los vecinos da una medida de correctitud del resultado.

Una de las ventajas más importantes de MBR es la habilidad de correr virtualmente sobre cualquier origen de datos, aún sin que éstos se modifiquen. Hay dos elementos claves en MBR son: la función distancia, usada para encontrar los vecinos más cercanos y la función combinación, que combina los valores de los más cercanos para hacer una predicción. Para algunos propósitos, estas funciones pueden ser expresadas como sentencias SQL en una base de datos relacional y correr directamente dentro de la base de datos para propósitos de mining, aunque posiblemente, usarla tienda a ser ineficiente. En otros casos MBR puede estar basado en tipos de datos más complejos, como texto e imágenes, cuando las funciones de distancia estén disponibles en esos dominios.

Otra ventaja importante de MBR es la habilidad de aprender sobre nuevas clasificaciones, sólo por introducir nuevas instancias en la base de datos. Una vez que las funciones correctas de distancia y combinación fueron encontradas, ellas tienden a permanecer estables, aún cuando nuevas instancias para nuevas categorías son incorporadas a los datos conocidos. Es fácil la incorporación de cambios al dominio y separación de rangos en MBR, más que en las otras técnicas, que necesitan ser readaptadas para incorporar nueva información.

3.3.3 Detección de Cluster

La detección de cluster es la construcción de modelos que encuentran registros de datos que son similares entre ellos, los grupos de similitudes son llamados clusters. Esto es inherente al data mining indirecto, ya que el objetivo es encontrar similitudes previamente desconocidas en los datos. Hay muchas técnicas para encontrar clusters, incluyendo métodos geométricos, estadísticos y redes neuronales.

El clustering de datos es un muy buen camino para empezar cualquier análisis de datos. La similitudes de los clusters pueden proveer el punto de comienzo para saber que hay en los datos y para tener una idea de cual es su mejor uso.

3.3.4 Análisis de Links

El análisis de links (enlaces) pone atención en las relaciones entre registros, para descubrir modelos basados en patrones de las relaciones. Es una aplicación de la teoría de grafos construida para data mining. Un área natural para la aplicación del análisis de links es en las telecomunicaciones. Cada llamada telefónica relaciona (enlaza) un cliente con alguien más (potencialmente otro cliente). La información de enlaces puede dar la base de una compañía de marketing exitosa.

Como herramienta de data mining, el análisis de links es soportado ineficientemente por la tecnología de base de datos relacionales. El gran área donde es aplicada actualmente es en la ejecución de leyes, donde los indicios de crímenes están unidos juntos, para ayudar a resolver nuevos crímenes. Las pocas herramientas disponibles se enfocan en la visualización de los links, más que en analizar los patrones.

3.3.5 Árboles de Decisión

Los árboles de decisión son modelos poderosos, producidos por una clase de técnicas que incluye árboles de clasificación y regresión, e inducción automática, chi-cuadrado. Los árboles de decisión son usados para data mining directo, particularmente para clasificación, ellos dividen los registros en el conjunto de "entrenamiento" en subconjuntos disjuntos, cada uno de ellos es descripto por una simple regla sobre cada uno de los campos.

Una de las ventajas principales de los árboles de decisión es que el modelo es explicativo, ya que toma la forma de una regla explícita, ésto permite evaluar los resultados, identificar atributos claves en el proceso. Las reglas pueden ser expresadas fácilmente con sentencias de lógica, en un lenguaje como SQL, con lo cual pueden ser aplicadas directamente a nuevos registros.

3.3.6 Redes Neuronales Artificiales

Las redes neuronales son probablemente la técnica más común de data mining, quizás para algunos lectores sinónimo de data mining, son simples modelos de las conexiones neuronales del cerebro, adaptados para usar computadoras digitales. Aprenden desde el conjunto de "entrenamiento", generalizando patrones dentro de el para clasificación y predicción. Las redes neuronales pueden también ser aplicadas a data mining indirecto, y predicciones "time-series". Nuevas aplicaciones y nuevas estructuras para redes neuronales están siendo investigadas.

Una de las principales ventajas de las redes neuronales es su amplia aplicabilidad. Existen gran cantidad de herramientas que soportan redes neuronales y en muchas plataformas. También son interesantes porque detectan patrones en los datos de una forma análoga al pensamiento humano.

Las redes neuronales tiene dos grandes desventajas. La primera es la dificultad de entender los modelos que producen. La segunda es su particular sensibilidad al formato de los datos de entrada. Diferentes representaciones de datos pueden producir diferentes resultados.

3.3.7 Algoritmos Genéticos

Los algoritmos genéticos (GA) aplican el mecanismo de selección genético natural para explorar usado para encontrar el conjunto óptimo de parámetros que describe una función predictiva y es usado para data mining directo. Los algoritmos genéticos son similares a los estadísticos, en que la forma del modelo necesita ser conocida por adelantado. Los algoritmos genéticos usan la selección, cruzamiento y la mutación de operadores para desarrollar sucesivas generaciones de soluciones. Sólo genera la que tiene más probabilidad de sobrevivir, hasta que converge en una solución óptima.

Los algoritmos genéticos pueden ser utilizados para resolver muchos tipos de problemas, como lo haría otras técnicas de data mining. Es utilizado para mejorar las técnicas MBR y redes neuronales.

3.3.8 OLAP (On-Line Analytic Processing)

No todo lo que es útil para analizar datos, es necesariamente data mining. OLAP es la forma de presentar datos relacionales a los usuarios, para facilitar el entendimiento de los datos y los patrones importantes dentro de ellos. Como la visualización, no es una herramienta específicamente para data mining, pero es una herramienta importante en el arsenal de medios usados para extraer y presentar información.

Los métodos OLAP están basados en bases de datos multidimensionales (MDDs). MDDs son una representación de los datos que permite a los usuarios introducirse en los datos, para entender varias conclusiones importantes. Este tipo de herramienta ayuda al objetivo de transformar los datos en información.

4 Reglas de Asociación

4.1 Introducción

Data Mining fue recientemente reconocida como un prometedor y nuevo campo en la intersección de Bases de Datos, Inteligencia Artificial y aprendizaje de máquina. El área puede ser definida vagamente como encontrar reglas interesantes en grandes colecciones de datos.

Recientemente, Agrawal, Imielwksy y Swami introdujeron una clase de similitudes, ***reglas de asociación***, y dieron algoritmos para encontrar tales reglas de asociación desde una base de datos binarios.

Las reglas de asociación son sentencias de la forma "para el 90% de las filas de la relación, si la fila tiene el valor 1 en la columna del conjunto W, entonces también tendrá 1 en la columna B".

Una regla de asociación es una expresión $W \Rightarrow B$, donde W es un conjunto de atributos y B es un único atributo. El significado intuitivo de esta regla es que en las filas de la base de datos donde los atributos en W tiene el valor "verdadero", también el atributo B tiende a tener el valor "verdadero". Por ejemplo, una regla puede expresar que los estudiantes que están tomando los cursos CS101 y CS120, también toman el curso CS130, ésta clase de información puede ser usada, por ejemplo, para asignar aulas para los cursos. Aplicaciones de reglas de asociación incluyen análisis del comportamiento de clientes, en un supermercado, en un ambiente bancario, diagnóstico y predicción.

4.2 Reglas de Asociación Booleanas

El propósito de descubrir asociaciones es encontrar ítems que implican la presencia de otros ítems en la misma transacción. Considere una base de datos de ventas donde cada venta (transacción) consiste de varios artículos (ítems) comprados por un cliente, aplicando una técnica para descubrir asociaciones para tratar de determinar si este conjunto de transacciones encubre afinidades entre colecciones de ítems. Estas afinidades entre ítems son representadas por reglas de asociación. Una regla muestra, en una forma textual, cuales ítems implican la presencia de otros ítems. La siguiente figura ilustra una regla derivada de un análisis de canasta de compras:

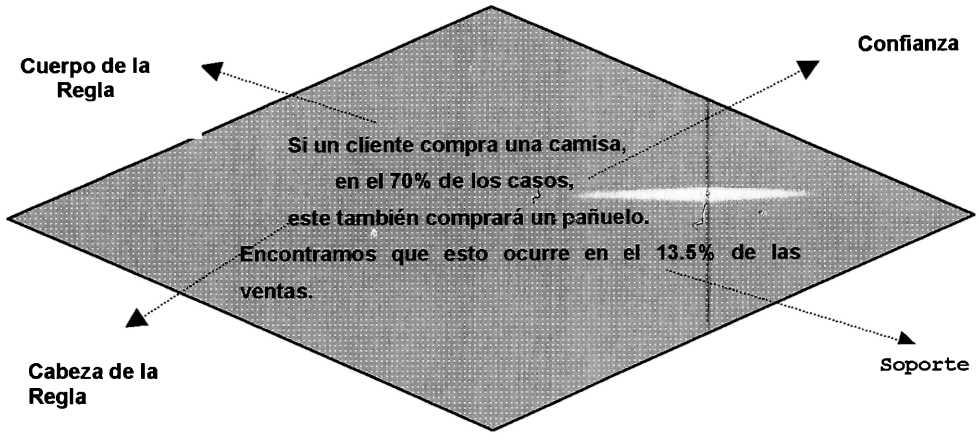


Figura. Una Regla de Asociación

Genéricamente, la regla tiene la forma "Si X, entonces Y", X es llamado cuerpo de la regla, e Y es la cabeza de la regla. Los algoritmos de asociación son completamente eficientes en la derivación de reglas. De hecho, la derivación de reglas no es el punto, por el contrario, el desafío para el analista es hacer un juicio sobre la validez e importancia de las reglas. Dos parámetros son importantes para este juicio: el factor soporte y el factor confianza.

El factor soporte indica la relativa ocurrencia de las reglas de asociación detectadas en el interior de todos los datos del conjunto de transacciones. Es una medida relativa determinada por la división del número de transacciones que soportan la regla por el número total de transacciones. Una transacción soporta la regla "cuando X entonces Y" si ítems X e Y en la regla también ocurren en la transacción. En el ejemplo de la figura, la regla es soportado por el 13,5% de todos los registros de la base de datos.

El factor de confianza de una regla de asociación es el grado en el que una regla es verdadera a través de registros individuales. Es calculado dividiendo el número de transacciones que soportan la regla por el numero de transacciones que soportan solo el cuerpo de la regla. En el ejemplo el factor de confianza es 70%.

La técnica de descubrimiento de asociación está basada en contar las ocurrencias de todas las posibles combinaciones de ítems. Primero, antes de buscar (mining) asociaciones, los identificadores de transacción (IDs) son ordenados secuencialmente. Luego la técnica comienza con el conteo de las ocurrencias de todos los ítems simples que se encuentran en el conjunto de datos con transacciones, y crea una columna uni-dimencional, o vector, donde las celdas contienen un contador de ocurrencias para cada ítem. Todas aquellas celdas que tiene valor por debajo del nivel de soporte son ignorados. Luego, un vector bi-dimensional, o matriz, es formada para guardar los contadores de las ocurrencias de cada ítems con otro posible ítem, y nuevamente las celdas son sacadas cuando el contador está por debajo del nivel de soporte. Este proceso se repite. Entonces la técnica involucra leer el conjunto de datos secuencialmente desde el comienzo hacia el final, cada vez una dimensión es agregada y realiza un simple conteo de las ocurrencias.

Claramente, el número promedio de ítems por la transacción puede afectar la ejecución. Por ejemplo, buscar reglas con un ítem sólo en el cuerpo de la regla, otro en la cabeza toma muchos menos tiempo que buscar aquellas que tiene por ejemplo ocho ítems en el cuerpo y dos ítems en la cabeza.

La salida de la aplicación del algoritmo junto al conjunto de transacciones es la lista de patrones que expresan afinidades entre ítems, siempre en un formato similar al de la figura. Para cada regla el factor de soporte y confianza es expresado y estadísticas adicionales pueden ser suplantadas, como el Lift de una asociación. Lift es explicado mejor con un ejemplo: si el factor de soporte para el ítem en todo el conjunto de transacciones es 20% (esto es, en el 20% de las transacciones, esta el ítem), y el factor de confianza por la asociación entre el ítem y el ítem es 70%, el Lift es 3,5, indicando que la ocurrencia esperada del ítem en una transacción es más que el triple si éste ocurre en una transacción que contiene ítem.

Las reglas basadas en factores de soporte y confianza altos representan un alto grado de relevancia que reglas con factores de soporte y confianza bajos. No obstante, los factores de soporte y confianza son muy altos, las reglas de asociación pueden ser no descubiertas. Si los factores son muy bajos, habrá muchas combinaciones posibles de asociaciones de ítems, que no son las que se buscaban. La probabilidad de correlaciones ilegitimas crece exponencialmente con el tamaño del conjunto de datos.

El descubrimiento de asociaciones tiene la gran ventaja de ser muy simple. Los factores de soporte y confianza son los únicos dos parámetros que se deben setear, y las reglas resultantes dan ellas mismas una interpretación intuitiva.

Adicionalmente, la técnica en si misma es fácil, porque es simplemente un problema de contar todas las ocurrencias de todas las combinaciones posibles de ítems e involucra leer una tabla secuencialmente desde arriba hacia abajo, cada vez una nueva dimensión es agregada. Así la técnica es particularmente adecuada para manejar grandes números de transacciones que pueden estar basadas en un número de ítems dentro de la magnitud de diez millones en el menor de los casos.

Una desventaja del descubrimiento de asociaciones es que no proveen una relación de importancia de una asociación dentro del contexto la aplicación.

4.2.1 Modelo Formal

Sea $I = I_1, I_2, ..., I_m$ un conjunto de atributos, llamados ítems, con dominio en el conjunto binario {0,1}, sea T la base de datos de transacciones. Cada transacción t es representada como un vector binario, con t[k] = 1 si t tiene el ítem I_k, y t[k] = 0 caso contrario. Hay una tupla en la base de datos para cada transacción. Sea X el conjunto de algunos ítems de I. Se dice que la transacción t *satisface* a X si para todos los ítems I_k en X, t[k] = 1.

Por *regla de asociación*, se entiende una implicación de la forma $X => I_j$, donde X es un conjunto de algunos ítems de I e I_j es un simple de I, que no esta presente en el conjunto X. La regla $X => I_j$ se satisface en el conjunto de transacciones T con el factor de confianza $0 <= c <= 1$ si y sólo si al menos el c% de las transacciones en T que satisfacen X también satisfacen I_j. Usaremos la notación $X => I_j | c$ para especificar que la regla $X => I_j$ tiene factor de confianza c.

Dado el conjunto de transacciones T, se quiere generar todas las reglas que satisfagan la siguiente restricción:

- Restricción de Soporte: se refiere al número de transacciones en T que soportan la regla. El *soporte* para una regla es definido como la fracción de transacciones en T que satisfacen la unión de los ítems del consecuente y antecedente de la regla. El soporte no debe confundirse con la confianza, mientras que la confianza en una medida de validez (poder) de la regla, el soporte corresponde a un significado estadístico.

El problema de encontrar reglas puede ser descompuesto en dos subproblemas:

1. Generar todas las combinaciones de ítems que tengan soporte por encima de una entrada, llamada *minsupport* (mínimo soporte *minsup*). Llamamos a todas esas combinaciones, *large itemsets* (conjunto de ítems grande), y todas aquellas combinaciones que no tengan soporte mayor al minsup *small itemsets* (conjunto de ítems pequeño).
2. Para cada itemset grande dado Y = I_1, I_2, ..., I_k, con k >= 2, generar todas las reglas (al menos k reglas) que usen los ítems del conjunto I_1, I_2, ..., I_k . El antecedente de cada una de estas reglas será un subconjunto X de Y tal que X tiene k-1 ítems, y el consecuente será Y-X. Para generar una regla X => I_j |c, donde X = I_1, I_2, ..., I_j, I_{j+1}, ..., I_k, toma el soporte de Y y lo divide por el soporte de X. Si la razón es mayor que c, luego la regla es satisfecha con factor de confianza c, o no lo es caso contrario.

Note que si el itemset Y es grande, luego cada subconjunto de Y también va a ser grande. Además todas las reglas derivadas de Y van a satisfacer la restricción de soporte porque Y satisface la restricción de soporte, es la unión de los ítems del consecuente y el antecedente de cada regla.

4.3 Reglas de Asociación Cuantitativas

Conceptualmente, este problema puede ser visto como encontrar asociaciones entre los valores "1" en una tabla relacional donde todos los atributos son booleanos. La tabla tiene un atributo correspondiente a cada ítem y un registro para cada transacción. Se lo denomina como el problema de Encontrar Reglas de Asociaciones Booleanas.

En dominios comerciales y científicos hay tipos de atributos más ricos. Los atributos pueden ser *cuantitativos* (edad, sueldo, etc.) o *categóricos* (marcas de auto). Los atributos booleanos pueden ser categorizados como un caso especial de atributos categóricos. Aquí se define el problema de encontrar reglas de asociación a pesar de atributos cuantitativos y categóricos. Se lo denomina como problema de encontrar Reglas de Asociación Cuantitativas.

Si se tiene una tabla Personas (Figura 1) con tres atributos no claves, Edad y Nro. Autos son atributos cuantitativos, mientras que Casado es un atributo categórico. Una regla de asociación cuantitativa presente en esta tabla puede ser:

<Edad: 30 .. 39> y <Casado: SI> => <Nro. Autos: 2>

Personas

IDs	Edad	Casado	Nro. Autos
100	23	no	1
200	25	si	1
300	29	no	0
400	34	si	2
500	38	si	2

(Soporte Mínimo = 40%, Confianza Mínimo = 50%)

Reglas (ejemplos)	Soporte	Confianza
<Edad: 30 .. 39> y <Casado: si> => <Nro. Autos: 2>	40%	100%
<Nro. Autos: 0 .. 1> => <Casado: no>	40%	66,6%

Figura -1-

4.3.1 Mapeo del problema de las Reglas de Asociación Cuantitativas en el problema de Reglas de Asociación Booleanas

Si todos los atributos son categóricos o aquellos que son cuantitativos tienen unos pocos valores, este mapeo es derecho. Conceptualmente, en vez de tener un único campo en la tabla para cada atributo, se tiene tantos campos como el número de valores de atributos. El valor de un campo booleano correspondiente a <attrib. 1, valor1> debería ser "1" si el atributo tiene el valor 1 en el registro original, y "0" en caso contrario.

Si el dominio de valores para un cuantitativo es aproximadamente grande, una aproximación obvia sería primero dividir los valores en intervalos y luego mapear cada par <attrib., intervalo> a un atributo booleano. Podemos ahora usar cualquier algoritmo para encontrar reglas de asociación booleanas (por ej. para encontrar reglas de asociación cuantitativas).

IDs	Edad: 20..29	Edad: 30..39	Casado: si	Casado: no	Nro. Autos 0	Nro. Autos: 1	Nro. Autos: 2
100	1	0	0	1	0	1	0
200	1	0	1	0	0	1	0
300	1	0	0	1	1	0	0
400	0	1	1	0	0	0	1
500	0	1	1	0	0	0	1

Figura -2-

La figura 2 muestra este mapeo para los atributos no claves de la tabla Personas dada en la figura 1. La edad es dividido en dos intervalos: 20..29 y 30 39. El atributo

categórico, casado, tiene dos atributos booleanos "Casado: si" y "Casado: no". Además el número de valores para Nro. Autos es pequeño, Nro. Autos no es dividido en intervalos, cada valor es mapeado a un campo booleano. El registro 100, el cual tenia <Edad: 23> ahora tiene <Edad: 20..29> igual a "1", <Edad: 30..39> igual a "0", etc.

4.3.1.1 Probreza en el Mapeo:

Estos son dos problemas con esta simple aproximación aplicada a atributos cuantitativos:

- "MinSup". Si el número de intervalos para un atributo cuantitativo (o valores, si el atributo no es particionado) es grande, el soporte para cualquier intervalo simple puede ser bajo. Además, sin usar grandes intervalos, algunas reglas que involucran estos atributos pueden no ser encontradas porque ellas necesitan el soporte mínimo.
- "MinConf". Siempre que se particionen valores en intervalos, alguna información se pierde. Algunas reglas pueden tener confianza mínima solamente cuando un ítem en el antecedente consiste de un único valor (o un intervalo pequeño). La información que se pierde aumenta en forma proporcional al tamaño del intervalo.

Por ejemplo en la Figura 2, la regla <Nro. Autos:0> => <Casado: no> tiene un 100% de confianza. Pero si tenemos dividido el atributo Nro. Autos en intervalos tal que 0 y 1 estén en la misma partición, se tiene la regla mas cerrada <Nro. Autos: 0..1> => <Casados: no>, la cual tiene solamente el 66,6% de confianza.

Esta situación se denomina "*catch-22*" (trampa-22), y es creada por estos dos problemas. Si los intervalos son muy largos, algunas reglas pueden no tener la confianza mínima; si estos son muy pequeños, algunas reglas pueden no tener el soporte mínimo.

Para salir de la situación "*catch-22*", se puede considerar todos los rangos continuos posibles sobre los valores del atributo cuantitativo, o sobre el intervalos particionados. El problema "MinSup" ahora desaparece puesto que podemos combinar valores/intervalos adyacentes. El problema "MinConf" esta todavía presente, sin embargo, la información perdida puede ser reducida aumentando el número de intervalos, sin llegar al problema "MinSup".

Desafortunadamente, incrementar el número de intervalos mientras simultáneamente se combinan intervalos adyacentes, introduce dos nuevos problemas:

- "Tiempo de Ejecución": si un atributo cuantitativo tiene $\underline{n}$ valores (o intervalos), esto esta sobre un rango promedio del orden $O(n^2)$ que incluye un valor o intervalo especifico. De aquí el numero de ítems por registro se incrementa, lo cual se incrementaría el tiempo de ejecución.
- "Muchas Reglas": si un valor (o intervalo) de un atributo cuantitativo tiene soporte mínimo, entonces también lo tendrá cualquier rango que contiene este valor/intervalo. Así el número de reglas se incrementa. Muchas de estas reglas no serían interesantes.

Hay un negociado entre el tiempo de ejecución más rápido con menos intervalos y la reducción de la perdida de información con más intervalos. Se puede reducir la perdida de información aumentando el número de intervalos, con el costo de un

incremento en el tiempo de ejecución y la generación de muchas reglas no interesantes.

4.3.1.2 Una Aproximación

Considerando rangos sobre valores/intervalos adyacentes de atributos cuantitativos para evitar el problema "MinSup". Para mitigar el problema "Tiempo de Ejecución", restringimos el grado para el cual valores/intervalos adyacentes pueden ser combinados introduciendo un parámetro "Máximo Soporte" especificado por el usuario, se para la combinación de intervalos se sus soportes combinados exceden este valor. Sin embargo, cualquier valor/intervalo simple cuyo soporte exceda el soporte máximo es todavía considerado.

Pero como decidir si se particiona un atributo cuantitativo o no. Y cuántas particiones se deberían realizar en el caso de decidir particionarlo?. Luego se introduce una medida de completitud parcial que da un manejo sobre la pérdida de información por particionamiento y ayudando a tomar estas decisiones.

Para discutir el problema de "Muchas Reglas", también se introducirá una medida de interés posteriormente. La medida de interés esta basada en el desvío de la esperanza y ayuda a descartar reglas no interesantes.

El algoritmo para descubrir reglas de asociación cuantitativas comparte la estructura básica, con el que encuentran reglas de asociación booleanas.

4.3.2 Enunciado y Descomposición del Problema

Ahora se da una declaración formal del problema de encontrar de reglas de asociación cuantitativas y alguna terminología.

Se utiliza un lema simple para tratar atributos categóricos y cuantitativos uniformemente. Para atributos categóricos, los valores del atributo son mapeados a un conjunto de enteros consecutivos. Para los cuantitativos que son particionados en intervalos, los valores son mapeados a enteros consecutivos, tal que el orden de los intervalos es conservado. Estos mapeos tratan a un registro de la base de datos como un conjunto de pares <atributo, valor entero>, sin perdida de generalidad.

Sea $I = \{i_1, i_2, ..., i_m\}$ un conjunto de literales, llamados atributos. Si P denota el conjunto de enteros positivos. I_v denota el conjunto Y x P. Un par $\langle x, v\rangle \in I_v$ denota el atributo x, con el valor asociado v. Si Y_R denota el conjunto $\{\langle x, l, u\rangle \in I \times P \times P \mid l \leq u$, si x es cuantitativo; $l = u$, si x es categórico$\}$. Así una tripleta $\langle x, l, u\rangle \in I_R$ denota un atributo cuantitativo x con un valor en el intervalo [l, u], o un atributo categórico x con valor l. Puede referirse a esta tripleta como un ítem. Para cualquier $X \subseteq I_R$, el atributo(x) denota al conjunto $\{ x \mid \langle x, l, u\rangle \in X \}$.

Con la definición anterior, solamente los valores están asociados con atributos categóricos, mientras que valores y rangos pueden ser asociados con los cuantitativos. En otras palabras, los valores de atributos categóricos son combinados.

Sea D un conjunto de registros, donde cada registro R es un conjunto de valores de atributos tal que $R \subseteq I_v$. Asumimos que cada atributo ocurre al menos en un registro.

Se dice que un registro R *soporta* $X \subseteq I_R$, si $\forall$ <x, I, u> $\in X$ ($\exists$ <x, q> $\in$ R tal que $I \leq q \leq u$).

Una *regla de asociación cuantitativa* es una implicación de la forma X => Y, donde $X \subset I_R$, $Y \subset I_R$, y atributos(X) $\cap$ atributos(Y) = $\varnothing$. La regla X => Y tiene en el conjuntos de registros D *confianza c* si el c% de los registros en D que soportan X también soportan Y. La regla X => Y tiene *soporte s* en el conjunto de registros D si el s% de registros en D soportan $X \cup Y$.

Dado un conjunto de registros D, el problema de encontrar reglas de asociación cuantitativas es encontrar todas las reglas de asociación cuantitativas que tienen soporte y confianza mayor que el mínimo soporte especificado por el usuario (llamado MinSup) y la confianza mínima (llamada MinConf) respectivamente. Note que el hecho de que los ítems en una regla puedan ser categóricos o cuantitativos tiene que ser ocultado en la definición de una regla de asociación.

Notación. Diremos que un ***ítem*** es una tripleta que representa un atributo categórico con su valor, o un atributo cuantitativo con su rango.(El valor de un atributo categórico puede ser representado como un rango donde el límite superior es igual al inferior). Usamos el termino ***itemset*** para representar un conjunto de ítems. El soporte de un itemset $X \subset I_R$ es simplemente el porcentaje de registros en D que soportan a X. Usamos el termino ***frequent itemset*** para representar un itemset con soporte mínimo.

Pr(X) denota la probabilidad de que todos los ítems en $X \subset I_R$ están soportados por un registro dado. Entonces soporte(X => Y) = Pr($X \cup Y$) y confianza(X => Y) = Pr(X\Y). (Pr($X \cup Y$) es la probabilidad de que todos los ítems en $X \cup Y$ estén presentes en el registro).

Llamaremos $X^\wedge$ a la generalización de X (y X es una especialización de $X^\wedge$) si atributos(X) = atributos($X^\wedge$) y $\forall$ x $\in$ atributos(X) [<x, I, u> $\in X$ y <x, I', u'> $\in X^\wedge$ => $I' \leq I \leq u \leq u'$]. Por ejemplo, el itemset {<Edad: 30 .. 39>, <Casado: si>} es una generalización de {<Edad: 30 .. 35>, <Casado: si>}.

4.3.2.1 Descomposición del Problema

Se descompone el problema de descubrir reglas de asociación cuantitativas en 5 pasos:

1. Determinar el número de particiones para cada atributo cuantitativo.
2. Para atributos categóricos, mapear los valores de los atributos a un conjunto de enteros consecutivos. Para los atributos cuantitativos que no son particionados en intervalos, los valores son mapeados a enteros consecutivos tal que el orden de los valores es respetado. Si un atributo cuantitativo es particionado en intervalos, el intervalo es mapeado a enteros consecutivos, tal que el orden de los intervalos es respetado. Desde este punto, el algoritmo solamente ve valores (o rangos sobre valores) para atributos cuantitativos. Que estos valores puedan representar intervalos es transparente para el algoritmo.
3. Encontrar el soporte para cada valor de atributo cuantitativo y categórico. Además, para atributos cuantitativos, los valores adyacentes son combinados muchas veces mientras su soporte sea menor que el soporte máximo especificado por el usuario. Ahora conocemos todos los rangos y valores con soporte mínimo para cada atributo cuantitativo, así también todos los valores con soporte mínimo para cada atributo categórico. Esto forma el conjunto de todos los ítems . Luego, buscar todos los conjuntos de ítems cuyo soporte es mayor que el

mínimo soporte especificado por el usuario ("MinSup"). Esos son los *"frequent itemsets"* itemsets frecuentes.

4. Usar los itemsets frecuentes para generar reglas de asociación. La idea general es que si, se dice, ABCD y AB son itemsets frecuentes, luego podemos determinar si al regla AB => CD tiene por cómputos la razón confianza = soporte(ABCD) / soporte(AB). Si confianza ≥ MinConf, entonces tenemos la regla. La regla tendrá mínimo soporte porque ABCD es frecuente.
5. Determinar las reglas interesantes de la salida.

Personas

IDs	Edad	Casado	Nro. Autos
100	23	no	1
200	25	si	1
300	29	no	0
400	34	si	2
500	38	si	2

Figura -3a-

Particiones para Edad

Intervalo
20 .. 24
25 .. 29
30 .. 34
35 .. 39

Figura -3b-

Mapeo de Edad

Intervalo	Entero
20 .. 24	1
25 .. 29	2
30 .. 34	3
35 .. 39	4

Figura -3d-

Mapeo de Casado

Valor	Entero
Si	1
No	2

Figura -3d-

Después del Particionamiento de Edad

Ids	Edad	Casado	Nro. Autos
100	20 .. 24	2	1
200	25 .. 29	1	1
300	25 .. 29	2	0
400	30 .. 34	1	2
500	35 .. 39	1	2

Figura -3e-

Itemsets Frecuentes: Muestra

Itemset	Soporte
{ <Edad: 20 .. 29> }	3
{ <Edad: 30 .. 39> }	2
{ <Casado:si > }	3
{ <Casado: no > }	2
{ <Nro. Autos: 0 .. 1 > }	3
{ <Edad: 30 .. 39 >, <Casado: si > }	2

Figura -3f-

Reglas: Muestra

Reglas	Soporte	Confianza
<Edad: 30 .. 39> y <Casado: si> => <Nro. Autos: 2>	40%	100%
<Edad: 20 .. 29> => <Nro. Autos: 0 .. 1>	60%	66,6%

Figura -3g-

Figuras -3- Ejemplo de Descomposición del Problema

Ejemplo. Considere la tabla Personas de la Figura -3a-. Hay 2 atributos cuantitativos: Edad y Nro. Autos. Se asume que en el paso 1 se decide particionar al campo Edad en cuatro intervalos, como se muestra en la figura -3b- .Conceptualmente, la tabla ahora se vería como lo muestra la figura -3c-. Después

del mapeo de los intervalos a enteros consecutivos, usando las figuras -3d-, la tabla resultante es la que se muestra en la figura -3e-. Asumiendo el MinSup de 40% y MinConf de 50%, la figura -3f- muestra algunos de los itemsets frecuentes, y la figura -3g- muestra algunas de las reglas.

Se ha reemplazado los números mapeados con los valores de la tabla original en las dos ultimas figuras. Note que el ítem <Edad: 20 .. 29> corresponde a la combinación de los intervalos 20 .. 24 y 25 .. 29, etc. No se muestra en este ejemplo el paso 5, determinar las reglas interesantes.

4.3.3 Particionar Atributos Cuantitativos

En esta sección, consideramos cuando deberíamos particionar los valores de atributos cuantitativos en intervalos, y cuantas particiones deberían haber. Primero, presentamos una medida de ***completitud parcial*** la cual da una noción de la información perdida por el particionamiento. Luego mostramos el particionamiento ***"equi-depth",*** que minimiza el número de intervalos requeridos para satisfacer este nivel de completitud parcial.

La idea de la medida de completitud parcial, es como sigue: sea R el conjunto de reglas obtenidas considerando todos los rangos sobre los valores, sin ninguna preparación previa, de los atributos cuantitativos; sea R' el conjunto de reglas obtenidas considerando todos los rangos sobre las particiones de los atributos cuantitativos. Una forma de medir la pérdida de información es ir de R a R' y ver para cada regla en R, cuan *"lejos"* está la regla más *"cercana"* (closets). Cuanto más lejos este la regla "closets", mayor es la pérdida de información. Por definición la regla más "close" (cercana) son generalizaciones, y usando la razón del soporte de las reglas como una medida de cuan lejos están las reglas.

4.3.3.1 Completitud Parcial

Se define completitud parcial sobre los itemsets, primero que en reglas, como se puede garantizar que un itemset "close" puede ser encontrado mientras que no se puede garantizar que la regla "close" va a ser encontrada. Después se muestra que podemos garantizar que la regla "close" va a ser encontrada si el nivel de MinConf para R' es menor que para R.

Sea C el conjunto que denota todos los itemsets frecuentes en D. Para cualquier $K \geq 1$, llamamos P K-completo con respecto a C si:

- $P \subseteq C$
- $X \in P$ y $X' \subseteq X \Rightarrow X' \in P$, y
- $\forall X \in C$ [$\exists X^{\wedge} \in P$ tal que:

  1. $X^{\wedge}$ es una generalización de X y soporte($X^{\wedge}$) $\leq K *$ soporte(X), y

  2. $\forall Y \subseteq X \exists Y^{\wedge} \subseteq X^{\wedge}$ tal que $Y^{\wedge}$ es una generalización de Y y

  soporte($Y^{\wedge}$) $\leq$ soporte(Y)]

Las dos primeras condiciones aseguran que P solo contiene itemsets frecuentes y que se puede generar reglas desde P. La primera parte de la tercera condición dice que para cualquier itemset en C, hay una generalización de ese itemset con al

menos K veces el soporte en P. La segunda parte dice que la propiedad que la generalización tiene al menos K veces el soporte también lo tiene para los correspondientes subconjuntos de atributos de ese itemset y su generalización. Si K = 1, P es idéntico a C.

Pos ejemplo, se asume que en alguna tabla, los siguientes son itemsets frecuentes en C:

Numero	Itemset	Soporte
1	{<Edad: 20 .. 30>}	5%
2	{<Edad: 20 .. 40>}	6%
3	{<Edad: 20 .. 50>}	8%
4	{<Autos: 1 .. 2>}	5%
5	{<Autos: 1 .. 3>}	6%
6	{<Edad: 20 .. 30>, <Autos: 1.. 2>}	4%
7	{<Edad: 20 .. 40>, <Autos: 1 .. 3>}	5%

Los itemsets 2, 3, 5 y 7 pueden venir de un conjunto 1.5-completo, entonces para cualquier itemset X, cualquiera de ellos, 2, 3, 5 o 7 es una generalización cuyo soporte es al menos 1.5 veces el soporte de X. Por ejemplo, el itemset 2 es una generalización del itemset 1, y el soporte del itemset 2 es 1.2 veces el soporte del itemset 1. Los itemsets 3, 5 y 7 no forman un conjunto 1.5-completo porque para el itemset 1, la única generalización entre 3, 5 y 7 es el itemset 3, y el soporte de 3 es mas de 1.5 veces el soporte de 1.

A partir de aquí se enunciarán lemas y corolarios, los cuales no están demostrados ya que está hecho en el paper Mining Quantitative Association Rules en Grandes Tablas Relacionales.

Lema1. Sea P un conjunto K-completo w.r.t. C, el conjunto de todos los itemsets frecuentes. Sea R_C en conjunto de las reglas generadas desde C, para un nivel de confianza mínimo MinConf. Sea R_P el conjunto de las reglas formadas desde P con confianza mínima igual a MinConf / K. Entonces para cualquier regla $A \Rightarrow B$ en R_C, hay una regla $A^{\wedge} \Rightarrow B^{\wedge}$ en R_P tal que

- $A^{\wedge}$ es una generalización de A, $B^{\wedge}$ es una generalización de B,
- el soporte de $A^{\wedge} \Rightarrow B^{\wedge}$ es a lo sumo K veces el soporte de $A \Rightarrow B$, y
- la confianza de $A^{\wedge} \Rightarrow B^{\wedge}$ es menos que 1/K veces, y a lo sumo K veces la confianza de $A \Rightarrow B$.

Entonces, dado el conjunto de itemsets frecuentes P que son K-completos w.r.t. del conjunto de todos los itemsets frecuentes, la confianza mínima cuando se generan las reglas de asociación desde P debe ser 1/K veces el nivel deseado para garantizar que una regla "close" va a ser generada.

En el ejemplo dado, los itemsets 2, 3 y 5 forman un conjunto 1.5-completo. La regla "<Edad: 20 .. 30> ⇒ <Autos: 1 .. 2>" tiene 80% de confianza, mientras que su correspondiente regla generalizada "<Edad: 20 ..40> ⇒ <Autos: 1 .. 3>" tiene 83.3% de confianza.

4.3.3.2 Número de Particiones - Determinación

Primero se verán algunas propiedades de atributos particionados (w.r.t. completitud parcial), las cuales se utilizan para decidir el número de intervalos para un nivel de completitud dado. Como ya se ha dicho, no nos concentraremos en las demostraciones, debido a que están realizas en el paper.

Lema2. Considerando un atributo cuantitativo x, y algún real K > 1. Asumimos una partición para x en intervalos (llamados intervalos bases) tal que para cualquier intervalo base B, cualquiera sea el soporte de B es menor que el MinSup*(K-1)/2 o B consiste de un valor simple. Sea P el conjunto denotado de todas las combinaciones de intervalos bases que tengan el mínimo soporte. Luego P es K-completo w.r.t. el conjunto de todos los rangos sobre x con el soporte mínimo.

Lema 3. Considerando un conjunto de n atributos cuantitativos, y algún real K > 1. Asumimos que cada atributo es particionado en intervalos tal que para cualquier intervalo base B, cualquiera sea el soporte de B es menor que el MinSup*(K-1)/(2*n) o B consiste de un valor simple. Sea P el conjunto denotado de todos los itemsets frecuentes sobre los atributos particionados. Luego P es K-completo w.r.t. el conjunto de todos los itemsets frecuentes (obtenidos sin particionamiento).

Para cualquier particionamiento dado, se puede usar el Lema 3 para computar el nivel de completitud parcial para ese particionamiento. Primeros se ilustra el procedimiento para un atributo sólo. En este caso, simplemente encontramos la partición con mayor soporte entre aquellos con más de un valor, sea el soporte de esta partición s. Entonces, para encontrar el nivel de completitud parcial K, se usa la formula s=MinSup*(K-1)/2 del Lema 2 y despejando tomamos la formula K=1+2*s/Minsup. Con n atributos, la formula es

$$K=1+((2*n*s)/(MinSup),$$

donde s es el máximo soporte para una partición con más de un valor, entre todos los atributos cuantitativos. A menor nivel de completitud parcial, la información perdida es menor. La formula refleja esto: como s decrece, implicando más intervalos, el nivel de completitud parcial decrece.

Lema 4. Para cualquier número de intervalos especificados, el particionamiento equi-depth minimiza el nivel parcial de completitud.

Corolario 1. Para cualquier nivel de completitud parcial dado, el particionamiento equi-depth minimiza el número de intervalos requeridos para satisfacer el nivel parcial de completitud.

Dado el nivel de completitud parcial deseado por el usuario, y el mínimo soporte, se puede calcular el número de particiones requeridas (asumiendo un particionamiento equi-depth). Desde el Lema 3, se sabe que tomando el nivel de completitud parcial K, el soporte de cualquier partición con más de un valor debe ser menor a MinSup*(K-1)/(2*n) donde n es el número de atributos cuantitativos. Ignorando el caso especial de particiones que contienen un único valor, y asumiendo que el particionamiento equi-depth divide el soporte idénticamente, debe haber 1/s particiones en orden para tomar el soporte de cada partición menor que s. Entonces se tiene

$$\text{Número de Intervalos} = (2 * n) /(m * (K - 1)$$

donde,

n = número de atributos cuantitativos

m = Mínimo soporte (como fracción)

K = Nivel de Completitud Parcial

4.3.4 Reglas Interesantes

Un problema potencial con la combinación de atributos cuantitativos es que el número de reglas encontradas puede ser muy grande. En el paper [10] se muestra una medida subjetiva de interés y sugiere que un patrón es interesante si es inesperado (sorprendente para el usuario) y/o accionable (el usuario puede hacer algo con él). También en este paper se distingue entre medidas de interés subjetivas y objetivas. En el paper [12] discute una clase de medida de interés objetiva basado en cuanto el soporte de una regla se desvía del soporte que podría ser, si el antecedente y el consecuente de una regla son independientes.

En el paper [11] se presenta una medida de interés llamada " Mayor que el valor esperado" para identificar reglas interesantes. Esta medida de interés mira la generalización y la especialización de la regla para identificar reglas interesantes. Presentamos esta medida de interés con un ejemplo:

Se consideran las siguientes reglas,

<Edad: 20 .. 30> $\Rightarrow$ <Autos: 1 .. 2> (8% soporte, 70% confianza)

<Edad: 20 .. 25> $\Rightarrow$ <Autos: 1 .. 2> (2% soporte, 70% confianza)

La segunda regla puede ser considerada redundante porque no contiene información adicional y es menos general que la primer regla. Dada la primer regla se espera que la segunda regla tenga la misma confianza que la primera y soporte igual a un cuarto del soporte de la primera. Aún si la confianza de la segunda regla tenga una pequeña diferencia en menos, digamos el 63% en vez del 70%, ésta no va a traer más información significativa que la primer regla. La noción de interés se define: sólo se quiere encontrar aquellas reglas cuyo soporte y/o confianza es mayor al esperado. (El usuario puede especificar si debe ser el soporte *y* la confianza, o soporte *o* confianza)

Finalmente se tiene una formalización de la idea de medida de interés "Mayor que valor esperado" de la siguiente forma:

Un itemset X es R-interesante con respecto a $X^{\wedge}$ si el soporte de X es mayor o igual a R veces el soporte esperado basado en $X^{\wedge}$ y para cualquier especialización X' tal que X' tenga el mínimo soporte y X-X' $\subseteq I_R$, X-X' es R-interesante con respecto a $X^{\wedge}$.

Similarmente, una regla $X \Rightarrow Y$ es R-interesante w.r.t. un ancestro $X^{\wedge} \Rightarrow Y^{\wedge}$ si el soporte de la regla $X \Rightarrow Y$ es R veces el soporte esperado basado en $X^{\wedge} \Rightarrow Y^{\wedge}$, o la confianza es R veces la confianza esperada basada en $X^{\wedge} \Rightarrow Y^{\wedge}$, y un itemset $X \cup Y$ es R-interesante w.r.t. $X^{\wedge} \cup Y^{\wedge}$.

4.4 Conclusión

El análisis de reglas de asociación puede ser muy útil cuando se esta haciendo análisis exploratorio, buscando relaciones interesantes que pueden existir en el conjunto de datos. El grado de utilidad de las asociaciones identificadas, puede ser usado para ayudar a predecir comportamiento. El mero hecho de que dos cosas ocurran próximas una de otra no es garantía de que las relaciones sean importantes o significativas. Así, probablemente las asociaciones identificadas por este tipo de análisis, pueden necesitar ser estudiadas más cuidadosamente usando otros métodos analíticos para interpretar relaciones.

5 Nuestro Enfoque

5.1 Introducción

Dado un conjunto de transacciones D, el problema de realizar un mining de reglas de asociación es generar todas las reglas de asociación que tengan el soporte y la confianza mayor a la especificada por el usuario, MinSup y MinConf respectivamente. A ésto se agrega que estas reglas deben ser encontradas sobre atributos cuantitativos y categóricos. Llamamos a este proceso: Mining de Reglas de Asociación Cuantitativas.

En el capítulo 4, mostramos como mapear el problema enunciado anteriormente al de las Reglas de Asociación Booleanas. Algunos algoritmos que encuentran este tipo de reglas son: AIS presentado en (Cabena y otros[4]), SETM propuesto en (Fayyad y otros [13]) y, dos nuevos algoritmos Apriori y Apriori Tid introducidos en (Agrawal – Srikant [10]). Este ultimo paper, muestra la performance relativa de estos algoritmos. Como resultado de esas pruebas se concluye que Apriori tiene mejor performance y una excelente escalabilidad en grandes bases de datos.

Nuestra elección del algoritmo (Apriori) a implementar se basa en lo mencionado anteriormente. Una vez seleccionado, teníamos que resolver el problema de poder manejar atributos cuantitativos y categóricos. Esto fue resuelto desarrollando una interface y un procedimiento para el mapeo de estos atributos.

En este capítulo, explicamos el algoritmo elegido para encontrar reglas de asociación, su implementación, la elección del lenguaje y las estructuras utilizadas. Luego desarrollamos nuestra solución al problema de los atributos cuantitativos y categóricos. Por último se mostraran los resultados de los experimentos de performance realizados al software obtenido.

5.2 Nuestro Algoritmo

Nuestro algoritmo se compone de dos grandes procesos. El primero, es básicamente la implementación del algoritmo A priori. Este trabaja con la base de datos generando la información que necesita el proceso que se encargará de encontrar las reglas de asociación (sin recurrir a la Base de Datos).

Como se explica en el capítulo 4, se puede descomponer el problema de descubrir todas las reglas de asociación en cuatro subproblemas. El tercero de ellos es encontrar todos los conjunto de ítems (itemsets) que tengan soporte mayor que MinSup. El soporte para un itemset es el número de transacciones que contienen el itemset. Aquellos que tengan soporte por encima del MinSup especificado por el usuario, son llamados itemsets grandes (large itemsets).

El algoritmo para descubrir los itemsets grandes realiza múltiples pasadas sobre la base de datos. En el primer paso, se cuenta el soporte de los ítems individuales y se determina cuales de ellos son grandes (large). En los pasos siguientes, se empieza con un conjunto *"semilla"* de itemsets grandes, encontrados en el paso anterior. Se usa el conjunto *"semilla"* para generar nuevos y potenciales itemsets grandes,

llamados itemsets *candidatos,* y se cuenta el soporte actual para esos candidatos durante el paso sobre la base de datos. Al final del paso, se determina cuales de los itemsets candidatos son actualmente grandes, y ellos van a ser el conjunto semilla del siguiente paso. Este proceso continúa hasta que no se encuentren más itemsets grandes.

Este algoritmo genera los itemsets candidatos a ser contados en la pasada actual usando sólo los itemsets grandes encontrados en el paso previo -sin considerar las transacciones en la base de datos. La idea básica es que cualquier subconjunto de un itemset grande va a ser grande. Por esto, los itemsets candidatos que tienen k ítems pueden ser generados realizando un *"join"* de los iemsets grandes con (k-1) ítems, y borrando aquellos que contengan cualquier subconjunto que no es grande. Este procedimiento da como resultado un número mucho menor de itemsets candidatos generados.

Asumiremos lo siguiente:

- los itemsets en cada transacción están almacenados lexicográficamente
- llamamos al número de ítems en un itemset *tamaño*, entonces a un itemset de tamaño k *k-itemset*
- un k-itemset se representa: ítem[1] ítem[2] ... ítem[k]
- Asociado con cada itemset hay un contador que guarda su soporte. Este campo es inicializado en cero al crear el itemset
- L_k es el conjunto de todos los k-itemsets grandes. Cada miembro de este conjunto tiene dos campos: i)el itemset candidato y ii)el contador de soporte.
- C_k es el conjunto de todos los k-itemsets candidatos (potenciales imtesets grandes). Cada miembro de este conjunto tiene dos campos: i)el itemset candidato y ii)el contador de soporte.

El pseudocodigo del algoritmo Apriori es:

1. L_1 = { 1-itemsets grandes}
2. para (K=2; mientras $L_{k-1} \neq \varnothing$; k++)
3. C_k = apriori_gen(L_{k-1});
4. para todas las transacciones t $\in$ D hacer
5. C_t = { subconjuntos de C_k en t }
6. para todos los candidatos c $\in$ C_k hacer
7. c.contador ++
8. end
9. L_k = { c $\in$ C_k tal que c.contador $\geq$ MinSup}
10. end
11. Respuesta = $\cup_k L_k$

El primer paso del algoritmo simplemente cuenta las ocurrencias de los ítems para determinar los 1-itemsets grandes. El siguiente paso, llamémoslo k, consiste en dos fases:

1. los (k-1)-itemsets grandes encontrados en el paso (k-1) son usados para generar los itemsets candidatos C_k, utilizando la función apriori_gen
2. luego se escanea la base de datos y el soporte de los candidatos en C_k es contado

Función Apriori_gen, (generación de candidatos). Esta función toma como argumento a L_{k-1}, el conjunto de todos los (k-1)-itemsets grandes. Para ello, primero realiza el paso del *"join"* (join(L_{k-1}, L_{k-1})), cuyo pseudocodigo es:

insert into C_k

select $p.item_1$, $p.item_2$, ..., $p.item_{k-1}$, $q.item_{k-1}$

from L_{k-1} p, L_{k-1} q

where $p.item_1 = q.item_1$, $p.item_2 = q.item_2$,..., $p.item_{k-2} = q.item_{k-2}$, $p.item_{k-1} < q.item_{k-1}$

Luego se realiza el paso de depuración, borrando todos los itemsets $c \in C_k$, tal que algún (k-1)-subconjunto de c no esta en L_{k-1},

para todos los itemsets $c \in C_k$ hacer

para todos los (k-1)-itemsets s de c hacer

si ($s \notin L_{k-1}$) entonces borrar c de C_k

end

end

El cuarto subproblema de la descomposición presentada en el capítulo 4 para descubrir todas las reglas de asociación, es en efecto, generar todas aquellas cuya confianza es mayor o igual a la especificada por el usuario (MinConf).

Hasta aquí vimos el primer gran proceso de nuestro algoritmo, el que genera la información (conjunto de itemsets grandes) de entrada al segundo procedimiento. Este se encarga de resolver el subproblema mencionado arriba.

Para generar las reglas de asociación, tomamos como argumento el conjunto L (salida del primer proceso) que contiene los itemsets grandes y sus soportes. Cada elemento del conjunto L es una lista l_i Cada una de éstas contiene los i-itemsets grandes con sus respectivos contadores de soporte.

Notación:

L = arreglo de n posiciones

L[i] = (lista l_i) conjunto de i-itemsets grandes con su contador de soporte

Tomamos como base la explicación dada en el capítulo 4 sección 4.3.2.1.4.

Comenzamos a recorrer L desde la posición 2 (L[2]), ya que para formar una regla necesitamos tener por lo menos dos ítems. Con cada elemento de l_i , i-itemsets (i > 1), se forman todos los subconjuntos no vacíos del i-itemset que se esta procesando. Para cada subconjunto **a** se busca su soporte en la l_k correspondiente, siendo k el número de ítems en dicho subconjunto. Luego, si:

(Soporte(a) / Soporte(i-itemset)) ≥ MinConf

la regla obtenida es

a → (i-itemset – a)

En esta instancia ya tenemos especificado los dos grandes procesos de nuestro algoritmo. La solución a los subproblemas 3 y 4 de la descomposición dada en el capítulo 4.

Para la implementación de los procedimientos hemos elegido el lenguaje C. El fundamento que nos llevó a esta elección es su capacidad/velocidad de procesamiento por ser de "bajo nivel". Esta ventaja, nos ayuda en le momento de generar todos los subconjuntos y listas dinámicas necesarias.

El código de nuestro algoritmo en C es el siguiente:

```
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <alloc.h>

/*-------------------------------------------------------------------------*/
/* Declaracion de Ctes. necesarias para calcular los espacios de memoria */

 #define LONG_ITEM 4    /* Longitud de cada item */
 #define CANT_ITEMS 100 /* Cantidad maxima de items por transaccion */
 #define MAXIMO_ITEMS_x_ITEMSET 6 /* Cant. max. de items por itemset */

/*-------------------------------------------------------------------------*/

/*elem es una estructura donde guardaremos los k-itemset grandes de cada Li */
/*tambien la vamos a utilizar para la lista Ck, quien contendra los
  k-itemset candidatos */

struct elem {
  char itemset[LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET];
  float soporte;
  struct elem *psig;
};

/* itemset tendra la concatenacion de items segun el valor de k */
/* soporte indicara la cant.de veces que ocurre el itemset en el total de
   transacciones de la base en proceso */

/*-------------------------------------------------------------------------*/

static struct elem *L [LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET];

/* este es el vector resultado, representa la union de todas las Li
   en L[0] estan los 1-itemset, en L[1] los 2-itemset, en L[2] los
   3-itemsets y asi siguiendo */
/*-------------------------------------------------------------------------*/

static unsigned L_items[LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET];

/* es el contador de los itemsets de cada Li */

/*-------------------------------------------------------------------------*/

static struct elem *Ck;
static struct elem *subcjto;
static struct elem *T;
static struct elem *Tk;
static struct elem *Ti;
static unsigned cant_transacc_en_la_base = 0;
static unsigned k = 0;          /* indica el Lk que se esta procesando */
static unsigned Lcant = 0;      /* cantidad de Li generadas */
```

```
static unsigned cant_items = 0; /* es la cantidad de items que leyo en la linea */

/*-------------------------------------------------------------------------*/
/* Parametros recibidos en el archivo "paramet.txt" */

static unsigned K_MAXIMO = 0; /* maximo valor para formar los k-itemset*/
static float  MINCONF = 0.0;
static float  MINSUP  = 0.0;
static unsigned veces = 0;
/*-------------------------------------------------------------------------*/

struct elem *liberar_mem (struct elem *p) {
/*---------------------------------------*/
/* libera la memoria alocada dinamicamente con memalloc */
  struct elem *q;
  static unsigned cant = 0;

  for (; p != NULL; p = q) {
     q = p->psig;
     p->soporte=0.0;
     p->itemset[0]='\0';
     p->psig=NULL;
     free(p);
     ++cant;
  }
  if (cant==0 && p!= NULL)
    p->itemset[0]='\0';

  return NULL;
}

struct elem *add_itemset(char *str, float cont, struct elem *r) {
/*-------------------------------------------------------------*/
/* genera una lista de itemsets alocando mem. dinamicamente si el itemset
  no esta en la lista, y si ya existe incrementa su soporte */

  int cond=0;
  struct elem *ant, *nuevo;
  veces++;
  if (r==NULL) {
        r = malloc(sizeof *r);
        if (r==NULL) {
           printf("NO HAY MEMORIA SUFIENTE PARA ASIGNAR \n");
           exit (4);
        }
        strcpy(r->itemset, str);
        r->soporte = cont;
        r->psig = NULL;
  } else
        if ((cond=strcmp(str,r->itemset))==0) {
            r->soporte = r->soporte + 1.0;
        } else
            if (cond < 0) {
                ant=r;
                nuevo = malloc(sizeof *nuevo);
```

```
            if (nuevo==NULL) {
              printf("NO HAY MAS MEMORIA PARA ASIGNAR \n");
              exit (4);
             }
             strcpy(nuevo->itemset,str);
             nuevo->soporte = cont;
             nuevo->psig = ant;
             r=nuevo;
          } else
            r->psig=add_itemset(str,cont,r->psig);

  return r;
}

struct elem *agregar_items(char *s, struct elem *lista, unsigned long_i) {
/*-------------------------------------------------------------------*/
/* en esta funcion se agregan los items leidos de una transaccion a Ck */

  static  unsigned i=0;
  char un_item[LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET];

  cant_items = (strlen(s) + ((LONG_ITEM * (long_i+1)) - 1)) / LONG_ITEM;

  if (cant_items > CANT_ITEMS) {
        printf("ERROR: cantidad de items supera el numero esperado\n");
        exit(3);
  }

  /* toma de un k-itemset y lo pone en la lista Ck */
  for ( i=0; i < cant_items; i++) {
    sprintf(un_item,"%.*s", (LONG_ITEM * (long_i+1)), s+(i*(LONG_ITEM * (long_i+1))));
    veces=0;
    lista=add_itemset(un_item,0.0, lista);
    /*printf("Veces que hizo recursion en una llamada a add_itemset: %d\n",veces);*/
  }

  return lista;
}


static void depurar_Ck (struct elem *lista) {
/*---------------------------------------*/
/* Para cada itemset de Ck se le divide el soporte por la cant. de transacc.
   de la base que estamos procesando y ese soporte se compara con el MinSup
   especificado por el usuario, aquellos itemsets que lo superan se almacenan
   en el L[k] correspondiente, ya que en el estan los itemsets grandes */

  struct elem *p;

  for (p=lista; p != NULL; p=p->psig) {
     p->soporte = p->soporte / cant_transacc_en_la_base;
     if ( p->soporte > MINSUP) {
           /*printf("Elemento de Ck el_item es: %s soporte: %2.2f\n",p->itemset,p->soporte);*/
           veces=0;
           L[k]=add_itemset(p->itemset,p->soporte,L[k]);
```

```
/*          printf("Veces que hizo recursion en una llamada a add_itemset(2): %d\n",veces);*/
            ++L_items[k];
        }
      }
      if (L_items[k] > 0)
        ++Lcant;
}

int pertenece(char *str, struct elem *p) {
/*---------------------------------------*/
/* busca un elemento en una lista,
   retorna 1 si lo encuentra y 0 en caso contrario */

   for (; p != NULL; p=p->psig) {
      if (!strcmp(str, p->itemset))
           return 1;
   }
   return 0;
}

struct elem *borrar_itemset (char *str, struct elem *r) {
/*-----------------------------------------------------*/
/* borra un itemset de la lista r, esta funcion es llamada
   cuando se depura a Ck */

   struct elem *act,*ant,*p;

   if (strcmp(str,r->itemset)==0) {
      r=r->psig;
   }
   else {
     ant=r;
     for (act=r; act!=NULL; act=act->psig) {
           if (strcmp(str,act->itemset)==0) {
             ant->psig=act->psig;
             break;
           }
           ant=act;
     }
   }
   return r;
}

static void apriori_gen () {
/*-------------------------*/
   static int i=0;
   static unsigned j=0;
   struct elem *p,*q,*l;
   /* item_p e item_q tendran la primera parte del itemset */
   char item_p[LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET];
   char item_q[LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET];
   /* item_p_pos_k e item_q_pos_k tendran respectivamente los ultimos items
    a comparar en el join */
   char item_p_pos_k[LONG_ITEM + 1];
```

```
char item_q_pos_k[LONG_ITEM + 1];
static char item_i[LONG_ITEM + 1];
static char un_itemset[LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET];
static char el_itemset[LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET];
char item_auxi [LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET];

Ck=liberar_mem(Ck);  /* liberamos la memoria alocada de la lista Ck */

/* ------------------------ JOIN ---------------------------------*/

for (p=L[k]; p != NULL; p=p->psig) {
      for (q=L[k]; q != NULL; q=q->psig) {
          item_p[0] = '\0';
          item_q[0] = '\0';
          if (k > 0) {
               sprintf(item_p,"%.*s", (LONG_ITEM * k), p->itemset);
               sprintf(item_q,"%.*s", (LONG_ITEM * k), q->itemset);
          }
          if (strcmp(item_p,item_q) < 0)
               break;
          sprintf(item_p_pos_k,"%.*s", (LONG_ITEM * (k+1)), p->itemset+(LONG_ITEM * k));
          sprintf(item_q_pos_k,"%.*s", (LONG_ITEM * (k+1)), q->itemset+(LONG_ITEM * k));
          if (strcmp(item_p,item_q)==0 && strcmp(item_p_pos_k,item_q_pos_k) < 0) {
               strcpy(item_auxi,p->itemset);
               strcat(item_auxi,item_q_pos_k);
               veces=0;
               Ck=add_itemset(item_auxi,0.0,Ck); /* genera Ck */
               /*printf("Veces que hizo recursion en una llamada a add_itemset(3): %d\n",veces);*/
          }
      }
}
/*------------------------ Depuracion---------------------------------*/

 for (l=Ck; l != NULL; l = l->psig) {
   strcpy(el_itemset,l->itemset);
   if (el_itemset[0] == '\0')
        break;

   /* formamos los subcjtos de el_itemset de Ck con (k-1) elementos */
   /* (k+1) indica la cantidad de items que tiene cada itemset */
   subcjto=liberar_mem(subcjto);
   for (i=k+1; i >= 0; --i) {
        un_itemset[0]='\0';
        for (j=0; j <= (k+1); j++) {
           if (i != j) {
                sprintf(item_i,"%.*s", LONG_ITEM , el_itemset + (j * LONG_ITEM ));
                strcat(un_itemset, item_i);
           }
        }
        veces=0;
        subcjto=add_itemset(un_itemset,0.0, subcjto);

        /*printf("Veces que hizo recursion en una llamada a add_itemset(4): %d\n",veces);*/
   }
```

```
      for (p=subcjto; p != NULL; p=p->psig) {
            if (pertenece(p->itemset,L[k]) == 0) { /*retorna 0 si no pertenece, 1 en c.c. */
              /* borra todo el_itemset de Ck porque el item que esta en
                   p->itemset no pertenece a L[k]); */
              Ck=borrar_itemset(el_itemset, Ck);
              break;
            }
      }
      subcjto=liberar_mem(subcjto);
    }
}

struct elem *generar_subcjtos(struct elem *C_n, struct elem *C_1, struct elem *C_rta) {
/*-------------------------------------------------------------------------------*/

   char item_auxi [LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET];
   char item_q_pos_k [LONG_ITEM + 1];
   struct elem *p;
   struct elem *q;
   unsigned j;

   for (q=C_n; q != NULL; q=q->psig) {
      j = (strlen(q->itemset) / LONG_ITEM) - 1;
      for (p=C_1; p != NULL; p=p->psig) {
         sprintf(item_q_pos_k,"%.*s", (LONG_ITEM * (j+1)), q->itemset+(LONG_ITEM * j));
         if (strcmp(p->itemset,item_q_pos_k) > 0) {
            strcpy(item_auxi,q->itemset);
            strcat(item_auxi,p->itemset);
            veces=0;
            C_rta=add_itemset(item_auxi,0.0,C_rta); /* genera C_rta */
            /*printf("Veces que hizo recursion en una llamada a add_itemset(5): %d\n",veces);*/
         }
      }
   }

   return C_rta;
}

static void subset (char *s) {
/*-------------------------*/

   char item_q_pos_k[LONG_ITEM + 1];
   static unsigned long_i=0;
   static unsigned j=0;
   struct elem *p;
   struct elem *q;
   char item_auxi [LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET];

   Tk=liberar_mem(Tk);
   T=liberar_mem(T);
   Ti=liberar_mem(Ti);
   /* formamos los subcjtos de la transaccion leida t con (k-1) elementos */
   /* transforma la transaccion leida s a lista de items */
   T=agregar_items(s,T,long_i); /* setea la vble cant_items */
```

```
   if (cant_items > (k+1)) { /* cumplen la cant.minima para generar
                                itemset de k items */

     /* T tiene el L1 de la transaccion leida y Ti va teniendo las Li
        de la misma transaccion leida */

     for (p=T; p != NULL; p=p->psig){
           veces=0;
         Ti=add_itemset(p->itemset,p->soporte,Ti);

         /*printf("Veces que hizo recursion en una llamada a add_itemset(6): %d\n",veces);*/
     }

     for (j=0; j <= k; j++) {
         Tk=generar_subcjtos(Ti,T,Tk);
         Ti=liberar_mem(Ti);
         for (p=Tk; p != NULL; p = p->psig){
               veces=0;
            Ti=add_itemset(p->itemset,p->soporte,Ti);

            /*printf("Veces que hizo recursion en una llamada a add_itemset(7): %d\n",veces);*/
         }

         Tk=liberar_mem(Tk);
     }


     for (p = Ck; p != NULL; p = p->psig) {

         for (q=Ti; q != NULL; q = q->psig) {
            if (strcmp(p->itemset,q->itemset) == 0)  {
                p->soporte = p->soporte + 1.0;
                break;
            }
         }
     }
   }
}

float buscar_soporte(struct elem *Lj, char *a_itemset) {
/*-------------------------------------------------*/

   struct elem *p;

   for (p=Lj; p != NULL; p=p->psig) {
      if (!strcmp(a_itemset, p->itemset))
          return p->soporte;
      }
   return (0.00);
}
```

```
char * generar_consecuente(struct elem *Cjto_1,struct elem *subcjto_a) {
/*-----------------------------------------------------------------*/
/* genera el consecuente de la regla con los itemsets del Cjto_1 que
   no estan en el subcjto_a */

   struct elem *p;
   struct elem *s;
   unsigned esta=0;
   static char q [LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET]; /* OJO ERA *Q */

   q[0]='\0';
   for (p=Cjto_1; p != NULL; p=p->psig) {
      esta=0;
      for (s=subcjto_a; s!=NULL; s=s->psig) {
            if (strcmp(p->itemset,s->itemset)==0) {
              esta=1;
              break;
            }
      }
      if (esta==0)
           strcat(q,p->itemset);
   }
   return q;
}


static void generar_reglas(struct elem *itemset_l, FILE *freglas) {
/*-----------------------------------------------------------------*/
/* genera todos los subconjuntos del itemset recibido, para cada
   subcjto no vacio busca su soporte en la L[k] correspondiente y si
   el soporte del itemset dividido el soporte del subcjto generado es
   mayor o igual a MinConf entonces genera la regla con antecedente
   igual al subcjto generado y el consecuente con los items del itemset
   recibido que no estan en el subcjto que genero , este proceso se repite
   para todos los subcjtos del itemset recibido */

   float soporte_l =0.0;
   float soporte_a =0.0;
   int long_i = 0;
   unsigned j=0;
   static char *q[LONG_ITEM * MAXIMO_ITEMS_x_ITEMSET];
   struct elem *p;
   struct elem *a;

   T=liberar_mem(T);
   Ti=liberar_mem(Ti);
   Tk=liberar_mem(Tk);
   soporte_l = itemset_l->soporte;
   T =agregar_items(itemset_l->itemset,T,long_i);
   Tk=agregar_items(itemset_l->itemset,Tk,long_i);

   for (j=0; j < k; j++) {
      for (a=Tk; a != NULL; a=a->psig) {
         soporte_a = buscar_soporte(L[j], a->itemset);
         if (soporte_a > 0.00) {
```

```
            subcjto=liberar_mem(subcjto);
            if (((soporte_I / soporte_a)*100) >= MINCONF) {

              subcjto=agregar_items(a->itemset,subcjto,long_i);
              *q=generar_consecuente(T,subcjto);
              fprintf(freglas,"%s %s %2.2f\n",a->itemset,*q,(soporte_I / soporte_a)*100);
            }
          }
        }

        Ti=liberar_mem(Ti);
        for (p=Tk; p != NULL; p=p->psig){
            veces=0;
            Ti=add_itemset(p->itemset,p->soporte,Ti);
            /*printf("Veces que hizo recursion en una llamada a add_itemset(8): %d\n",veces);*/
            }

        Tk=liberar_mem(Tk);
        Tk=generar_subcjtos(Ti, T, Tk);
    }

}


#pragma argsused

int main (int argc, char *argv[]) {
/*-------------------------------*/

    /* Definicion un archivo de salida para grabar las reglas encontradas */
    FILE *freglas;
    char *file_salida = "reglas.txt";

    /* Definicion del archivo de entrada a este algoritmo, el que tiene la
       base con las transacciones a procesar */
    FILE *fp;
    char *fname = "transacc.txt";

    char  items_tran [LONG_ITEM * CANT_ITEMS];
    struct elem *auxi;
    struct elem *p;
    struct elem *Lk;  /* es la lista que se armara con los
                         k-itemset grandes */

    /*-----------------------------------------------------------------*/
    /* Definicion del archivo de parametros Minsup, Minconf y K-maximo */
    FILE *parametros;
    char *file_input = "paramet.txt";
    if ((parametros = fopen(file_input,"r")) == NULL){
          perror(file_input);
          return errno;
    }
    fscanf(parametros,"%f %f %d", &MINCONF, &MINSUP, &K_MAXIMO);
    close(parametros);
    /*-----------------------------------------------------------------*/
```

```
if ((fp= fopen(fname,"r")) == NULL) {
        perror(fname);
        return errno;
}

/* Lee una vez la Base para generar a L1 (contiene los 1-itemset Grandes) */
/* ------------------------------------------------------------------------*/
Ck=NULL;
while ( fscanf(fp,"%s",items_tran) != EOF) {
        if ((strlen(items_tran) % LONG_ITEM) == 0) {
           Ck=agregar_items(items_tran,Ck,k);
           /* agrega los 1-itemset candidatos a Ck para k=1 */
           cant_transacc_en_la_base ++;
           }
        else {
            printf("%u %u \n",strlen(items_tran), (strlen(items_tran) % LONG_ITEM));
            printf("Registro %s ignorado\n", items_tran);
        }
}

fclose(fp);
for (p=Ck; p != NULL; p=p->psig)
   p->soporte = p->soporte + 1.0;
   /* sumamos 1 al soporte porque estaba inicializado
        en cero cuando se creo C1 */

depurar_Ck(Ck); /* esta funcion recalcula el soporte de los itemsets y
                   saca de Ck los que no tienen el Minimo Soporte */


/* ------------------------------------------------------------------------*/

 while (( k < K_MAXIMO - 1) && (L_items[k] != 0)) {

       apriori_gen();

       /* lee nuevamente la BD y para cada transaccion "t" llama a subset*/
       if ((fp= fopen(fname,"r")) == NULL) {
                perror(fname);
                return errno;
       }
       while ( fscanf(fp,"%s",items_tran) != EOF) {
                if ((strlen(items_tran) % LONG_ITEM) == 0)  {
                  subset(items_tran);
                }
       }
       fclose(fp);
       ++k;


       depurar_Ck(Ck); /* esta funcion recalcula el soporte de los itemsets y
                          saca de Ck los que no tienen el Minimo Soporte */
}
```

```
  /* ----------------------------------------------------------------*/
  /* MOSTRAMOS LOS ITEMSET GRANDES GENERADOS */
  for (k=0; k < Lcant; k++) {
     printf("\n\nCantidad de itemset de L[%u] = %u \n\n",k+1,L_items[k]);
     for (auxi=L[k]; auxi != NULL; auxi=auxi->psig)
        printf("\tItemset: %s \t Soporte : (%2.2f) \n",auxi->itemset,auxi->soporte);
  }
  /* ----------------------------------------------------------------*/
  /* GENERAMOS LAS REGLAS  */
  if ((freglas= fopen(file_salida,"w")) == NULL) {
        perror(file_salida);
        return errno;
  }
  for (k=1; k < Lcant; k++) {
     for (auxi=L[k]; auxi != NULL; auxi=auxi->psig){
        generar_reglas(auxi,freglas);
     }
  }
  fclose(freglas);

  return 0;
}
```

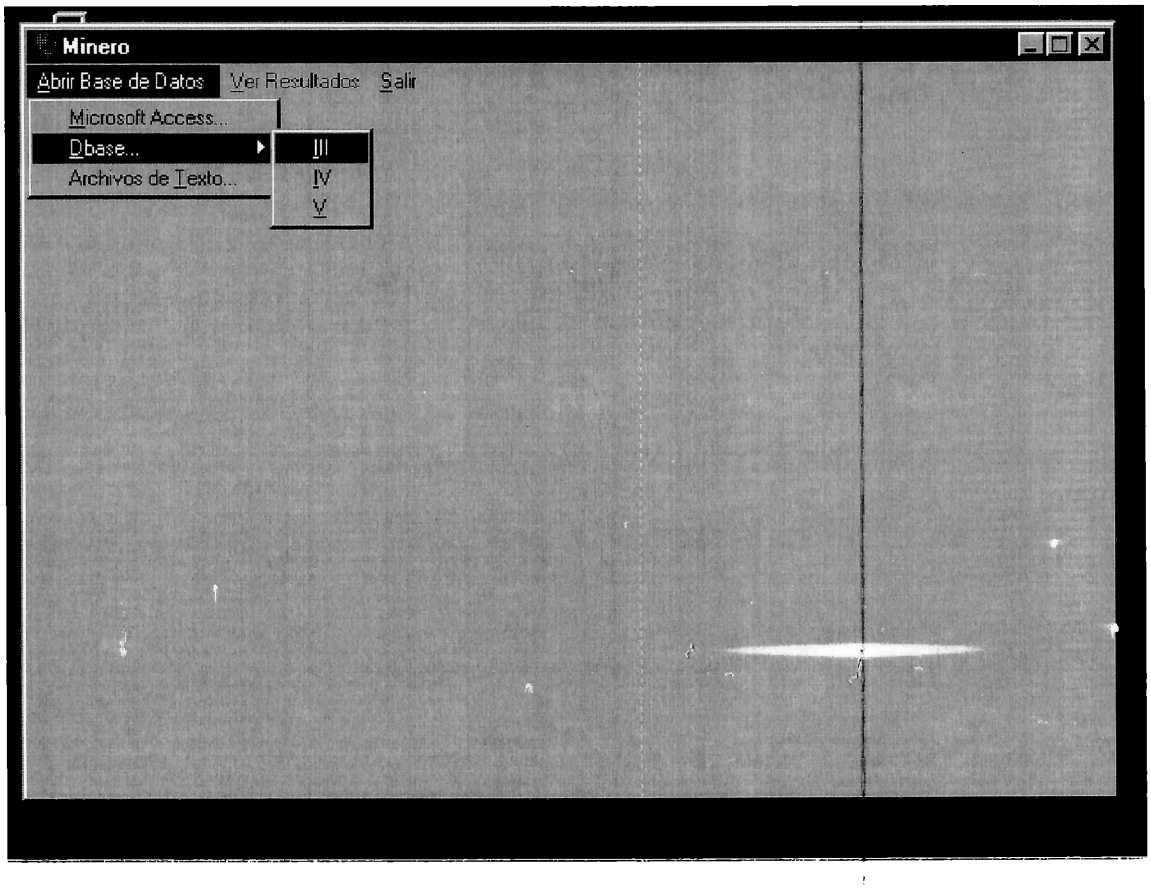
5.3 Procesamiento de los datos de entrada al Algoritmo

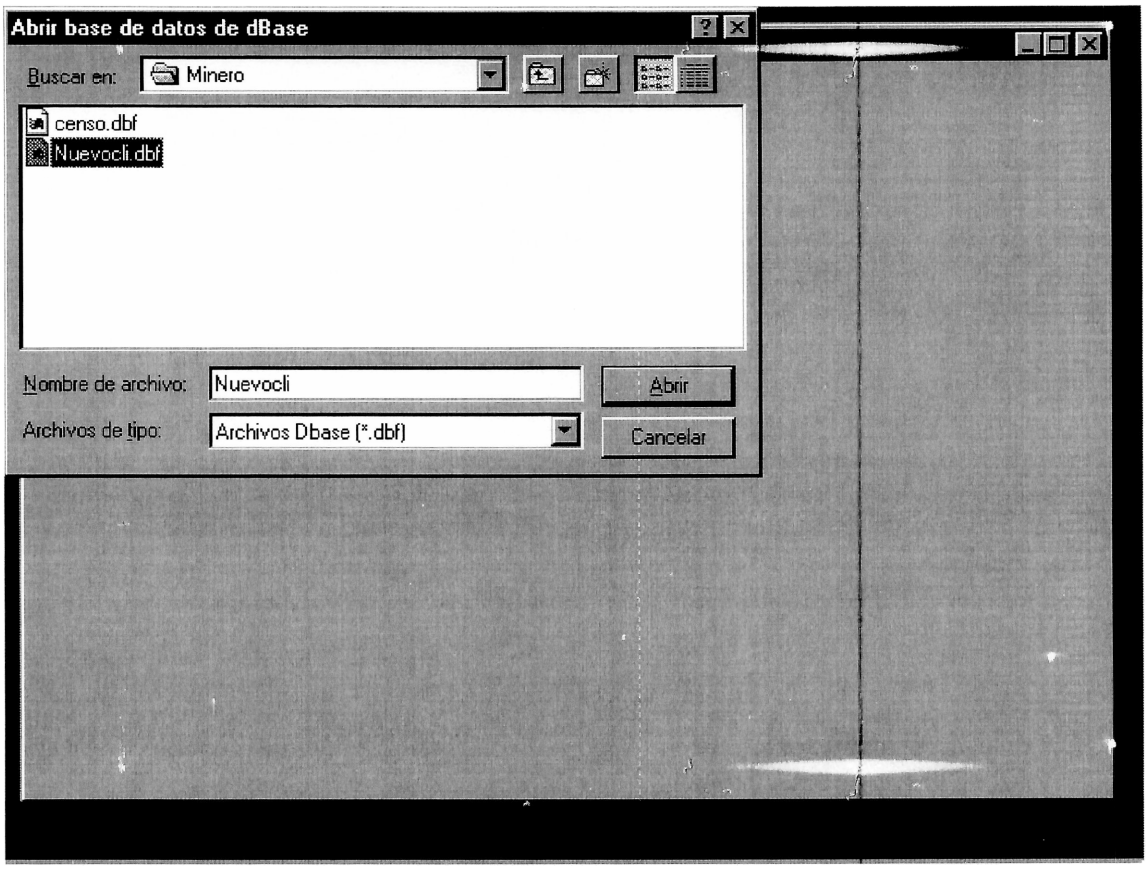
Para resolver el manejo de atributos cuantitativos y categóricos realizamos una interface y un procedimiento, cuyo resultado es el archivo de entrada a nuestro algoritmo (5.2). Además visualiza las reglas de asociación obtenidas de la ejecución de la rutina en C.

La interface tiene dos funciones: la primera es la interacción para la selección de la Base de Datos, los campos que intervienen en el mining, el seteo de MinConf, MinSup y el parámetro K. Este último indica el tamaño máximo de los itemsets a considerar. La segunda es la visualización de las reglas.

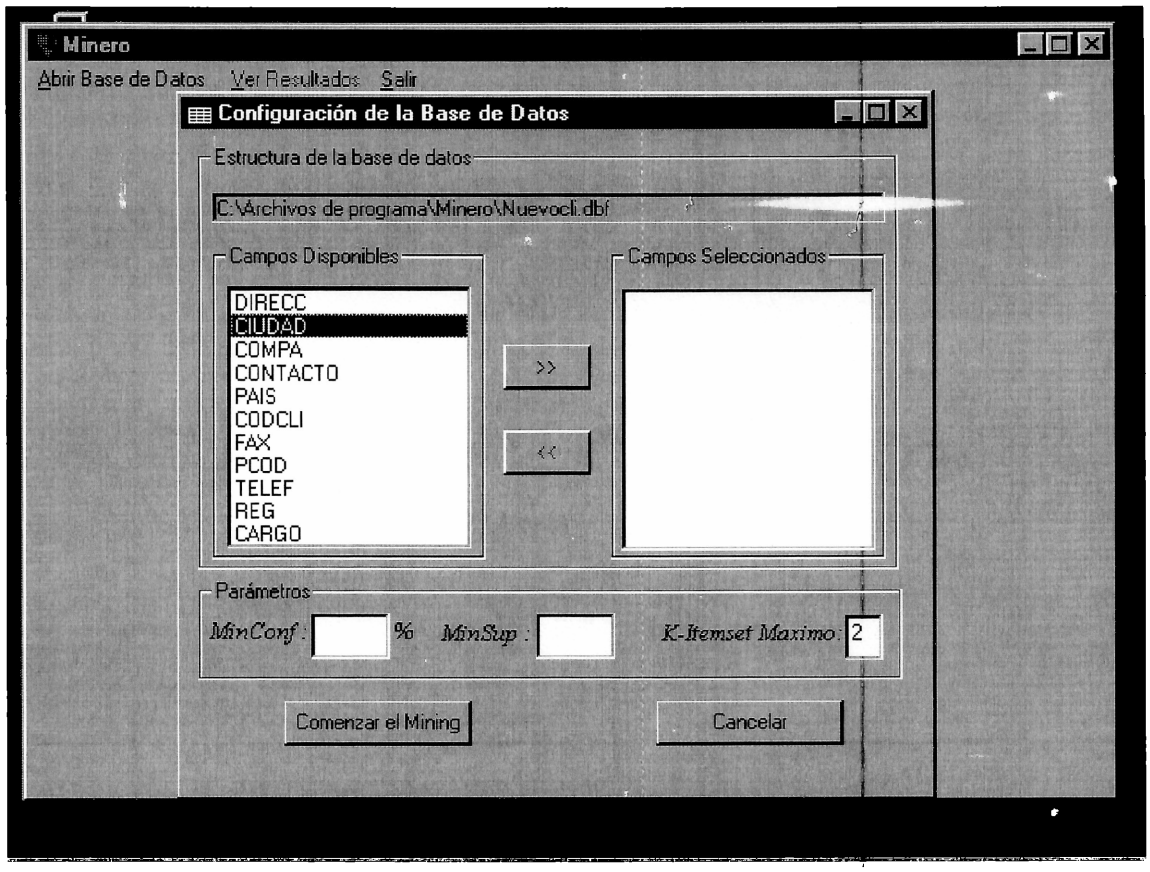
En detalle:

Selección y apertura de la Base de Datos con la que se desea trabajar

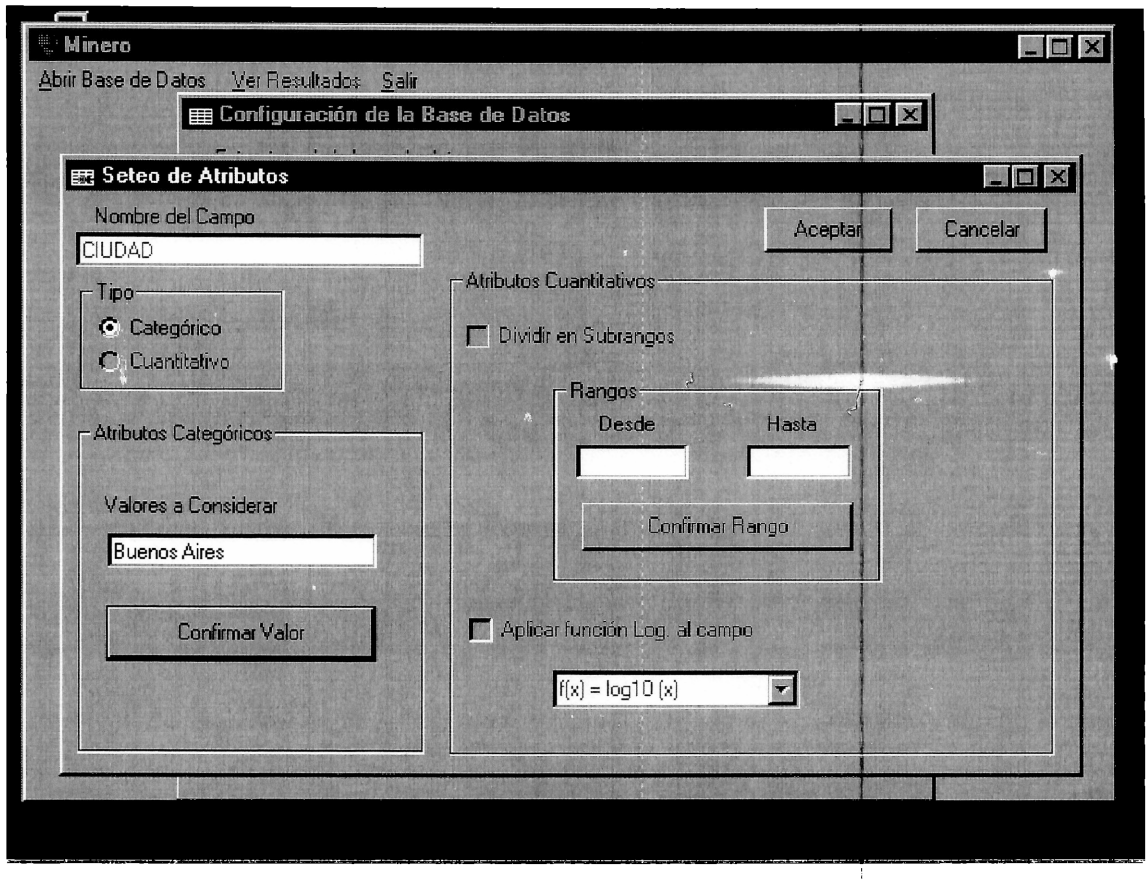




Especificación de los campos a usar en el mining.



Por cada campo elegido se activa la siguiente ventana



Aquí se debe definir el tipo del campo: categórico o cuantitativo.

Si es Categórico, existen dos posibilidades para el mapeo:

i) especificar los valores de interés en forma manual

ii) utilizar todos los valores del campo disponibles en la Base de Datos. Esto se realiza automáticamente.

Si es Cuantitativo:

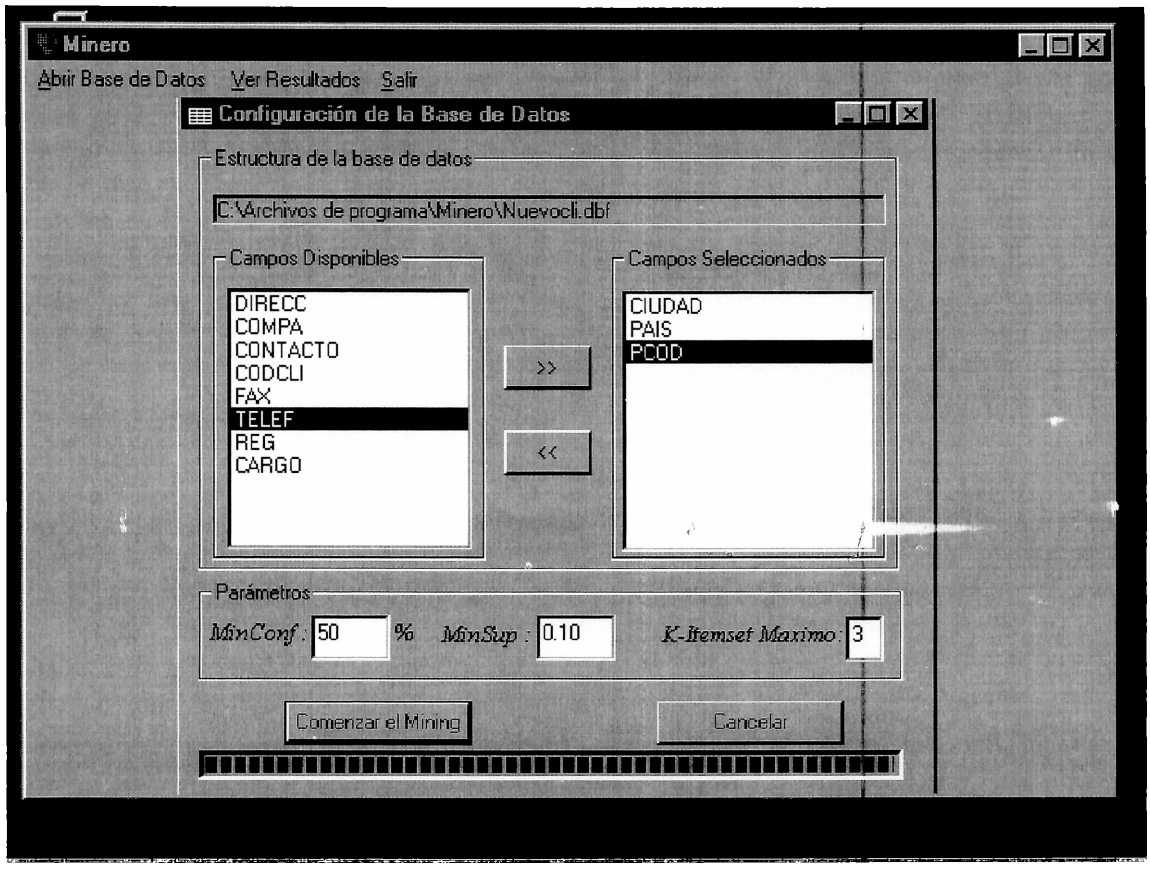
i) el mapeo es directo, similar al ítem ii) para atributos categóricos

ii) especificar manualmente los rangos que se desean para el mapeo

iii) aplicar una función de las disponibles para realizar otro tipo de mapeo, que reduce la cantidad de valores generados

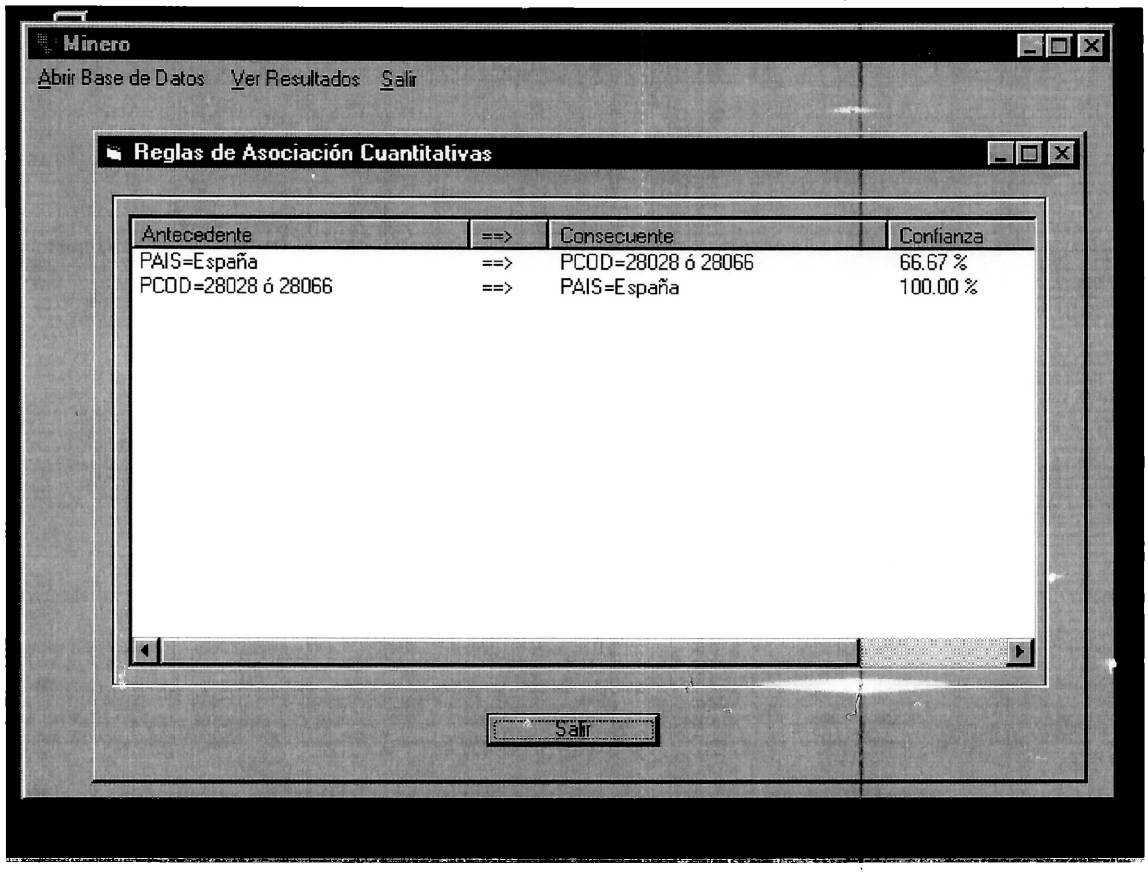
A medida que se interactúa con la interface, para cada campo elegido y especificado, se dispara un procedimiento. Este genera la tabla para el campo en cuestión, la cual contendrá la codificación del mismo.

Una vez que se seleccionan todos los campos de interés, se debe setear los parámetros MinConf, MinSup, y K.



Cuando se le da comienzo al mining, se recorre toda la Base de Datos. Para cada registro, los campos seleccionados son codificados según su valor en las tablas correspondientes. Con dicha codificación se genera una transacción (tendremos una por cada registro leído).

El conjunto de transacciones generadas es el archivo de entrada a nuestro algoritmo (escrito en C). Finalizada su ejecución, obtendremos un conjunto de reglas. Estas deben ser decodificadas con las tablas anteriores de los campos para su visualización.



5.4 Experimentos de performance

Generamos transacciones para evaluar la escalabilidad del algoritmo sobre una base de datos sintéticos que tienen las siguientes características:

- La cantidad de transacciones en la base es de 100
- Para cada transacción hay un máximo de ítems predefinido en 25
- En toda la base, la cantidad máxima de ítems distintos es 100

A partir de éstos valores se generaron las transacciones. Cada una de ellas contiene un número de ítems, éste es especificado por una función random con ***distribución uniforme***, la cual devuelve un valor entre 0 y 1 que es multiplicado por el máximo de ítems por transacción definido (25). Por ésto, la cantidad de ítems de todas las transacciones varían uniformemente entre 1 y 25.

Una vez obtenida la cantidad de ítems para generar una transacción, el valor para cada uno de ellos lo define una ***distribución normal*** con media=50 y un desvío de 15. La elección de la media tiene relación con la cantidad máxima de ítems en una transacción y la cantidad de ítems distintos en toda la base. La idea es que la media=50 nos indica que el ítem con valor 50 es el más frecuente, y con desvío=15 se genera una mayor cantidad de ítems con valores en el rango (50-15)..(50+15), asegurándonos la existencia de itemsets grandes.

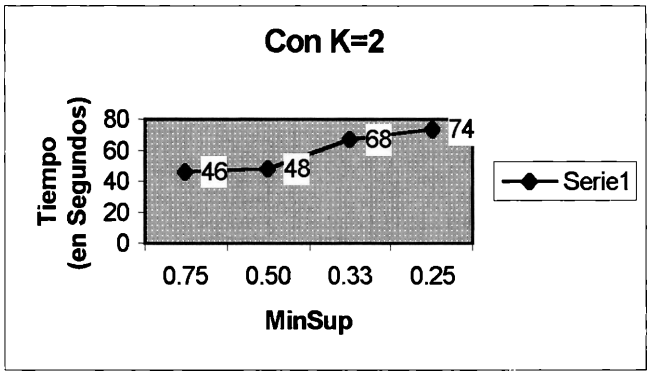
El tamaño de la base de datos generada es de 27Kbytes.

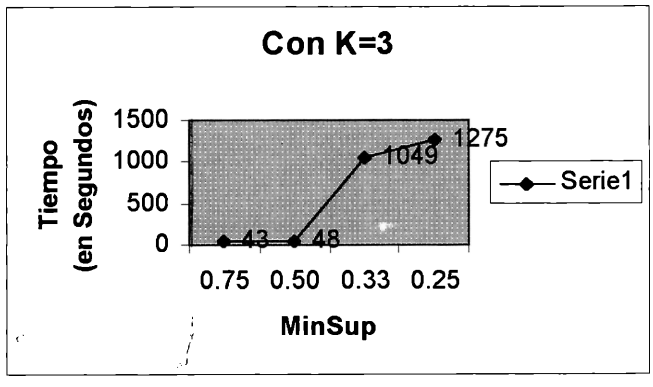
Las siguientes pruebas fueron realizadas sobre un equipo Olivetti Distributed Network Processor, con sistema operativo Unix System V release 4.

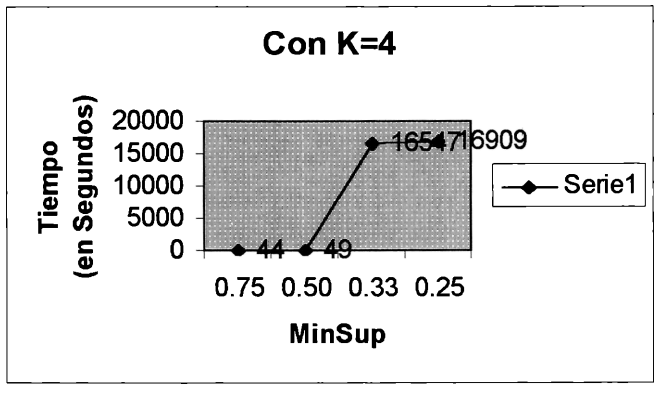
Realizamos dos clases de experimentos, fijando el parámetro K (cantidad máxima de ítems en un itemset frecuente) en 2, 3 y 4; tomando dos tipos de medidas:

1. Tiempo de Ejecución variando el Mínimo Soporte requerido para un itemset frecuente.
2. Cantidad de reglas de asociación obtenidas variando el Mínimo Soporte requerido para un itemset frecuente.

Gráficas - Tiempo de Ejecución



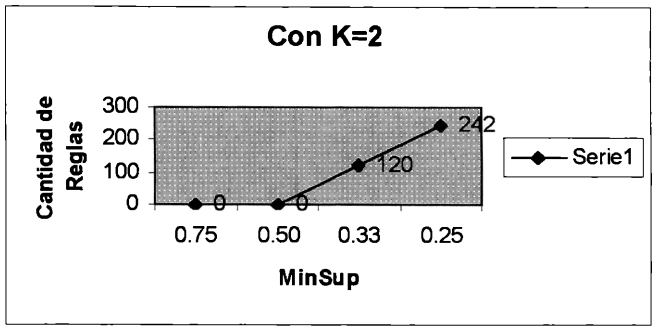


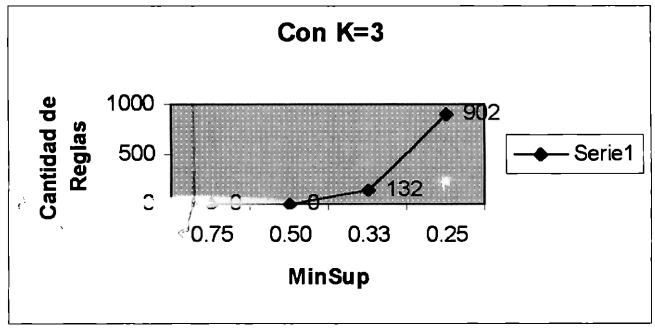


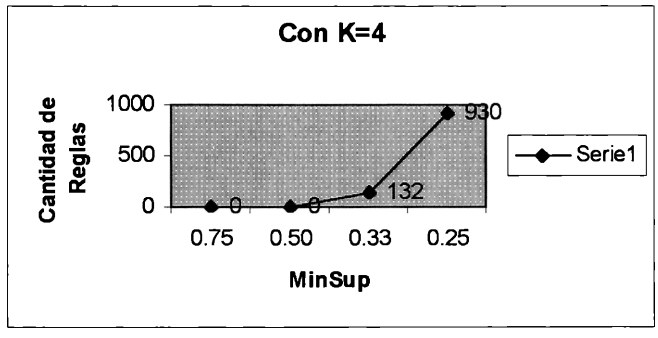
Las gráficas muestran que con valores altos de MinSup: 0.75 y 0.50, casi no existen diferencias de tiempos para los distintos valores de K.

Si observamos el MinSup = 0.33, con K=2 tardó 68 segundos, con k=3 el tiempo se elevó a 17 minutos aproximadamente y con K=4 se llegó a más de 4 horas. Con MinSup = 025, con K=2 llevó 74 segundos, con k=3 se elevó a 21 minutos y con K=4 superó las 4 horas. En conclusión, se nota un incremento importante del tiempo para valores chicos de MinSup a medida que el valor de K se agranda.

Gráficas - Cantidad de Reglas de Asociación







En éstos gráficos se puede ver que no existen en la base itemsets que tengan soporte alto, como 0.75 y 0.50; ya que para esos valores no se han generado reglas. Sin embargo, para MinSup = 0.33 se han generado 120 reglas con k=2 y 132 reglas para los demás valores de k. Con MinSup = 0.25, con k=2 la cantidad de reglas es de 242, más que el doble de cuando MinSup era 0.33. Con k=3 y k=4 para MinSup=0.25 la cantidad de regias es muy parecida, y si comparamos estos valores con los que obtuvimos para MinSup=0.33, la cantidad de reglas se incrementó siete veces más. Se concluye que a medida de que se achica el valor de MinSup, aumenta en gran proporción la cantidad de reglas obtenidas.

CONCLUSIONES

Uno de los puntos de nuestro trabajo fue dar una apreciación de la importancia y potencialidad de Data Mining. Dar una idea clara de qué puede hacer.

En esencia, Data Mining, es un conjunto de técnicas que nos permite acceder a la información que está oculta en nuestras bases. En grandes bases de datos, especialmente, es extremadamente importante tomar la información apropiada, segura y útil, que no se puede encontrar con herramientas estándares de SQL. Usar las técnicas de Data Mining involucra más que el simple uso de redes neuronales o algoritmos genéticos. Debemos adoptar una aproximación paso a paso. Los objetivos deben ser identificados y los datos preparados para el análisis que queremos realizar.

Hay muchos casos, donde una simple herramienta de SQL o de estadística puede proveernos de la información que deseamos, pero en otros casos no la obtenemos sin Data Mining.

La elección de la técnica correcta de Data Mining no puede ser tomada inconscientemente, ya que el 20% de los datos ocultos e inaccesibles son muy significativos. Sin Data Mining, ciertas predicciones y análisis simplemente no son posibles.

El segundo punto de nuestro trabajo fue plantear el problema de encontrar reglas de asociación, en bases de datos que contengan atributos categóricos y cuantitativos. A partir de ésto desarrollamos una aproximación que nos permite resolver lo planteado.

Apéndice A – Un Ejemplo Práctico

Desarrollaremos un ejemplo práctico sobre una base de datos con información de un censo nacional.

Debido a limitaciones del hardware y para lograr una mayor claridad en el ejemplo, la tabla a utilizar contiene los registros necesarios. Los campos de la misma fueron tomados de un formulario del censo nacional realizado en al año 1991. Este censo se realizó tomando como unidad cada inmueble (hogar) y luego las personas que habitan en él. Vamos a trabajar con la tabla Personas, de la base de datos del censo.

La tabla Personas contiene la información recogida, distribuida en los siguientes campos:

Nombre y Apellido

DNI/LC/LE

Dirección

Edad

Sexo

Estado Civil

Cantidad de Hijos

Ocupación

Nivel de Instrucción

Ingresos

Los datos utilizados se muestran a continuación:

NYAP	DNI	DIR	EDAD	SEXO	ESTADO CIVIL	CANT HIJOS	OCUPACION	NIVEL_INST	SUELDO
Carola Marin	23511428	2 n 1470	26	F	Soltero	0	Empleado	Secundario	500
Marcelo Mathieu	16258633	3 n 1470	34	M	Soltero	0	Empleado	Terciario Inc	800
Guadalupe Marin	24125463	4 n 1470	22	F	Soltero	0	Estudiante	Secundario	400
Nilda Massoni	9125442	5 n 1470	52	F	Casado	3	Profesional	Universitario	750
Hugo Marin	9255345	6 n 1470	53	M	Casado	3	Profesional	Universitario	750
Dionisio Marin	24125002	7 n 1470	24	M	Soltero	0	Profesional	Universitario	600
Florencia Vera	23566111	8 n 1470	26	F	Soltero	0	Profesional	Universitario	800
Lorena Fontanari	22349080	9 n 1470	27	F	Soltero	0	Estudiante	Secundario	408
Dario Arriaga	21544375	10 n 147	28	M	Soltero	0	Contratista	Secundario	650
Eduardo Rodrigu	12345666	11 n 147	38	M	Casado	1	Empleado	Terciario Inc	410
Marcela Neme	13111223	12 n 147	36	F	Casado	1	Empleado	Secundario	500
Celeste Flessat	10554875	13 n 147	39	F	Soltero	0	Empleado	Secundario	400
Hernan de Paz	19885336	14 n 147	32	M	Soltero	0	Empleado	Secundario	350
Alenjandra Bren	13554336	15 n 147	36	F	Casado	1	Empleado	Secundario Inc	330
Claudia Minje	12345666	16 n 147	38	F	Casado	2	Empleado	Primario	250
Enrique Marino	10225648	17 n 147	46	M	Soltero	0	Empleado	Secundario	350
Gustavo Robinet	16246872	18 n 147	34	M	Casado	0	Empleado	Secundario	350
Graciela Paular	10546871	19 n 147	40	F	Soltero	0	Empleado	Secundario Inc	400

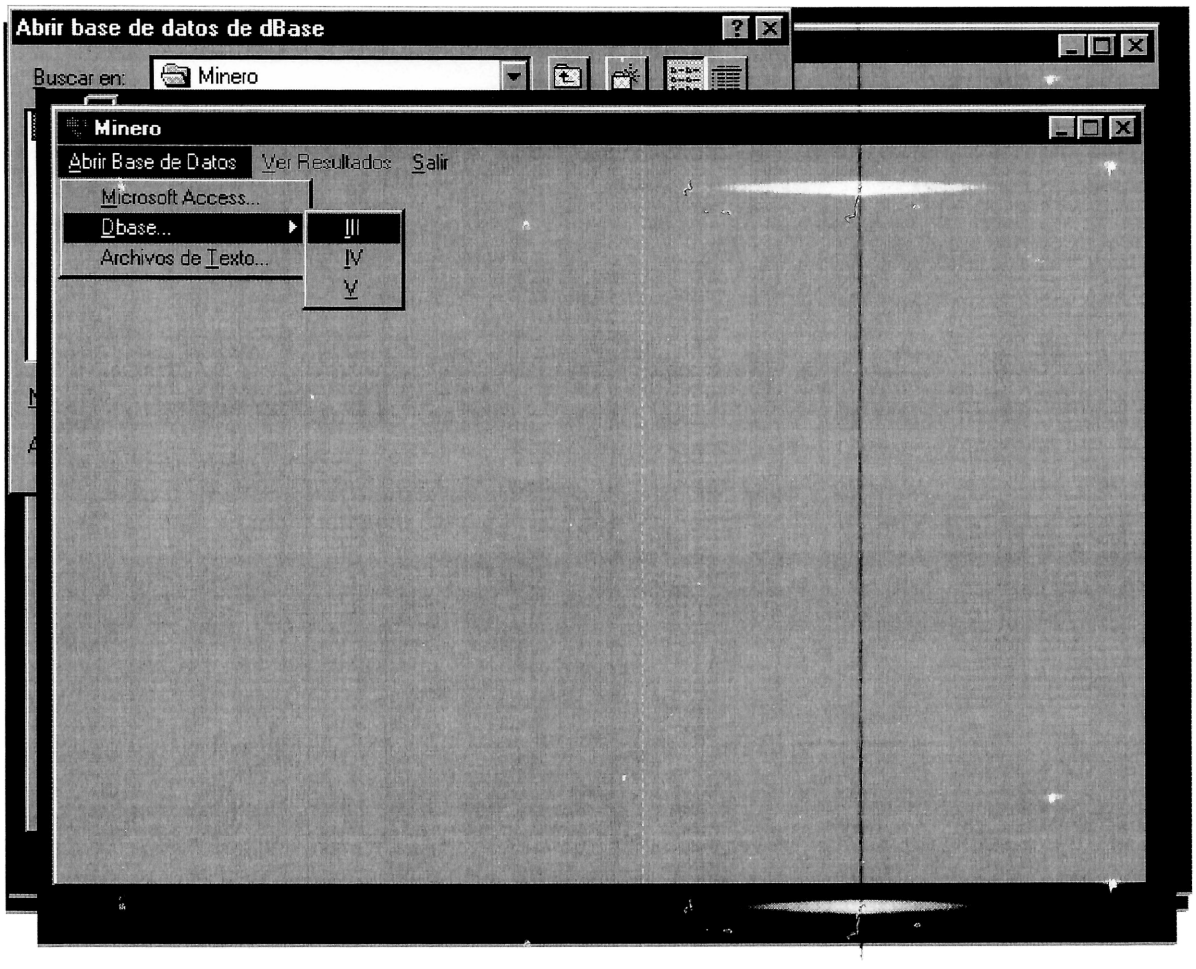
Antes de comenzar con el mining debemos tener especificados nuestros objetivos. Como se vio en él capítulo 2 estos no deben ser generales, sino claros, precisos y detallados.

El objetivo general es obtener información sobre el nivel de alfabetización de la población de adultos (mayores de 30 años) en ambos sexos. Para lograr esto definimos tres objetivos con las características mencionadas anteriormente:

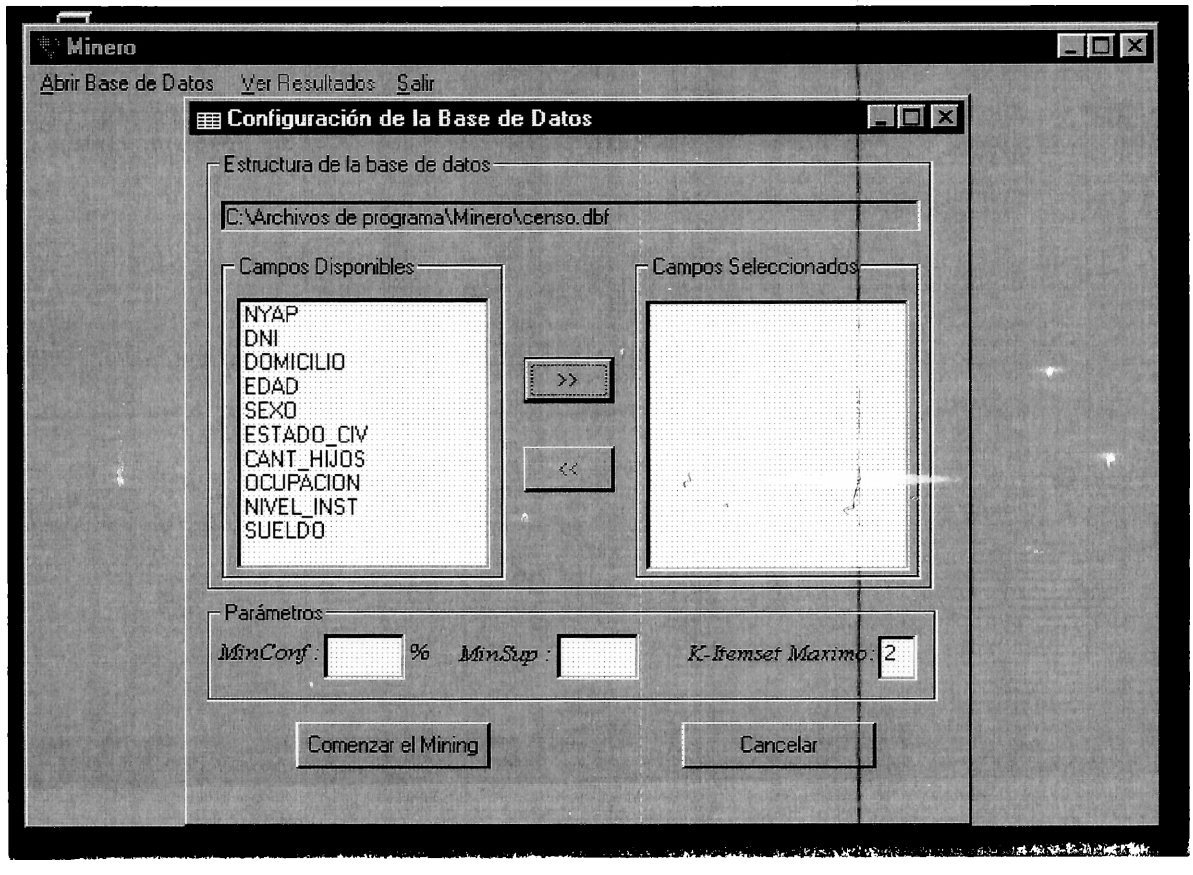
1. Qué nivel de instrucción tiene la población por sexo
2. Qué nivel de instrucción tiene la población por edad
3. Qué nivel de instrucción tiene la población por sexo y edad

Ya definidos los objetivos, empezamos a interactuar con el software desarrollado.

El primer paso es abrir la tabla Personas:



Todos los campos que contiene esta tabla son visualizados automáticamente en la siguiente ventana:



El siguiente paso es identificar los campos necesarios para cumplir con los objetivos. Comenzando de esta manera la etapa de preparación de los datos que se necesitan para el mining.

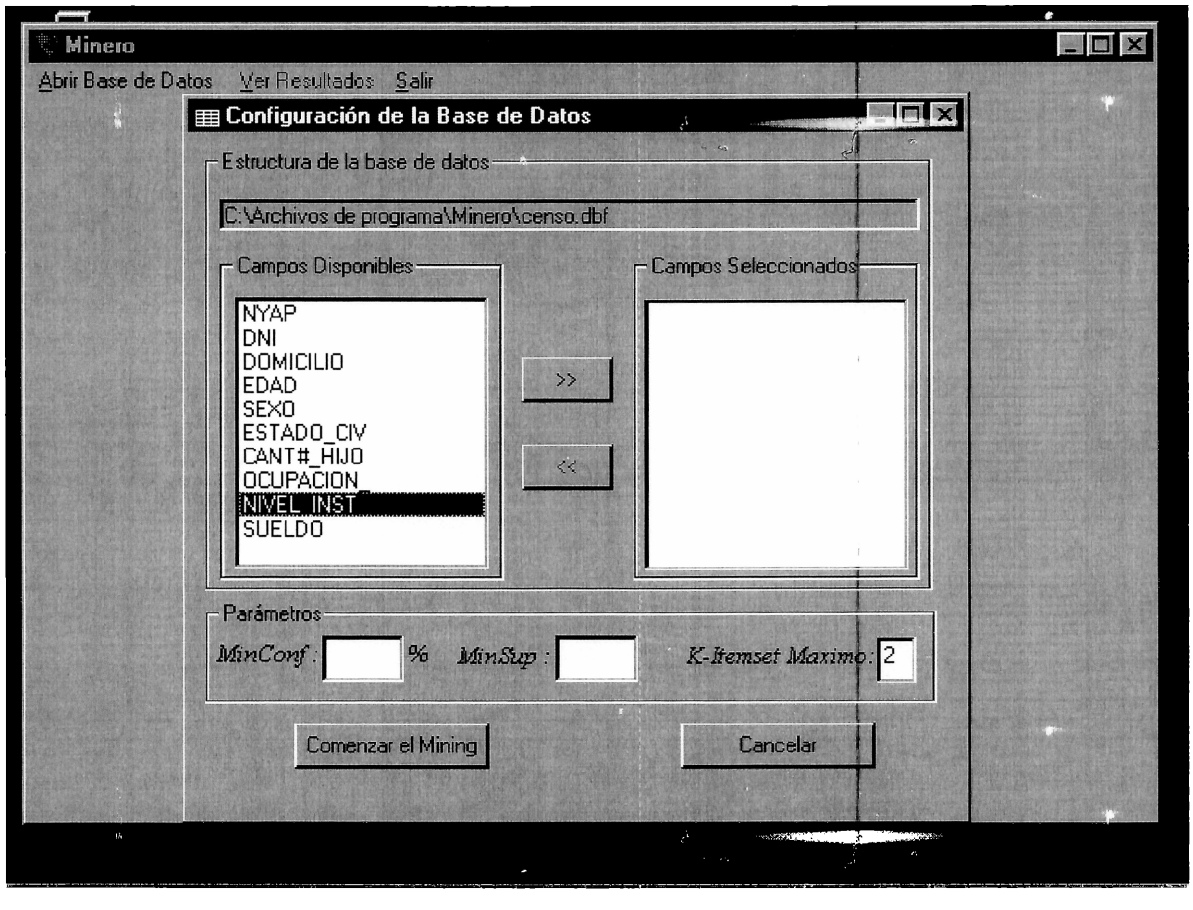
Esta etapa se compone de dos fases:

1. Se selecciona uno de los campos ***(Nivel de Instrucción)*** a utilizar, se definen todas sus propiedades y se crea internamente una tabla con el nombre del campo que se actualizara en la fase 2. Estos procesos se repite para cada uno de los campos con los que se va a trabajar ***(Edad y Sexo)***.
2. Comienza una vez finalizada la fase 1. En esta instancia se genera automáticamente el archivo de entrada al algoritmo (descripto en la sección 5.2) que obtiene las reglas de asociación. Para esto, se recorre la tabla Personas y con las propiedades obtenidas en 1 para cada campo, se actualizará la tabla correspondiente con la codificación e los valores existentes en Personas para dicho campo.

La fase 1 para cada atributo:

- **Nivel de Instrucción**

  Elección. Se realiza seleccionando en la Lista de Campos Disponibles el campo en cuestión y presionado el bottom **>>**



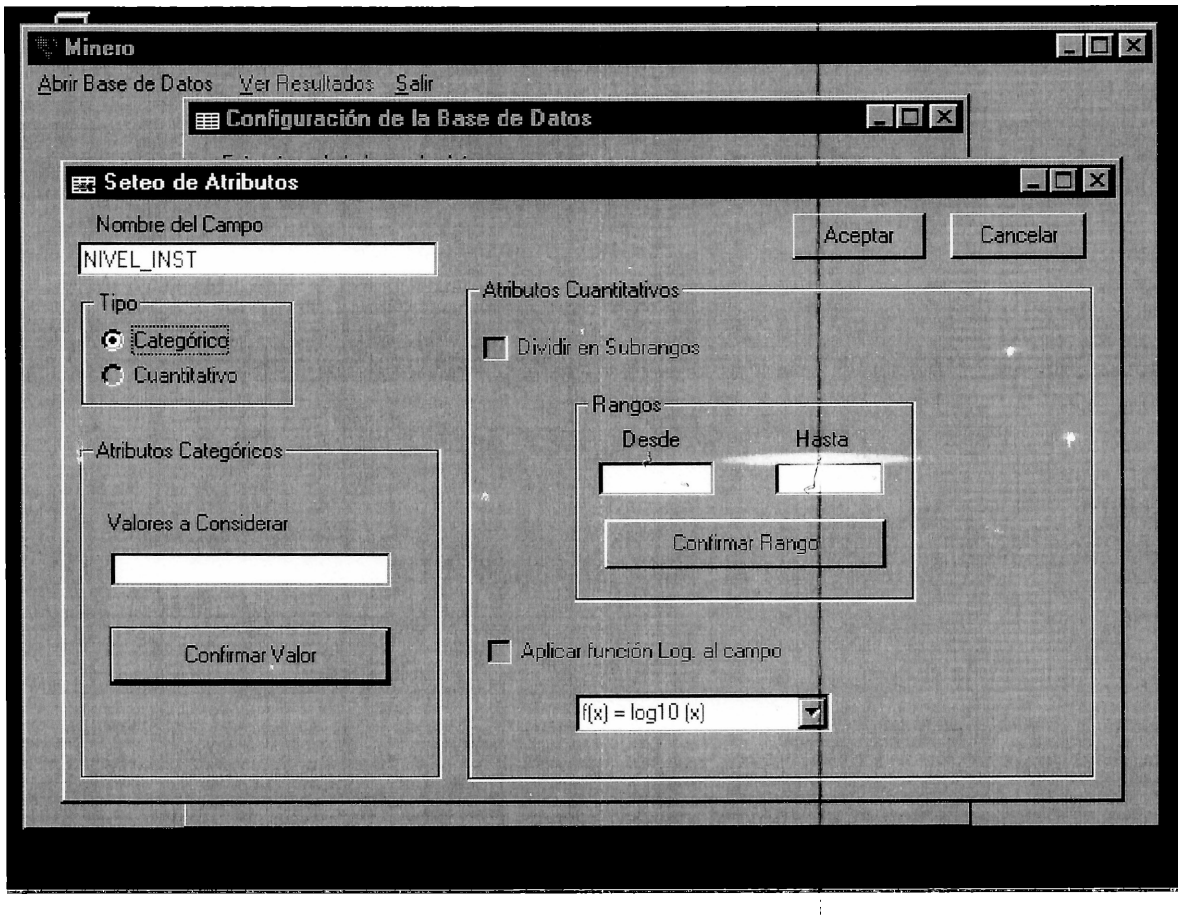
Seteo de Atributos. La selección activa la ventana "Seteo de Atributos", en la cual primero se debe identificar el tipo del campo: ***categórico o cuantitativo.*** Nivel de Instrucción es categórico.

Los atributos categóricos se pueden setear de dos formas:

a.-si se quiere estudiar solo un subconjunto del total de valores existentes en la tabla Personas. Estos se deben ingresar en forma manual en la caja de texto **"Valores a Considerar"**. Una vez finalizada la carga de valores se debe clickear **Aceptar**

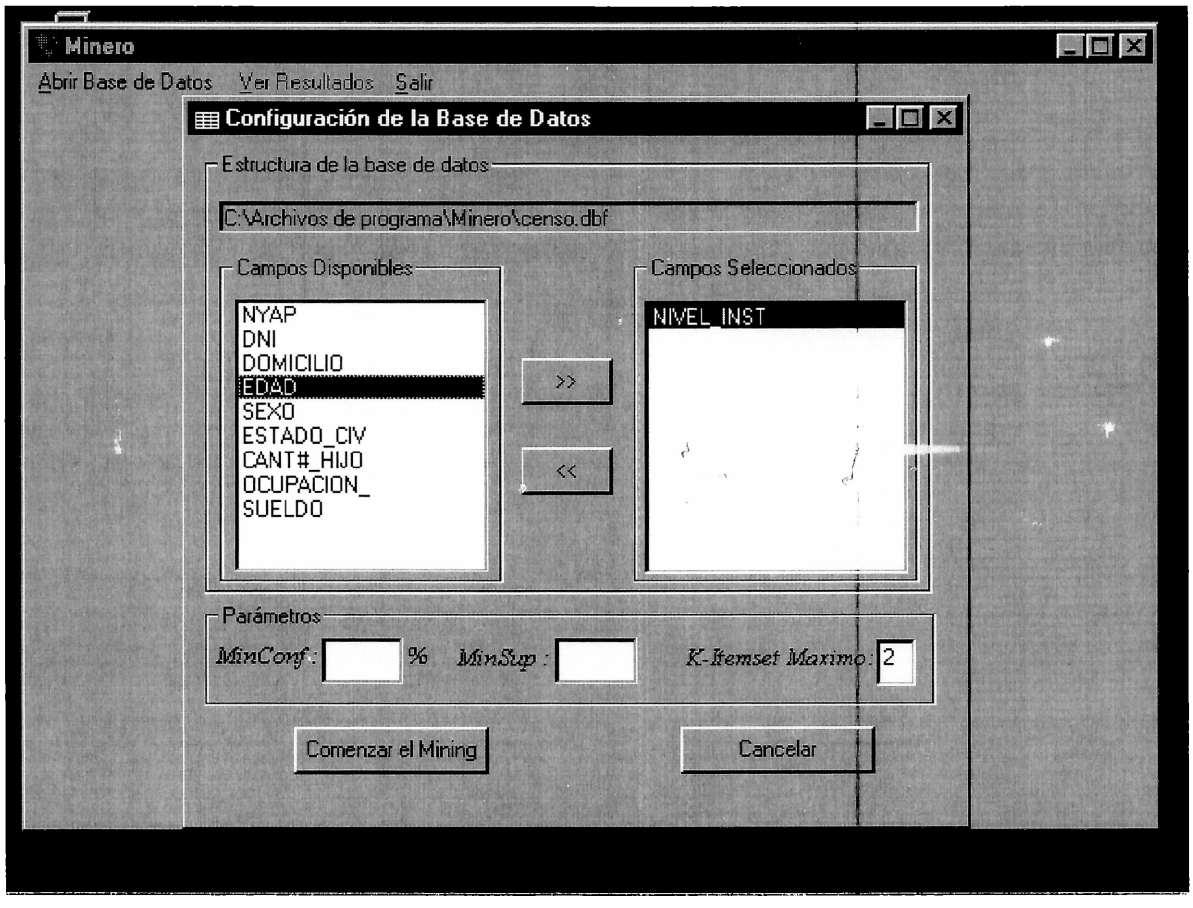
b.-si se quiere estudiar todos los valores existentes en la tabla Personas se debe hacer un clic directamente en el bottom **Aceptar**

Para el campo en proceso adoptamos la opción b, debido a que se quiere analizar todos los valores existentes en la tabla.



- **Edad**

  Selección. Se realiza seleccionando en la Lista de Campos Disponibles el campo en cuestión y presionado el bottom >>



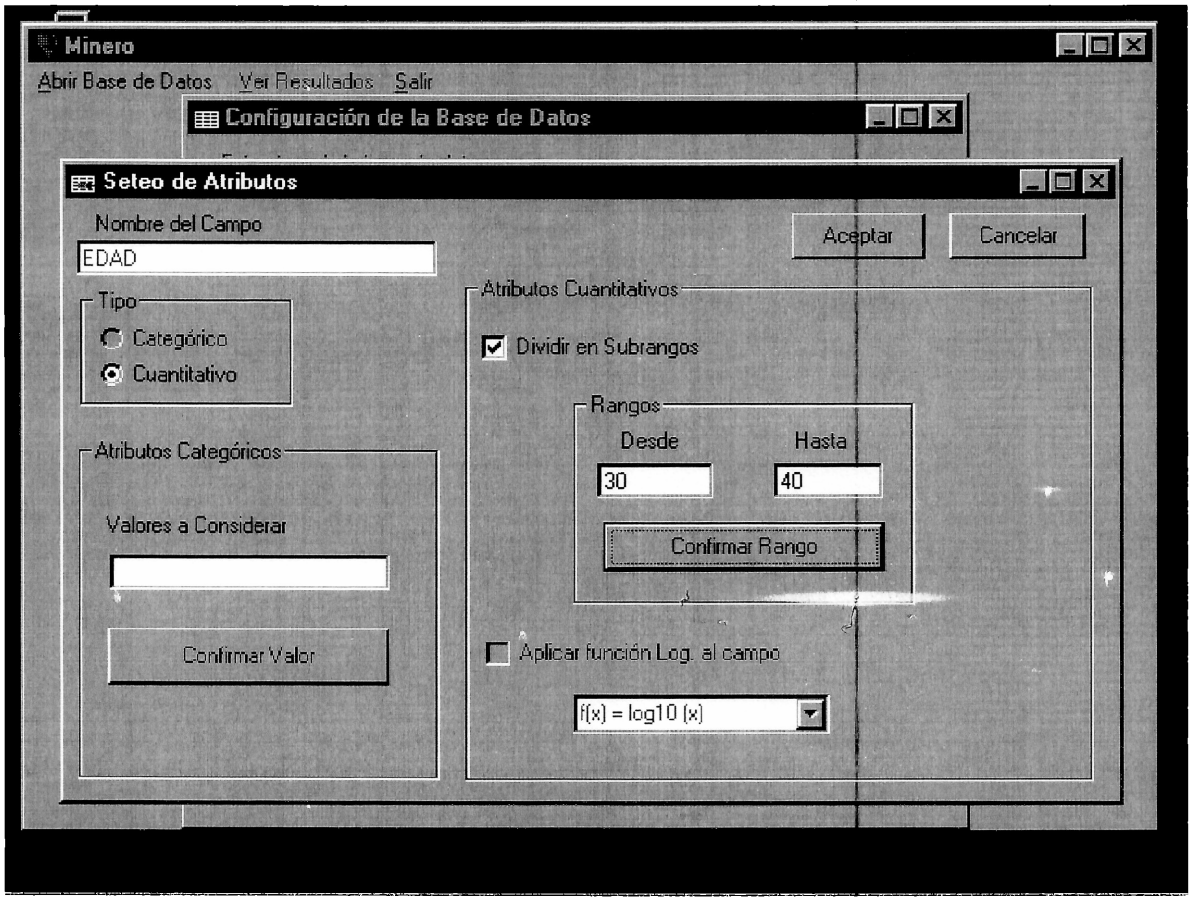
Seteo de Atributos. Edad es un campo cuantitativo. Este tipo de campo se puede trabajar de 3 formas:

a.-al igual que en los campos categóricos, se puede tomar todos los valores para el campo existente en la tabla Personas, clickeando directamente **Aceptar.** Se recomienda se uso solo en casos donde los valores son pocos.

b.-se puede crear rangos manualmente. Para luego mapear los valores del campo en los rangos.

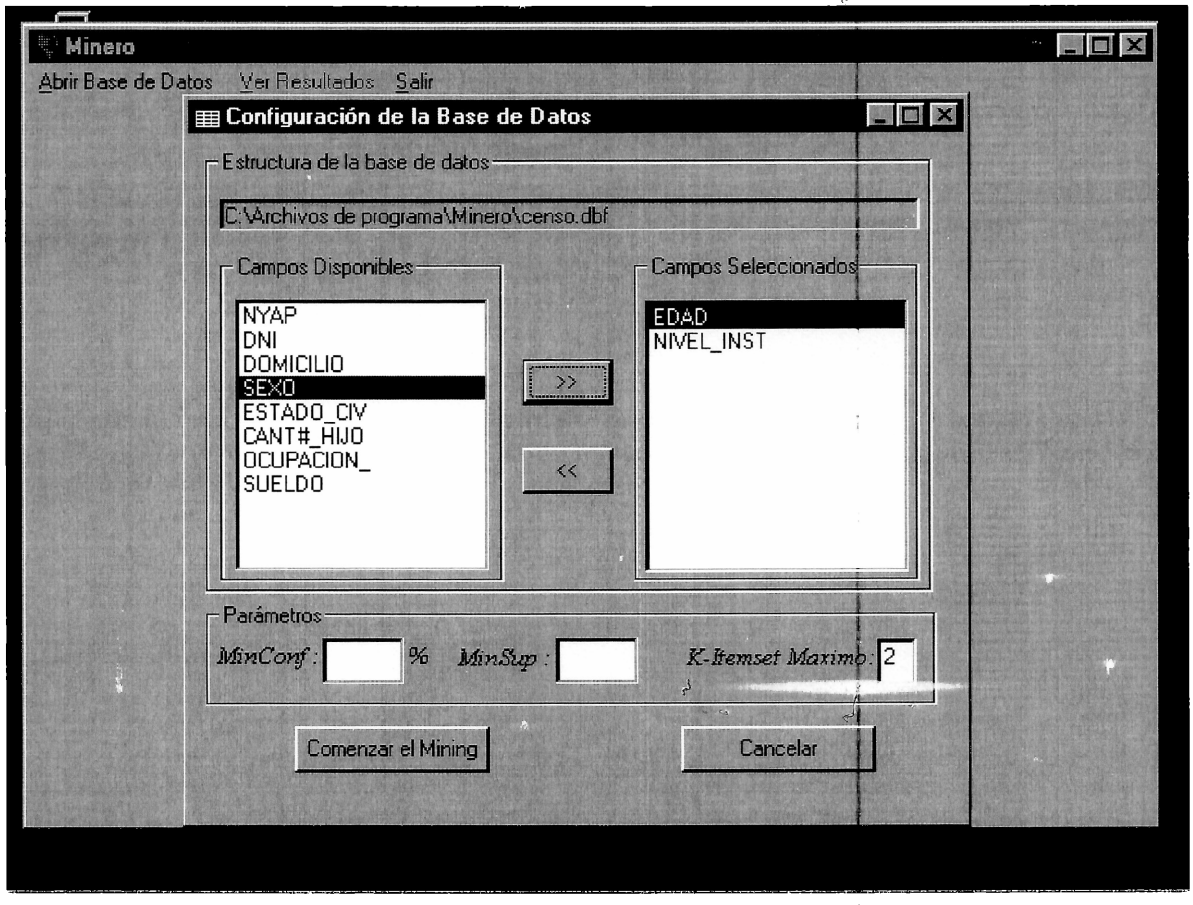
c.-se puede aplicar una función de las disponibles para acotar los valores existentes en la tabla.

A partir de nuestros objetivos, la población en que estamos interesados son los mayores de 30 años. Setearemos este campo de la forma b, y para obtener una información mas detallada crearemos los siguientes rangos: 30 a 40, 41 a 50, 51 en adelante.



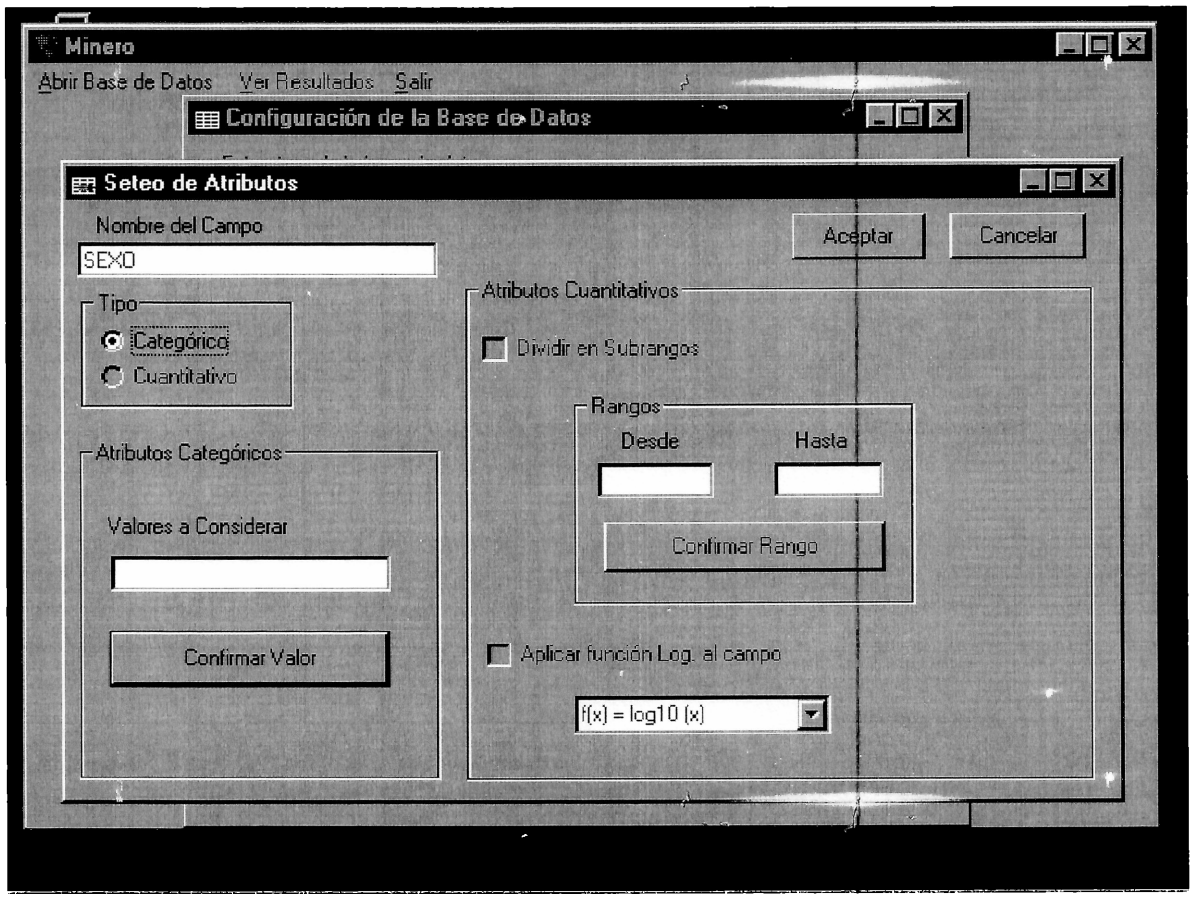
Sexo

Selección. Se realiza seleccionando en la Lista de Campos Disponibles el campo en cuestión y presionado el bottom >>



Seteo de Atributos. Sexo es un campo categórico y además se lo puede considerar booleano ya que tiene dos valores: Femenino y Masculino.

Una vez mas por los objetivos planteados, se debe considerar todos sus valores. Por esto el procedimiento es igual al utilizado para el campo Nivel de Instrucción.



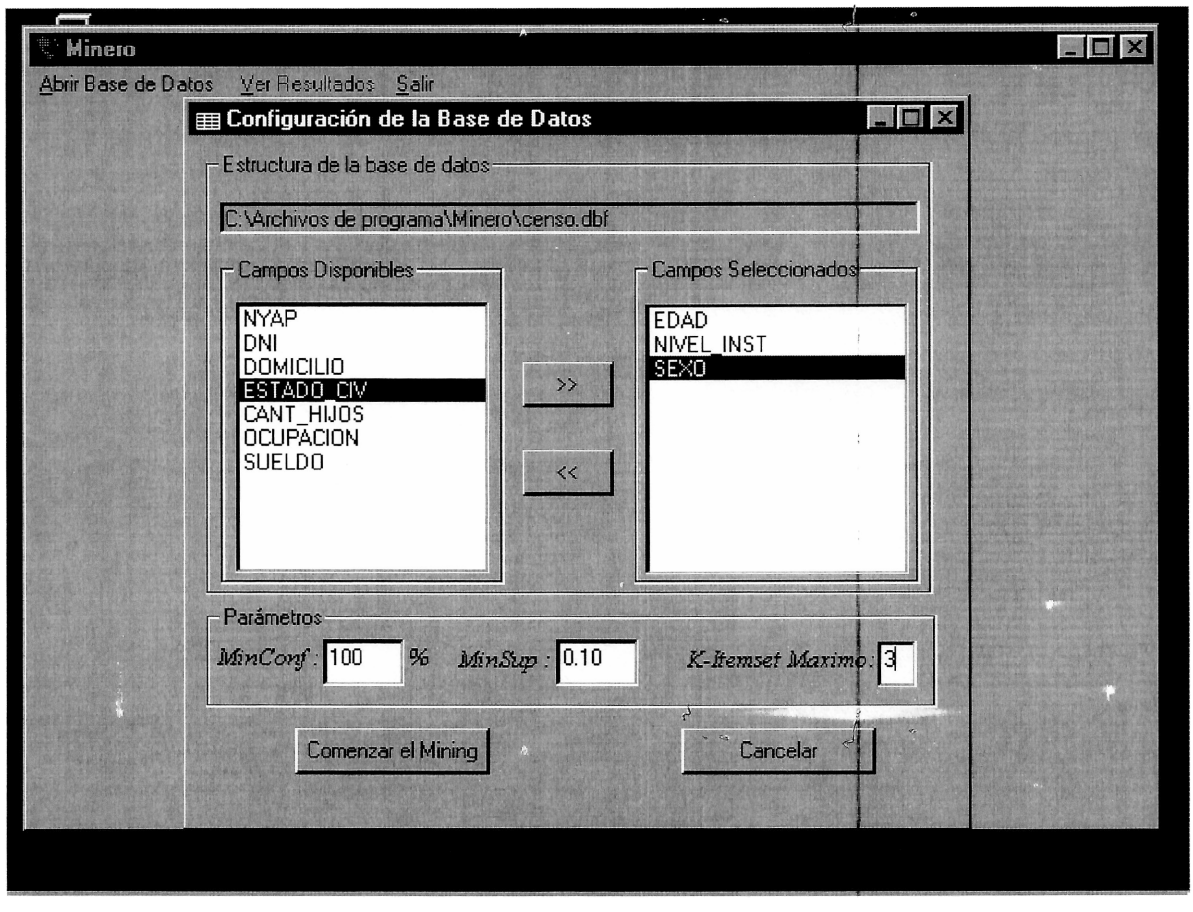
En este punto se finaliza con la fase 1 de la preparación de los datos. Debemos setear los parámetros ***Minsup***, ***MinConf*** y ***K***.

Minsup: 0.10

MinConf: 100 %

K=3.Por los objetivos especificados queremos a lo sumo tres items en cada regla. En este caso coincide con la cantidad de campos seleccionados. Si quisiéramos reglas con dos items, entonces K debería ser setado en dos.

Para comenzar con el mining se debe clickear el bottom **Comenzar el Mining.**



Ahora comienza la fase 2 de preparación de los datos. Las tablas creadas en la fase 1 son: Nivel de Instrucción, Edad y Sexo.

Se recorre la tabla Personas para actualizar las tablas anteriores y crear el archivo de entrada al algoritmo.

Tabla del Campo Nivel de Instrucción

Códigos	Valores
NI01	Ninguno
NI02	Primario Incompleto
NI03	Primario
NI04	Secundario Incompleto
NI05	Secundario
NI06	Terciario Incompleto
NI07	Terciario
NI08	Universitario

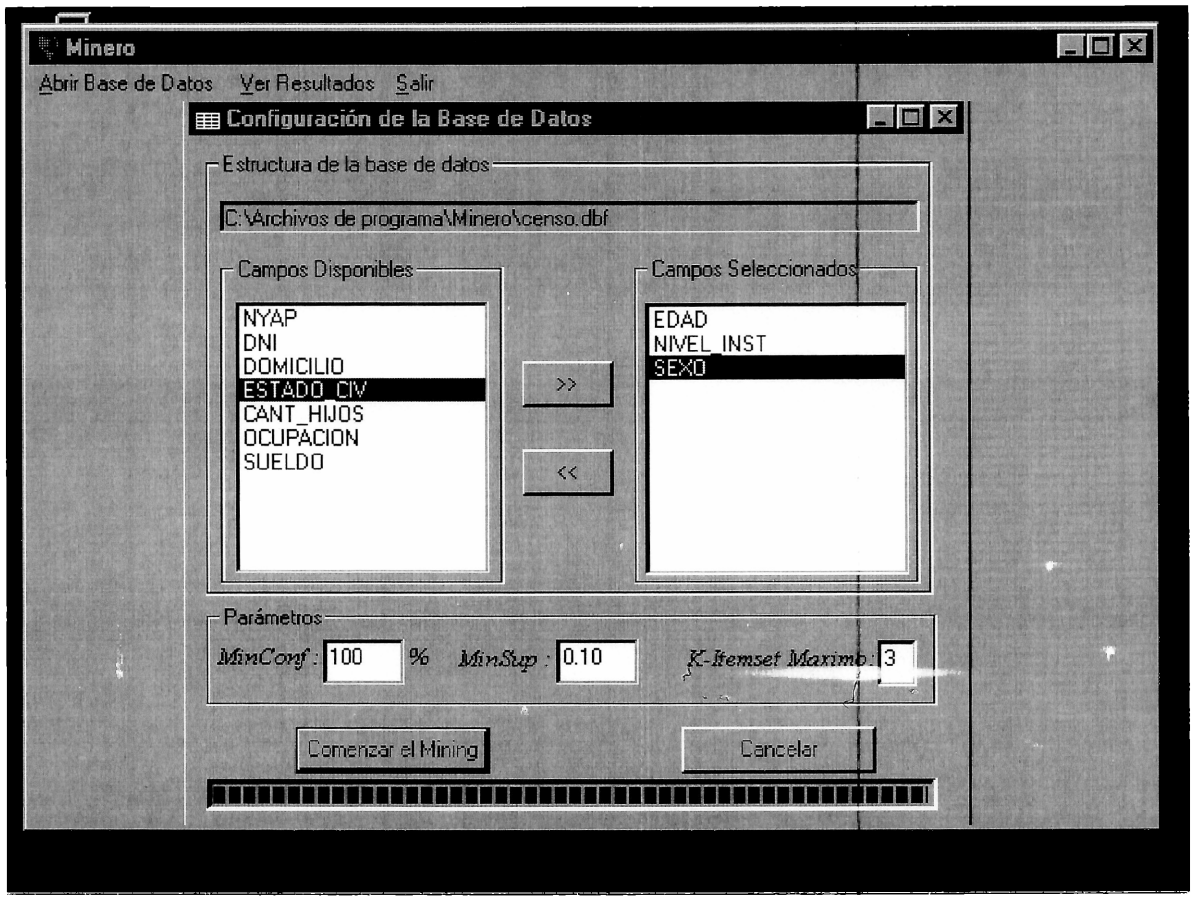
Tabla del campo Edad

Código	Limite inferior	Limite Superior
ED01	30	40
ED02	41	50
ED03	51	100

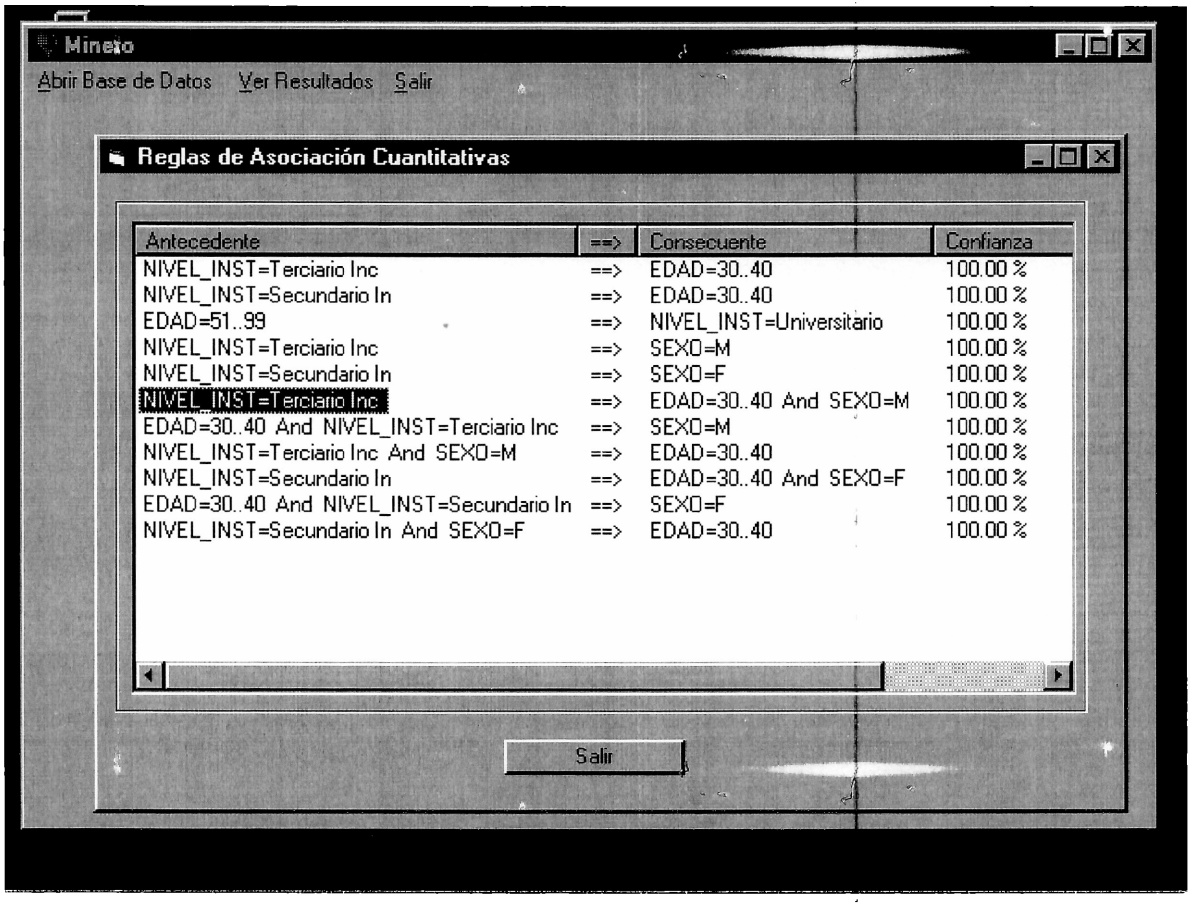
Tabla del campo Sexo

Código	Valores
SE01	F
SE02	M

Una vez creado el archivo de entrada automáticamente se dispara el algoritmo para obtener las reglas. Mientras se ejecuta este proceso se observa una barra de estado que indica lo que resta para su finalización.



Al terminar se habilita una opción del menú principal ***"Ver Resultados"***. Esta opción dispara un proceso que decodifica las reglas con las tablas obtenidas en la etapa de preparación de datos y se muestran de la siguiente manera.



Conclusiones

Como se observa en la pantalla anterior el algoritmo encontró todas las reglas de asociación cuantitativas que tienen confianza 100% y el soporte de sus itemsets mayor al 0.10 %.

Si analizamos la regla seleccionada en la pantalla, la misma nos dice que el 100% de los registros de la tabla Personas donde el campo Nivel_Inst contenga el valor "Terciario Inc", van a tener en su campo Edad un valor entre "30..40" y en su campo Sexo el valor "M" (Masculino).

Para demostrar que la regla anterior se cumple veremos a continuación los dos registros en donde aparece el valor "Terciario Inc" en el campo Nivel_Inst:

REG.	NYAP	DNI	DIR	EDAD	SEXO	ESTADO CIVIL	CANT HIJOS	OCUPACION	NIVEL_INST	SUELDO
1	Carola Marin	23511428	2 n 1470	26	F	Soltero	0	Empleado	Secundario	500
2	Marcelo Mathieu	16258633	3 n 1470	34	M	Soltero	0	Empleado	Terciario Inc	800
3	Guadalupe Marin	24125463	4 n 1470	22	F	Soltero	0	Estudiante	Secundario	400
4	Nilda Massoni	9125442	5 n 1470	52	F	Casado	3	Profesional	Universitario	750
5	Hugo Marin	9255345	6 n 1470	53	M	Casado	3	Profesional	Universitario	750
6	Dionisio Marin	24125002	7 n 1470	24	M	Soltero	0	Profesional	Universitario	600
7	Florencia Vera	23566111	8 n 1470	26	F	Soltero	0	Profesional	Universitario	800
8	Lorena Fontanari	22349080	9 n 1470	27	F	Soltero	0	Estudiante	Secundario	408
9	Dario Arriaga	21544375	10 n 147	28	M	Soltero	0	Contratista	Secundario	650
10	Eduardo Rodrigu	12345666	11 n 147	38	M	Casado	1	Empleado	Terciario Inc	410
...	...	...	...	...	...	...	...	...	...	

Se puede ver que el registro 2 tiene Edad = 34 y Sexo = M, y el registro 10 tiene Edad = 38 y Sexo = M.

Para corroborar la confianza de la regla realicemos los cálculos necesarios: el Soporte (Nivel_Inst="Terciario Inc") = 0.11, éste valor sale de dividir la cantidad de registros en donde aparece "Terciario Inc" (2), por la cantidad total de registros en la tabla (18). Se puede observar que el soporte es mayor al MinSup especificado anteriormente (0.10).

El Soporte(Nivel_Inst="Terciario Inc" Edad="30..40" Sexo = "M") también da 0.11.

La confianza de la regla se calcula como sigue:

(Soporte(Nivel_Inst="Terciario Inc" Edad="30..40" Sexo = "M") /

Soporte (Nivel_Inst="Terciario Inc")) * 100

O sea: (0.11 / 0.11) * 100 = 100%

Con ésto llegamos a la conclusión de que el 100 % de las personas de la tabla que tienen nivel de instrucción el Terciario Incompleto son personas con edades entre 30 y 40 años y de sexo masculino.

De la misma forma se analizan las demás reglas encontradas.

BIBLIOGRAFIA

1) Adriaans - Zantige.
Data Mining
Addison - Wesley. 1996.

2) Berry - Linoff.
Data Mining Techniques
Wiley. 1997.

3) Westphal - Blaxton.
Data Mining Solutions.
Wiley. 1998.

4) Cabena – Hadjinian – Stadler – Verhees – Zanasi –
Discovering Data Mining From Concept to Implementation.
IBM 1998.

5) Simoudis – Han – Fayyad
Second International Conference on Knowledge Dicovery & Data Mining
1996.

6) H. Mannila – H. Toivonen – A. Verkamo
Efficient Algorithms For Discovering Association Rules
Appeared in AAAI Workshop on Knowledge Discovery in DataBases
Seattle, Washington, July 1994.

7) R. Agrawal – T.Imielinsky – A. Swami
Mining Association Rules between Sets of Items in Large DataBases
In Proc. Of the 1993 Innternational Conference on Management of Data
May,1993

8) R.Agrawal – R. Srikant
Mining Generalized Association Rules
In Proc. Of the 21st. VLDB Conference
Zurich, Swizerland, 1995.

9) R.Agrawal – R. Srikant – Vu Quoc
Mining Association Rules with Items Constraints

10)R. Agrawal - R. Srikant
Fast Algorithms For Mining Association Rules
In Proc. of the 20th Int'l Conference on Very Large Databases.
Santiago, Chile, Septiembre 1994.

11)R. Agrawal and R. Srikant
Mining Quantitative Association Rules in Large Relational Tables

12)G. Piatetsky – Shapiro
Knowledge Discovery in DataBases
AAAI/MIT Press, Menlo Park, CA 1991.

13) Fayyad, U.- Piatetsky-Shapiro, G.- Smyth, P.- Uthurusamy, R. (Eds.):
Advances in Knowledge Discovery and Data Mining
The MIT Press. 1996.

14) Weiss, S. Indurkhya, N.
Predective Data Mining
Morgan Kaufmann. 1998