PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO – CHILE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

SISTEMA PARA EL APOYO A LA TOMA DE DECISIONES EN EL ÁMBITO HOTELERO, BASADO EN TÉCNICAS DE DATA MINING.

Juan Carlos Reyes Fernández

Kendru Alejandro Estrada Rodríguez

INFORME FINAL DEL PROYECTO PARA OPTAR AL TÍTULO PROFESIONAL DE INGENIERO DE EJECUCIÓN EN INFORMÁTICA.

Julio de 2005

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO – CHILE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

ACTA DE APROBACIÓN

Juan Carlos Reyes Fernández
Kendru Alejandro Estrada Rodríguez

Silvana Roncagliolo de la Horra
Profesor Guía

Julio de 2005

RESUMEN

Este proyecto, consiste en la realización de un sistema que permite encontrar patrones interesantes en los datos almacenados a través del tiempo por un sistema de información en una empresa específica. Estos patrones son clasificados en tres tipos: reglas de asociación, grupos homogéneos (Clústering) y modelos predictivos. Dichos patrones son encontrados gracias a la utilización de técnicas de Data Mining, y son útiles al momento de tomar decisiones en la empresa, puesto que revelan información no conocida hasta el momento sobre el comportamiento de sus clientes, dado éste por sus gustos y preferencias hacia los servicios ofrecidos en dicha organización. La solución presentada se integra con el sistema de información operacional de la empresa a través de la conexión a su base de datos, con el fin de extraer la información acumulada, prepararla para su utilización y realizar el análisis correspondiente.

La empresa para la cual se realizó el proyecto es el Hotel San Martín de Viña del Mar. Para tal organización, corresponder al ámbito turístico le implica requerir un amplio conocimiento sobre las preferencias y gustos de los clientes para poder así ofrecer un servicio de calidad acorde a estos gustos y preferencias.

ABSTRACT

This Thesis Project consists of the execution of a system that is able to find interesting patterns among the data stored within a certain period of time by an information system in a private establishment. These patterns have been classified into three types: association rules, clustering and predictive models. Such patterns can be found by using Data Mining techniques. These patterns are useful when making decisions in such establishment because they reveal information about the clients' preferences for the services offered. The solution proposed above was applied to the operational system of the establishment through the data base connection in order to collect the stored information, prepare it to be used and perform the corresponding analysis.

This project was held at Hotel San Martín in Viña del Mar. The fact that this establishment is part of a touristic world means that the information about its clients' preferences is crucial to be able to offer quality services or improve the ones available.

Por el apoyo brindado durante toda mi vida dedico este trabajo a mis padres:

Kendru Estrada Duran – María Rodríguez Ahumada

Kendru Estrada Rodríguez

Por creer siempre en mis capacidades y darme el apoyo necesario durante mis años académicos dedico este trabajo a mis padres:

Ernesto Reyes Torres - Inés Fernández Donoso

Juan Carlos Reyes Fernández

Agradecimientos

Agradecemos en primer lugar, el constante y valioso apoyo entregado por nuestra profesora guía Silvana Roncagliolo de la Horra, también agradecemos a todas las personas que aportaron con sus ideas y conocimientos técnicos sobre temas afines al desarrollo del proyecto.

Kendru Estrada R.
Juan Carlos Reyes F.

INDICE

I.- INTRODUCCION

Actualmente, la industria turística está atravesando un periodo de grandes cambios que han sido propiciados por causas de diferente índole, como por ejemplo, los gustos de los consumidores han variado, lo que se manifiesta en una tendencia creciente por parte de los clientes a demandar viajes personalizados adaptados a sus preferencias; el mayor conocimiento que poseen los clientes dada la mayor facilidad de acceso a fuentes de información disponibles y a un nivel más alto de experiencia en viajes; y el incremento de la competencia por la existencia de otros destinos turísticos, lo que la ha llevado a enfrentarse a un mayor crecimiento de la demanda.

Como respuesta a este nuevo entorno, el disponer de información valiosa que les ayude a fijar estrategias y tomar decisiones que favorezcan el desarrollo del negocio, resulta fundamental para las empresas del sector, como es el caso de los hoteles.

El presente proyecto, considerando lo importante que significa estar al tanto de las preferencias de los consumidores debido a los cambios nombrados anteriormente, tiene como propósito realizar un análisis del comportamiento de los clientes de un hotel específico de la V Región. Para que de esta manera la tecnología de información, le permita estar en condiciones de aplicar a tiempo aquellas innovaciones que le puedan ayudar a incrementar su nivel de competitividad.

Dotarse de instrumentos que impliquen solamente gestionar con eficacia las actividades y procesos de negocio ya no es suficiente, sino que son necesarias herramientas que permitan generar conocimiento que sirva de apoyo a los directivos y usuarios del sistema para tomar decisiones y anticiparse a los cambios. Para lograr obtener este conocimiento relativo al comportamiento de los clientes se realiza la construcción de un sistema computacional el cual utiliza técnicas de Data Mining (Minería de Datos) aplicadas sobre bases de datos históricas generadas por un sistema de información automatizado ya implantado en el hotel.

En lo que sigue del informe, se detallan los objetivos generales y específicos del proyecto (Capítulo 2), posteriormente se describen aspectos importantes sobre el problema en sí, y qué aplicaciones ya están disponibles en el mercado para solucionar tales problemas (Capítulo 3), se presenta una investigación sobre aspectos relacionados con Data Mining (Capítulo 4), se da a conocer la programación de tareas llevada a cabo para el desarrollo del proyecto (Capítulo 5), posteriormente se describe el desarrollo del proyecto incluyendo metodologías y herramientas utilizadas, como así también el detalle del diseño del sistema y su construcción (Capítulo 6). Para finalizar se presentan las conclusiones obtenidas (Capítulo 7), también se presenta un glosario de términos y las referencias utilizadas sobre los textos y sitios Web que sirvieron de apoyo para el proyecto.

1.- Definición del problema

En la actualidad, la información es uno de los recursos más importantes en la toma decisiones, es por esta razón que generalmente se registran grandes cantidades de información relevante para esta tarea. Las empresas del sector turístico, dentro de las cuales se encuentra el Hotel San Martín no son una excepción; en estas entidades se almacena diariamente numerosa información sobre los clientes y su relación con la empresa, sin embargo, al momento de tomar decisiones toda esta información puede ser difícil de analizar aún con herramientas de resumen, y es posible que existan patrones en la información, que las personas no sean capaces de identificar debido a la gran cantidad de información existente. Por tal motivo, es necesario resolver éste tipo de problemas desarrollando herramientas que permitan analizar la información acumulada y así poder encontrar éstos patrones dentro de la información (Figura 1). Al desarrollar éste tipo de herramientas se busca ayudar al usuario a tomar mejores decisiones y utilizar la información de una manera inteligente. Los patrones aquí mencionados, pueden ser importantes al momento de tomar decisiones estratégicas, puesto que estarían relacionados con el comportamiento del cliente dentro de la organización. Este comportamiento estaría íntimamente ligado con sus gustos y preferencias, información que es de suma importancia y de la cual no se tiene un conocimiento certero al momento de planificar estrategias de marketing.

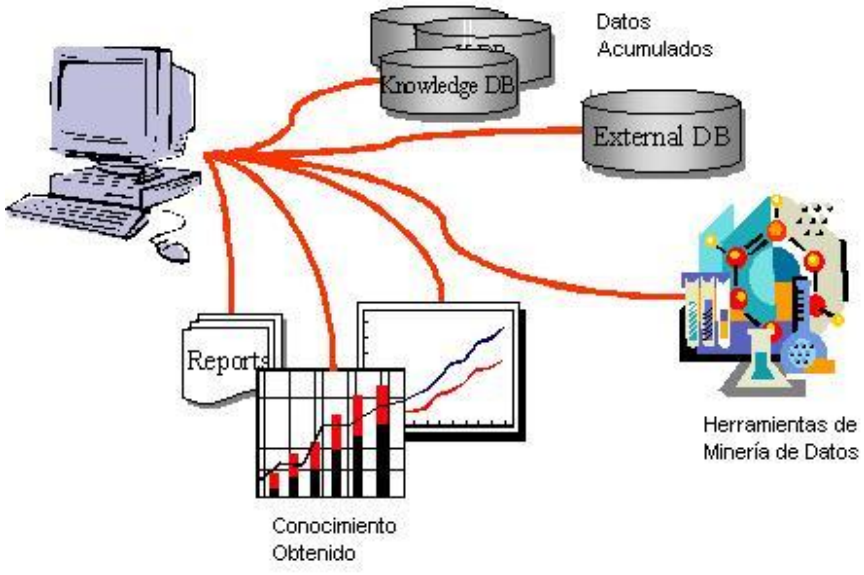


Figura 1. Esquema de la solución

2.- Solución propuesta

Al analizar el problema descrito en el punto anterior, se propone mediante el presente proyecto, realizar un sistema informático que explore la información acumulada en las bases de datos históricas, analice dicha información y entregue conocimiento útil para la toma de decisiones (ver esquema Figura 1).

Para obtener conocimiento útil desde las bases de datos históricas, se utilizarán técnicas de Data Mining, las cuales permitirán realizar las siguientes tareas necesarias para lograr los objetivos deseados:

- ***Encontrar reglas de asociación en el comportamiento de los clientes***
  Reglas que permitan encontrar asociaciones de implicancia importantes entre los servicios ocupados por los clientes, para saber así, qué servicios determinan la utilización de otros servicios. En el análisis de requerimientos se da una explicación mas detallada sobre el tema.

- ***Agrupar clientes***
  El agrupamiento de clientes permitirá segmentar el mercado objetivo de tal manera de formar grupos de clientes con gustos definidos. Este análisis ayudará a dirigir campañas publicitarias a determinados grupos de clientes.

- ***Predecir el comportamiento de los clientes en alguna situación dada***
  Dado el comportamiento en situaciones anteriores de los clientes, se puede generar un modelo y a partir de éste predecir el comportamiento de los futuros clientes.

II.- OBJETIVOS

Al desarrollar un sistema, sea cual sea su naturaleza es importante establecer las metas que se deben cumplir para llevar a buen término el proyecto. A continuación se presentan los objetivos generales y específicos a cumplir.

1.- Objetivos generales

Los objetivos generales del proyecto son:

- Desarrollar un sistema que permita solucionar la problemática del análisis de grandes volúmenes de datos acumulados en las bases de datos históricas del Hotel San Martín, para obtener así modelos y patrones predictivos y descriptivos del comportamiento del cliente y su relación con la empresa, con el fin de servir de apoyo a la toma de decisiones estratégicas.
- Acercar la tecnología de la minería de datos al usuario final de una manera amigable.

2.- Objetivos específicos

Los objetivos específicos del proyecto son:

- Recuperar información histórica almacenada en las bases de datos del Hotel y procesarla mediante técnicas de Data Mining para obtener información útil sobre el comportamiento del cliente.
- Almacenar la información útil en una Base de Datos para ser utilizada como apoyo a la toma de decisiones.
- Presentar la información útil en forma amigable para el usuario, esto es en lenguaje comprensible y no expresado en términos muy técnicos.
- Generar informes gráficos con la información requerida.

III.- ESTADO DEL ARTE

El Data Mining o minería de datos es una parte de la ciencia informática que ha penetrado fuertemente en el mundo las empresas, debido a que es en éstas donde generalmente se posee una inmensa cantidad de datos, que han sido almacenados a lo largo de los años por sistemas de información convencionales, y es necesario realizar un análisis de estos datos para obtener información que sea útil para las empresas.

El concepto de minería de datos junto con conceptos como Datawarehouse y CRM (Customer Relationship Management), están siendo utilizados cada vez más en las empresas para realizar una gestión en donde lo más importante es el cliente. Esto se realiza registrando una gran cantidad de datos acerca del cliente, almacenándolos en bases de datos unificadas no volátiles, y analizando estos datos para obtener conocimiento útil relacionado con el cliente. Es en esta última parte donde se utiliza el Data Mining, para encontrar patrones en la información, que sirvan a los directivos para tomar decisiones sobre las relaciones que se deben establecer con los clientes.

En el ámbito de investigación y científico, se utiliza el Data Mining para establecer relaciones entre variados resultados de experimentos, para realizar deducciones desde un conjunto inmenso de premisas y en general para cualquier tarea en la que se necesite analizar grandes cantidades de datos.

En la actualidad existen diversas soluciones, para el problema que se va a afrontar, la mayoría de estas soluciones, son sistemas de Data Mining genéricos capaces de analizar una base de datos de cualquier tipo, los resultados que entregan estos sistemas, son mas bien técnicos y deben ser interpretados por especialistas que conozcan el tema, y no serán útiles para algún directivo de empresa no preparado. Algunos de estos sistemas son:

Intelligent Miner: es un software de minería de datos de IBM, que provee un amplio rango de algoritmos de Data Mining incluyendo asociación, clasificación, regresión, modelado predictivo, análisis de patrones secuenciales y clustering (agrupamiento), también contiene herramientas para el trabajo con algoritmos de redes neuronales, métodos estadísticos, herramientas de preparación de datos, herramientas de visualización y provee integración con el software de bases de datos DB2 de IBM [HK01].

Enterprise Miner: fue desarrollado por SAS Institute Inc., este sistema provee múltiples algoritmos de minería de datos incluyendo clasificación, regresión y análisis estadístico. Una característica distintiva de Enterprise Miner es la variedad de herramientas de análisis estadístico que posee, las cuales han sido construidas basadas en la larga historia de SAS en el mercado del análisis estadístico [HK01].

MineSet: fue desarrollado por Silicon Graphics Inc. (SGI), también provee múltiples algoritmos de minería de datos, una característica distintiva de MineSet es el conjunto de robustas herramientas gráficas (usando las poderosas gráficas de los computadores de SGI), incluyendo visualizador de reglas, visualización de árboles, visualización multidimensional para datos y resultados del Data Mining [HK01].

Clementine: fue desarrollado por Integral Solutions Ltda. (ISL). Este sistema provee un entorno de desarrollo integrado de Data Mining, para usuarios finales y desarrolladores. Múltiples algoritmos de Data Mining. Una característica distintiva de Clementine es que es orientado a objetos, lo cual permite a los usuarios añadir sus algoritmos y utilidades, al entorno de programación visual de Clementine. Clementine fue adquirido por SPSS Inc. [HK01].

Weka: Weka es una colección de algoritmos de aprendizaje de máquina para tareas de Data Mining. Los algoritmos pueden ser ejecutados directamente para un conjunto de datos o pueden llamarse de su código en Java. Weka contiene herramientas para pre-procesado, reglas de clasificación, regresión, agrupamiento, asociación, y visualización. Es también una buena suite para el desarrollo de nuevos esquemas de aprendizaje. Weka es Open-Source y es distribuido bajo GNU General Public License [Web-1].

Las alternativas nombradas anteriormente, entregan resultados poco amigables (o demasiado técnicos) para el usuario, el cual debería poseer conocimientos avanzados sobre áreas que no serían de su dominio de experiencia tales como la estadística. Además la mayoría de estos sistemas poseen un elevado costo, lo cuál significaría una cierta desventaja; este no es el caso de Weka puesto que es Open Source, sin embargo esta herramienta consiste, como se dijo anteriormente, en una colección de algoritmos para tareas de Data Minig, por lo cual resolverían solo una parte del proceso de descubrimiento de conocimiento, dejando de lado tareas tan relevantes como la transformación de los datos objetivo.

En cuanto a los sistemas hoteleros existentes en el mercado, no se conocen versiones que posean herramientas de análisis de datos como las que ofrece el sistema propuesto. Este es el caso del sistema utilizado por el Hotel San Martín, el cual está abocado exclusivamente al manejo operacional del rubro hotelero, sin incluir análisis avanzado de datos. Por esto se hace necesario el desarrollo de un sistema a la medida para resolver esta carencia.

IV.- INVESTIGACION

1.- Data Mining

Según Han y Kamber [HK01] la minería de datos se refiere a "la extracción (o minería) de conocimiento desde grandes cantidades de datos". Esto está relacionado con el hecho de que existe mucha información valiosa que es desconocida, debido a que es imposible analizar una gran cantidad de información sin ayuda de herramientas automatizadas.

La tecnología Data Mining en el último tiempo se está haciendo cada vez más popular, pues existen un sinnúmero de sistemas que producen una cantidad inmensa de datos; estos datos con frecuencia contienen valiosa información que puede resultar muy útil para diversas tareas estratégicas de las empresas. Esta información al estar oculta implica que sobre los datos acumulados sea aplicado Data Mining para dejar al descubierto este importante conocimiento.

Data Mining corresponde a uno de los pasos de un proceso mayor, denominado KDD *(Knowledge Discovery in Databases)*. En la Figura 2 se presentan los pasos del modelo KDD.

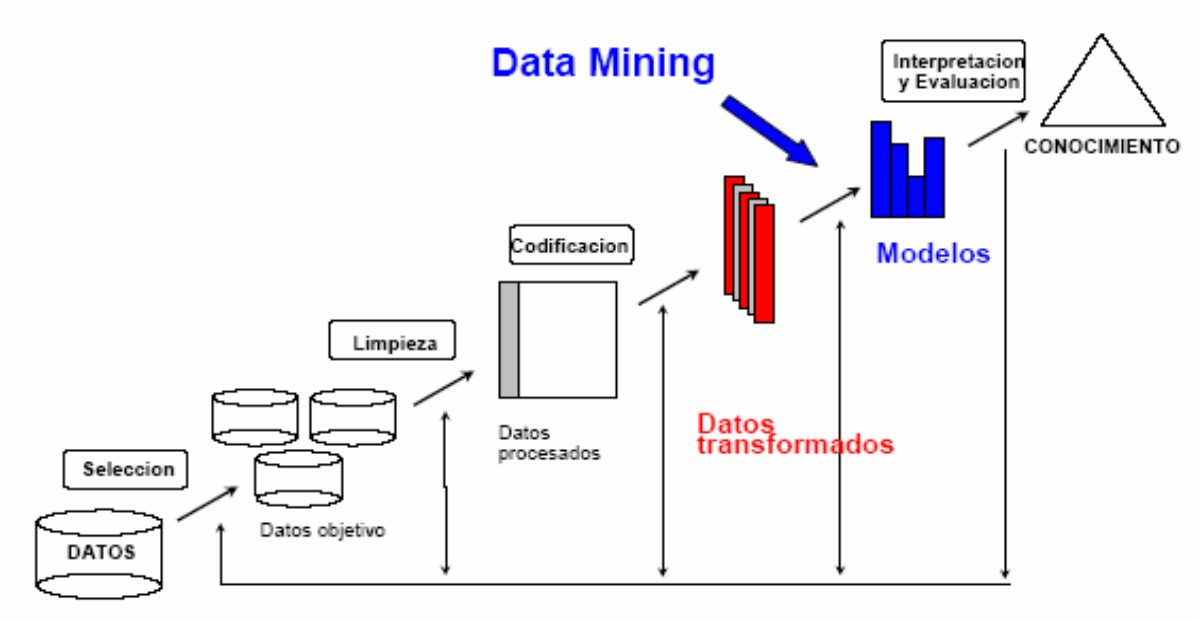


Figura 2. Proceso de Descubrimiento de Conocimiento

El proceso KDD es un proceso iterativo, el cuál consta de pasos básicos [HK01] e involucra decisiones por parte del usuario, siendo interactivo. Se han identificado los siguientes pasos como componentes del proceso: Selección de un conjunto de datos objetivo, Preprocesamiento y limpieza de los datos, Transformación y Reducción en la dimensión de los datos, Selección del método de minería de datos y de la técnica (algoritmo) de minería de datos e Implementación de la técnica para realizar la extracción de patrones, Interpretación o Evaluación de los patrones extraídos, y Consolidación del conocimiento descubierto [HK01].

2.- Técnicas y Enfoques para Data Mining

Las técnicas son implementaciones específicas de los algoritmos que se utilizan para llevar a cabo las operaciones de Data Mining; por otra parte, los enfoques están dados por la metodología utilizada y tienen relación con las ramas de la ciencia que resuelven el problema en si.

Cada enfoque puede contemplar una amplia gama de técnicas, se utilizará una u otra dependiendo del enfoque elegido y de la naturaleza del problema a tratar.

No todos los algoritmos para resolver un determinado problema de Data Mining son iguales y cada uno de ellos tendrá una serie de características, ventajas e inconvenientes. La conveniencia de aplicar un determinado algoritmo depende no sólo del tipo de problema con el que se desee resolver sino que depende en gran medida del tipo de los datos con los que se está tratando. En este sentido conviene analizar los distintos enfoques y algoritmos que existen para poder aplicar de una buena manera estas técnicas.

Los algoritmos utilizados en Data Mining provienen de otras áreas de investigación como la estadística o la inteligencia artificial con lo que convendrá analizar estos algoritmos para poder utilizarlos en el momento apropiado y de la manera adecuada.

2.1- Enfoques

Hasta el momento han sido varios los enfoques que han surgido y que van dirigidos hacia la investigación. Entre los enfoques más significativos están:

Enfoques estadísticos

Estos enfoques se dirigen hacia la construcción de modelos estadísticos para el análisis de datos.

A este tipo de enfoques pertenecen tanto la inferencia bayesiana, utilizada como herramienta para la extracción de conocimiento, como las redes bayesianas, empleadas para descubrir relaciones causales entre objetos.

Machine learning

Este enfoque utiliza un modelo cognitivo para imitar el proceso de aprendizaje humano.

Son numerosas las técnicas que pertenecen a este enfoque, podrían citarse, entre otras, el aprendizaje por ejemplos, el clústering conceptual y la inducción mediante árboles de decisión. Todas estas técnicas tienen en común la generalización de modelos cognitivos fáciles de entender por los humanos, sin embargo suponen que el conjunto de datos es pequeño, lo que implica una gran restricción al disponer de grandes volúmenes de datos.

Enfoques orientados a las bases de datos

En este tipo de enfoques se utilizan técnicas específicas de bases de datos para extraer reglas de las bases de datos relaciónales. Dentro de este tipo de enfoques se podrían incluir técnicas como la generalización y el uso de jerarquías conceptuales para la extracción de reglas, así como el uso del algoritmo "*A priori"*, propuesto por Agrawal [Ag93] para la extracción de reglas de asociación de grandes bases de datos transaccionales.

Enfoques matemáticos

Estos enfoques se dirigen hacia la construcción de modelos matemáticos para la extracción de reglas, regularidades (hechos ocurridos frecuentemente) y demás patrones del modelo. A este enfoque, entre otras, pertenece la teoría de los *Rough Sets*, utilizada para el descubrimiento de conocimiento en bases de datos como herramienta para encontrar dependencias en los datos y para derivar tablas de decisión de las que se pueden obtener las reglas de decisión. Al mismo tiempo, los Rough Sets se han utilizado en el cálculo de reductos de una tabla de decisión, cálculo que consiste en la eliminación de los atributos o propiedades que sean dependientes o redundantes en dicha tabla.

Enfoques integrados

Integran varios enfoques con el fin de obtener mayor eficiencia. Así, por ejemplo, el sistema QUEST, desarrollado por IBM, integra técnicas de *Data Mining* con enfoques orientados a las bases de datos, el sistema DBMINER, combina técnicas de *Data Mining* con tecnología de bases de datos.

Además de estos, existen otros enfoques dentro de los cuales pueden incluirse las técnicas de visualización y de representación del conocimiento.

3.- Operaciones de Data Mining

A continuación se describe cada una de las diferentes operaciones de Data Mining [Web-2].

Modelos predictivos

Los modelos predictivos se basan en la experiencia del aprendizaje humano, donde se utiliza la observación para construir un modelo esencial de "algo" y luego se utiliza este modelo para clasificar "algo" similar. Por ejemplo

observando varios perros, se crea un modelo genérico de perro que es un ser vivo de cuatro patas que ladra y luego las distintas razas de perro se asocian con el modelo de perro que se ha creado previamente, sin necesidad de memorizar todos y cada uno de los tipos de perro que existen.

En los algoritmos de Data Mining se utilizan los datos para formar un modelo con las características esenciales de esos datos. Los modelos se crean en dos fases, una de entrenamiento y otra de prueba. Durante la fase de entrenamiento se utiliza un conjunto de datos históricos y en la fase de prueba se valida el modelo con datos reales. Al final estos modelos se reducen a reglas del tipo IF-THEN.

La clasificación se utiliza para determinar a que clase pertenece un registro de la base de datos. La clase debe pertenecer a un conjunto finito y determinado previamente. Este tipo de aprendizaje es aprendizaje supervisado.

Segmentación de bases de datos

El objetivo de segmentar una base de datos es particionar la base de datos en conjuntos de objetos con características similares, es decir conjuntos homogéneos.

Dentro de un conjunto los elementos deben ser homogéneos, pero los elementos de diferentes segmentos deben ser diferentes. Para clasificar un elemento se utiliza una distancia que se define en función del algoritmo.

Este tipo de operación se utiliza para descubrir subpoblaciones dentro de una base de datos y poder crear perfiles que caractericen a esas subpoblaciones.

Este tipo de aprendizaje es aprendizaje no supervisado, puesto que en ningún momento se le indica a los algoritmos las clases a utilizar.

Las técnicas que se utilizan son clustering demográfico o clustering neuronal, que se diferencian en el tipo de datos que permiten como entrada, en la distancia que utilizan para calcular la pertenencia a una clase y en la forma de organizar los segmentos de salida.

Análisis de asociaciones

Mediante esta técnica se trata de buscar relaciones entre diferentes registros de la base de datos. Estas relaciones suelen llamarse asociaciones. Un ejemplo de este tipo de operación es el descubrimiento de aquellos productos o servicios que los clientes suelen comprar o utilizar juntos. Como consecuencia, los resultados a menudo se utilizan para realizar venta cruzada o para realizar el seguimiento de los objetivos de ventas de una empresa (verificar el cumplimiento de los mismos), o el cambio de precio de algún bien.

Dentro de esta operación de Data Mining se distinguen tres variantes:

- **Descubrimiento de asociaciones**, que se utiliza principalmente para observar registros que ocurren simultáneamente.
- **Descubrimiento de patrones secuenciales**, que son registros que ocurren distanciados en el tiempo.
- **Descubrimiento de patrones en series temporales**, que está orientado a la obtención de modelos que permitan predecir con certeza el próximo valor de una secuencia de valores observados a lo largo del tiempo.

V.- PLAN DE TRABAJO

Para tener un mayor control sobre las tareas a llevar a cabo durante el desarrollo del proyecto, se realizó una planificación dallada presentada en el Anexo "A" mediante una Carta Gantt.

Un Diagrama de Gantt es una forma fácil para calendarizar tareas. Es esencialmente una gráfica en donde las barras representan cada tarea o actividad. La longitud de cada barra representa la longitud de tiempo asociada a la tarea.

La ventaja principal del Diagrama de Gantt es su simplicidad. Se utilizará esta técnica puesto que, no solamente es fácil de usar, sino que también lleva por si misma a una comunicación valiosa con los usuarios finales. Otra ventaja del uso de un Diagrama de Gantt es que las barras que representan actividades o tareas son trazadas a escala, esto es, el tamaño de la barra indica la longitud relativa del tiempo que llevará a terminar la tarea.

VI.- DESARROLLO

1.- Paradigma elegido

Se espera desarrollar un software de calidad mediante una mayor transparencia y control sobre el proceso. Para lograr este objetivo, es necesario incorporar una estrategia de desarrollo que acompañe al proceso, métodos y a las herramientas utilizadas.

Como modelo de desarrollo de software, se ha optado por el proceso de desarrollo propuesto por la "Rational Software Corporation", resultado del esfuerzo de las tres últimas décadas en desarrollo de software: el Proceso Unificado de Desarrollo (RUP) [Web-6].

RUP es uno de los procesos de desarrollo más generales de los existentes actualmente, ya que en realidad está pensado para adaptarse a cualquier proyecto. Las características principales del Proceso Unificado de Desarrollo son las siguientes:

- **Guiado por casos de uso:** La razón de ser de un sistema de software es servir a usuarios ya sean humanos u otros sistemas; un caso de uso es una facilidad que el sistema de software debe proveer a los usuarios. Los casos de uso reemplazan la antigua especificación funcional tradicional y constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema.

- **Centrado en la arquitectura:** La arquitectura involucra los elementos más significativos del sistema y está compuesta entre otros por plataformas de software, sistemas operativos, motores de bases de datos, protocolos, consideraciones de desarrollo y requerimientos no funcionales. Los casos de uso guían el desarrollo de la arquitectura, y la arquitectura se realimenta en los casos de uso, los dos juntos permiten gestionar y desarrollar adecuadamente el software.

- **Iterativo e Incremental:** Para hacer más manejable un proyecto es recomendable dividirlo en ciclos. Para cada ciclo se establecen fases de referencia, cada una de las cuales debe ser considerada como un mini proyecto cuyo núcleo fundamental está constituido por una o más iteraciones de las actividades principales básicas de cualquier proceso de desarrollo.

- **Desarrollo basado en componentes:** La creación de sistemas intensivos en software requiere dividir el sistema en componentes con interfaces bien definidas, que posteriormente serán ensamblados para generar

el sistema. Esta característica en un proceso de desarrollo permite que el sistema se vaya creando a medida que se obtienen o que se desarrollan y maduran sus componentes.

- **Utilización de un único lenguaje de modelado (UML):** UML es utilizado como único lenguaje de modelado para el desarrollo de todos los modelos del sistema.

- **Proceso integrado:** Se establece una estructura que abarque los ciclos, fases, flujos de trabajo, mitigación de riesgos, control de calidad, gestión del proyecto y control de la configuración; el proceso unificado establece una estructura que integra todas estas facetas.

1.1.- Fases del ciclo de desarrollo

El Proceso Unificado de Desarrollo considera que cualquier desarrollo de un sistema de software debe pasar por cuatro fases (preparación inicial, preparación detallada, construcción y transición), la Figura 3 muestra las fases de desarrollo y los diversos flujos de trabajo involucrados dentro de cada fase, además representa el esfuerzo llevado a cabo en cada uno de los flujos descritos según la fase.

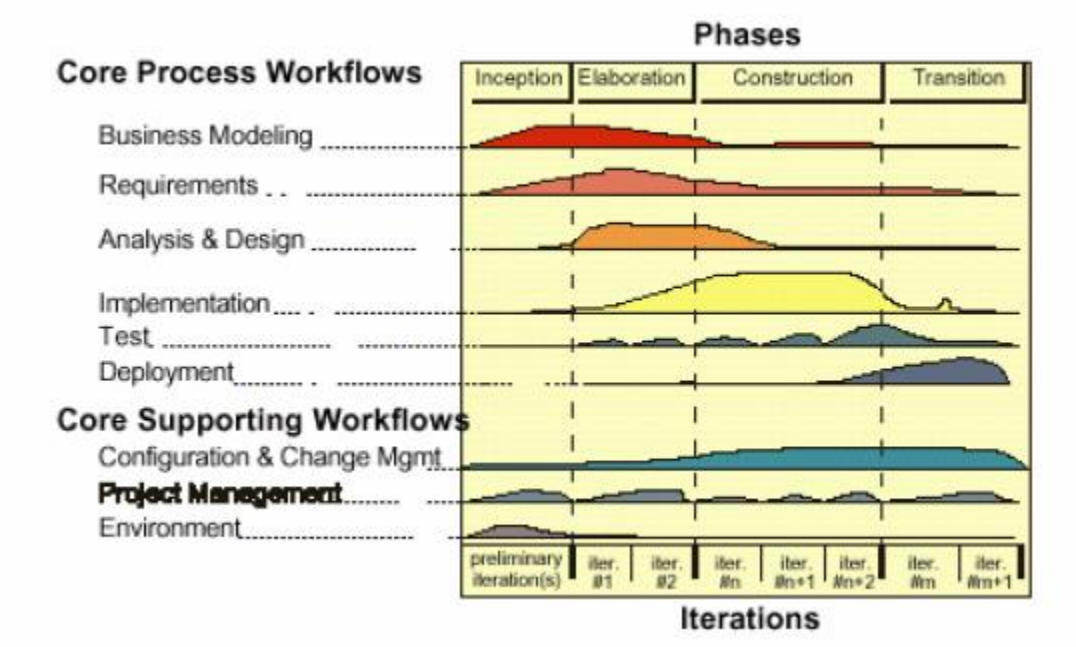


Figura 3. Fases de desarrollo de RUP

Fase 1: Preparación inicial: "Inception".

Tiene como objetivo principal establecer los objetivos para el ciclo de vida del producto de software. En esta fase se establece el problema a resolver, con el fin de delimitar el alcance del proyecto.

Fase 2: Preparación detallada: "Elaboration".

Su objetivo principal es plantear la arquitectura para el ciclo de vida del producto. En esta fase se realiza la captura de la mayor parte de los requerimientos funcionales, manejando los riesgos que interfieran con los objetivos del sistema, acumulando la información necesaria para el plan de construcción y obteniendo suficiente información para hacer realizable el cumplimiento de los objetivos.

Fase 3: Construcción: "Construction".

Su objetivo principal es alcanzar la capacidad operacional del producto. En esta fase a través de sucesivas iteraciones e incrementos se desarrolla un producto de software listo para operar, éste es frecuentemente llamado versión beta.

Fase 4: Transición: "Transition".

Su objetivo principal es realizar la entrega del producto de software funcionando, una vez realizadas las pruebas de aceptación por un grupo especial de usuarios y habiendo efectuado los ajustes y correcciones que sean requeridos.

2.- Descripción de la metodología

2.1.- Orientación a objetos

Como metodología de desarrollo se utilizó la Orientación a Objetos. Mediante esta metodología el dominio del problema se caracteriza mediante un conjunto de objetos con atributos y comportamientos específicos. Los objetos se manipulan mediante una colección de funciones llamadas métodos, y se comunican entre ellos mediante un protocolo de mensajes. Esta metodología entrega como producto un conjunto de modelos orientados a objetos. Estos modelos describen los requisitos, el diseño, el código y los procesos de pruebas para un sistema o producto.

2.1.1.- Programación orientada a objetos

Es una forma de programación que ha ingresado fuertemente en el desarrollo de aplicaciones comerciales en el último tiempo, superando a la programación estructurada (Pascal, Basic, C, Fortran, etc.) y aplicaciones 4GL

(aquellas que permiten el desarrollo de aplicaciones directamente sobre Bases de Datos). Esta forma de programación se basa en el uso de clases y objetos. Las clases son plantillas que permiten la creación de objetos, los cuales poseerán una serie de atributos y métodos definidos en la clase desde la cual fueron creados. Estas clases conservan el patrón aplicado a la lógica de capas, es decir, separar las funcionalidades y requerimientos de una aplicación en elementos que interactúen entre sí, pero que permitan su modificación sin afectar mayormente a aquellos objetos con los que interactúan. La gran ventaja sobre la programación estructurada, que basa su lógica en controlar el flujo de las funcionalidades, radica en que la POO modela lo que ocurre en el mundo real a través de las funcionalidades, creando una "imagen" de lo que ocurre con los procesos y los elementos en el negocio.

3.- Herramientas a utilizar

3.1.- Herramientas de desarrollo

3.1.1.- Java

Java es un lenguaje de programación creado por Sun Microsystems, en un intento de resolver simultáneamente todos los problemas que se le plantean a los desarrolladores de software por la proliferación de arquitecturas incompatibles, tanto entre las diferentes máquinas como entre los diversos sistemas operativos y sistemas de ventanas que funcionaban sobre una misma máquina, añadiendo la dificultad de crear aplicaciones distribuidas en una red como Internet.

Las características principales que ofrece Java respecto a cualquier otro lenguaje de programación, son las que a continuación se describen [Web-5].

Simple

Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. C++ es un lenguaje que adolece de falta de seguridad, pero C y C++ son lenguajes más difundidos, por ello Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje.

Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el garbage collector (reciclador de memoria dinámica). No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello y como es un thread de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria.

Java reduce en un 50% los errores más comunes de programación con lenguajes como C y C++ al eliminar muchas de las características de éstos, entre las que destacan:

- Aritmética de punteros
- No existen referencias
- Registros (struct)
- Definición de tipos (typedef)
- Macros (#define)
- Necesidad de liberar memoria (free)

Orientado a objetos

Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas cosas para mantener el objetivo de la simplicidad del lenguaje. Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulamiento, herencia y polimorfismo. Las plantillas de objetos son llamadas, como en C++, clases y sus copias, instancias. Estas instancias, como en C++, necesitan ser construidas y destruidas en espacios de memoria.

Robusto

Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error.

Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.

Portable

Más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. Los enteros son siempre enteros y además, enteros de 32 bits en complemento a 2. Además, Java construye sus interfaces de usuario a través de un sistema abstracto de ventanas de forma que las ventanas puedan ser implantadas en entornos Unix, Pc o Mac.

3.1.2.- Entorno de desarrollo: Netbeans

Al utilizar el lenguaje de desarrollo orientado a objetos Java, se hace necesario contar con alguna herramienta que permita el soporte de este lenguaje, en otras palabras es necesario contar con un entorno de desarrollo que permita el correcto desarrollo de código en este lenguaje, como así también la correcta administración de los packages y las

clases creadas en la etapa de construcción del sistema. En vista de esta necesidad tan obvia, se ha escogido como entorno de desarrollo a Netbeans. Esta herramienta de desarrollo sirve a los programadores para escribir, compilar, corregir errores y para ejecutar programas. Existe también un número enorme de módulos para extender Netbeans IDE. Cabe destacar que Netbeans IDE es un producto libre y gratuito, sin restricciones de utilización, no así otros entornos de desarrollo como Jbuilder o Borland J++ [Web-3].

3.2.- Herramienta de gestión

3.2.1.- Microsoft Project

Como mecanismo de control se ha utilizado la aplicación Microsoft Project la cual proporciona herramientas para la definición de tareas dentro del proyecto y seguimiento de las mismas, manejo de la calendarización y herramientas para generar informes y diagramas como cartas Gantt. Project es una aplicación de Microsoft que ayuda al usuario a crear planes de proyectos, comunicarlos a otros usuarios y adaptarse a los cambios a medida que éstos se van produciendo. Es un sistema de planificación de proyectos versátil y fácil de utilizar.
Microsoft Project funciona, en muchos sentidos, en forma similar a otras aplicaciones de Microsoft. Las barras de menús, los comandos, las barras de herramientas, los menús contextuales y los cuadros de diálogo tienen mucho en común con Microsoft Excel, Microsoft Word y Microsoft PowerPoint, lo cual facilitará los primeros pasos de su uso.

3.3.- Herramienta de administración de bases de datos

3.3.1.- Microsoft SQL Server

Microsoft SQL Server 7 es un sistema gestor de bases de datos relacionales (SGBDR). Una base de datos relacional proporciona una forma de organizar información almacenándola en tablas de bases de datos. La información relacional se puede agrupar en tablas, y también se pueden definir relaciones entre tablas; de ahí el nombre, base de datos relacional. Los usuarios acceden a la información que está en el servidor a través de una aplicación. Los administradores acceden al servidor directamente para realizar tareas de configuración, administrativas y de mantenimiento de la base de datos. SQL Server es una base de datos dimensionable, lo que quiere decir que puede almacenar cantidades de datos y que puede soportar muchos usuarios accediendo a los datos al mismo tiempo.

SQL Server nació en 1989 y ha cambiado de forma significativa desde entonces. Se han realizado grandes mejoras de dimensionabilidad, la integridad, la facilidad de administración, el rendimiento y las características del producto.

4.- Requerimientos del sistema

4.1.- Análisis de requerimientos

En este punto se estudian los requerimientos del sistema; para lograr una mejor comprensión, se ha agregado a la descripción de éstos, un ejemplo para cada caso con el objeto de que se entiendan de una manera mas clara.

4.1.1.- Descripción de los requerimientos

4.1.1.1.- Requerimientos funcionales

El sistema debe ser capaz de:

- **Encontrar relaciones entre el comportamiento de los clientes**

  Dado un conjunto de datos relativos al comportamiento de los clientes, el sistema debe entregar relaciones de implicancia entre las acciones que generalmente hacen los clientes de manera conjunta, y la probabilidad asociada a dicha secuencia de acciones.

  Ejemplo.
  La respuesta puede tener la siguiente forma:

  Si un cliente tiene entre 40-50, años, y viene en el mes de febrero con su familia, existe un 60% de probabilidad de que gaste un monto superior a $ 500.000 en su estadía.

  Formalmente:

  Edad(X, 40...50) /\ fecha _ visita(X, 1/02...28/02) /\ familia(X, SI) => gasto(X, 500.000...1.000.000)
  Probabilidad 60%

- **Predecir**

  Teniendo un conjunto finito de respuestas a una pregunta específica, dadas estas respuestas por un conjunto de valores de un atributo en particular, crear un modelo que resuelva esta pregunta recibiendo un conjunto de valores diferentes.

La predicción recibe como entrada un conjunto de casos en que se conoce la respuesta, y entrega como salida un modelo que servirá para dar respuesta a la misma pregunta pero con otros valores de los atributos.

Ejemplo.
El gerente desea clasificar los clientes de acuerdo al monto gastado durante su estadía, la respuesta a esta pregunta estaría dada por algunas características específicas del cliente (atributos), tales como: nacionalidad, edad, profesión, temporada de visita, etc. El modelo entregado debería servir para distinguir qué atributo es el más influyente en la respuesta obtenida, así como también, qué respuestas se obtendrían si se modifican los valores de los atributos.

Una respuesta dada por el sistema podría ser la siguiente:

Si nacionalidad=ARGENTINA, edad=[40-50], temporada=ALTA, profesion=ABOGADO => monto_gastado=[200.000-300.000]

Los atributos se encuentran ordenados desde el más influyente al menos influyente en la respuesta obtenida.

- **Agrupar**

A partir de un conjunto de datos históricos, el sistema debe ser capaz de crear grupos de clientes con atributos similares, sin que se entregue una etiqueta, o una consulta para definir el grupo, la idea es que se agrupen solo por la semejanza de sus atributos, así se podrán obtener nuevos tipos de clientes.

Ejemplo.
Este procedimiento permitirá al usuario dirigir una estrategia publicitaria a un cierto grupo de clientes que tengan preferencias bien definidas dadas por la técnica de agrupamiento, evitando así realizar inversiones en publicidad que no sea efectiva.

Una posible respuesta sería la siguiente:

Cantidad de grupos: 3
GRUPO 1: edad=30, monto_gastado=120.000, temporada=ALTA
GRUPO 2: edad=45, monto_gastado=250.000, temporada=ALTA
GRUPO 3: edad=60, monto_gastado=480.000, temporada=BAJA

Nota: El atributo monto_gastado es el promedio de lo que gastan los cliente pertenecientes al grupo, durante su estadía

- **Presentación de resultados en forma amigable para el usuario**

Creación de gráficos:
El sistema debe ser capaz de crear gráficos con los resultados de las operaciones de Data Mining realizadas.

Interpretación de resultados:
Los resultados obtenidos después de aplicados los algoritmos de Data Mining deben ser presentados al usuario de manera amigable y comprensible, y no en términos demasiado técnicos.

4.1.1.2.- Requerimientos no funcionales

- **Facilidad de uso**

El software debe ser fácil, de usar a pesar de realizar tareas complejas, debe tener una interfaz amigable para el usuario, ocultando los detalles menos relevantes y debe tener la mayoría de las opciones con valores por defecto, aunque debe ser posible cambiarlas, para que a medida que el usuario se vaya familiarizando con el sistema, le vaya sacando mas provecho a éste.

5.- Técnicas de Data Mining utilizadas

Las técnicas son implementaciones específicas de los algoritmos que se utilizan para llevar a cabo las operaciones de Data Mining. En la siguiente sección se presenta la explicación y el funcionamiento de los algoritmos elegidos en el proyecto.

5.1.- Reglas de asociación

Las reglas de asociación tienen como objetivo encontrar asociaciones interesantes entre un gran conjunto de datos. Una regla de este tipo propone sentencias como: “si un cliente compra A y B entonces compra C el 80% de las veces”.

Como se puede ver en esa expresión, se establecen relaciones entre los atributos (A B y C). Aquí no se tiene un atributo que actúe como resultado de una función a modo de clase (a diferencia de los árboles de decisión), sino que en el consecuente de la regla se pueden tener cualquiera de los atributos de los datos que se estudian, y además no tiene por qué ser único, en el lado derecho se puede encontrar también cualquier combinación de estos, igual que en el antecedente [Ag93]. En la Figura 4 se presenta el esquema de las reglas de asociación.

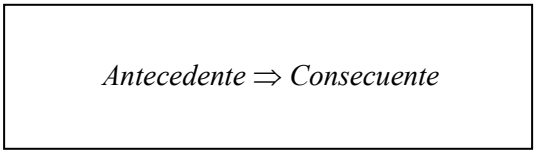


Figura 4. Esquema de una regla de asociación

Un problema de este tipo es inabordable desde el punto de vista de la clasificación.

Del campo de la minería de datos, donde se manejan masivas bases de datos recogidas por industrias y comercios, surgió la idea de encontrar asociaciones interesantes entre esa gran cantidad de datos que fuesen de ayuda a la hora de tomar decisiones de negocios.

Aunque han sido aplicadas en muchos dominios para la ayuda de toma de decisiones y la predicción, la principal aplicación que ha fomentado su uso es el análisis de datos de ventas, lo que generalmente se ha denominado la "cesta de la compra". El proceso ayuda a analizar los hábitos de compra de los clientes encontrando asociaciones entre los diferentes artículos que estos meten en sus "cestas".

Poniendo de ejemplo un hotel, podría darse el caso de que el análisis de las transacciones de ventas arrojara el resultado de que la gente que solicita habitaciones con vista al mar, tiende a consumir preferentemente un determinado menú. Esto podría inducir al gerente a planear estrategias relacionadas a los menús a ofrecer para estos clientes objetivo, de manera que se ayude a incrementar las ventas de menús. El sistema que realiza el análisis representaría este resultado con una regla como:

Habitación_vista_mar => Menú5 [soporte = 2%, confianza = 60%]

(Menú5 se refiere a un menú específico)

En el ejemplo visto aparecen dos medidas no explicadas hasta el momento, el ***soporte*** y la ***confianza***, las cuales se describen a continuación.

5.1.1.- El Soporte y la Confianza

El soporte y la confianza son dos medidas de lo interesante que puede ser una regla.

El ***soporte*** es el porcentaje de transacciones analizadas en las cuales una "Habitación con vista al mar" y el "Menú5" fueron vendidos juntos.

La ***confianza*** indica la proporción de clientes de entre los que solicitaron habitación con vista al mar, que también compraron el Menú5.

Ejemplo:
Se tienen los siguientes datos en la tabla de la figura 5.

Vista_Habitacion	Menu	Nacionalidad
Mar	Menu5	Argentina
Mar	Menu1	Chilena
Avenida	Menu2	Chilena
Avenida	Menu3	Chilena
Mar	Menu5	Chilena
Mar	Menu5	Peruana
Mar	Menu4	Argentina

Figura 5. Tabla de habitación, menú consumido y nacionalidad del cliente

Para estos datos se tiene que la regla "Mar=>Menu5" es respaldada por las siguientes medidas:

Soporte:

$$\frac{\text{Transacciones que poseen Vista al Mar y Menu5 conjuntamente}}{\text{Total de transacciones de la BD}}$$

$$\frac{3}{7} = 0{,}428571$$

Confianza:

$$\frac{\text{Transacciones que poseen Vista al Mar y Menu5 conjuntamente}}{\text{Total de transacciones que poseen Vista al Mar}}$$

$$\frac{3}{5} = 0{,}6$$

5.1.2.- Conceptos básicos

Definición formal del problema [Ag93]:

- Sea $I = \{i_1 , i_2 \ldots\ldots i_m\}$ un conjunto de atributos, denominados **ítems**.
- Sea D un conjunto de transacciones donde cada transacción T es un conjunto de ítems, denominado **itemset**, de manera que $T \subseteq I$.
- A cada transacción se le asocia un identificador único: TID.
- Sea A un itemset, se dice que T contiene a A si y solo si $A \subseteq T$.

Entonces una regla de asociación es una implicación de la forma:

$$A \Rightarrow B, \text{ donde } A \subset I, B \subset I \text{ y } A \cap B = \varnothing$$

La regla $A \Rightarrow B$ se cumple en el conjunto de transacciones D con un ***soporte*** s, donde s es el porcentaje de transacciones en D que contienen $A \cup B$.

La regla $A \Rightarrow B$ tiene una ***confianza*** c en el conjunto de transacciones D si c es el porcentaje de transacciones en D que conteniendo A también contienen B.
En términos de probabilidades se tiene que:

$$\text{Soporte } (A \Rightarrow B) = P(A \cup B)$$

$$\text{Confianza } (A \Rightarrow B) = P(B|A)$$

Idea:

Dado un conjunto de transacciones *D*, se puede definir el problema de la **generación de reglas de asociación** como encontrar todas las reglas de asociación que satisfacen un mínimo umbral de soporte (min_sup) y un mínimo umbral de confianza (min_conf). A estas reglas se las denomina reglas **fuertes**.

Un itemset que contiene k ítems es un k-itemset. En el ejemplo del hotel se tendría el 2-itemset {Habitación_vista_al_mar, Menú5).
La **frecuencia de ocurrencia** o **recuento de soporte** de un itemset es el número de transacciones que contienen el itemset. Un itemset satisface el soporte mínimo si:

$$\text{frecuencia de ocurrencia} \geq \text{min\_sup x nº transacciones en } D$$

El número de transacciones requeridas para que un itemset satisfaga el soporte mínimo es denominado como **recuento de soporte mínimo**. Por lo que la expresión anterior equivaldría a:

$$\text{frecuencia de ocurrencia} \geq \text{recuento de soporte mínimo}$$

Un itemset que satisface el soporte mínimo es un itemset **frecuente**.
Al conjunto de k-itemsets frecuentes se le denota por L_k.

El proceso de generación de reglas de asociación conlleva dos pasos generales:

- Encontrar todos los itemsets frecuentes.
- Generar reglas de asociación fuertes a partir de los itemsets frecuentes.

5.1.3.- Algoritmo elegido: A priori

El algoritmo elegido es un algoritmo básico para encontrar itemsets frecuentes. Luego de explicar su funcionamiento, se mostrará la manera de generar reglas de asociación con los itemsets frecuentes encontrados.

El algoritmo A priori se basa en el conocimiento previo o "a priori" de propiedades de los itemsets frecuentes para reducir el espacio de búsqueda y aumentar la eficiencia. A esa propiedad se la llama propiedad A priori.

Para desarrollar su tarea A priori usa una aproximación iterativa denominada búsqueda *level-wise*, donde los k-itemsets son usados para explorar los (k+1)-itemsets [Ag93].

Para lograr una mayor comprensión del modo de funcionamiento del algoritmo, se presentará a continuación un ejemplo práctico con datos relativos al giro de la empresa con la que se trabaja.

Suponiendo que se tiene la BD relacional con las transacciones del hotel. En ella se tienen las transacciones que han hecho lo clientes, cada una con sus propios ítems.

- *D* es el número de transacciones de esa BD.
- *I* es el conjunto de servicios (ítems) disponibles en el hotel, por ejemplo: {Habitación_vista_mar, Menú5, Consumo_Bar, Lavandería, Desayuno2}.
- Cada transacción *T* que ha hecho un cliente es un conjunto de servicios, es decir un itemset. Las reglas que se generan son Booleanas, por lo tanto estas transacciones contendrán la presencia o ausencia de los artículos disponibles en esa compra, aunque solo se mantendrán en cada itemset los elementos que se han comprado. Por ejemplo, alguien podría haber tomado el servicio {Menú5, Consumo_Bar}. Claramente se aprecia que $T \subseteq I$.

El algoritmo A priori trabajará con conjuntos de itemsets para encontrar aquellos que superen el soporte mínimo y agruparlos en conjuntos de k ítems o servicios, a los que se denomina L_k. Por lo tanto se podría tener el 2-itemset {Habitación_vista_mar, Menú5} que supere el soporte mínimo dentro de L_2 y que finalmente acabe generando una regla como la que se vio en la Figura 4.

La propiedad A priori

Si un itemset *I* no satisface el umbral de soporte mínimo P(*I*) < min_sup, entonces no es frecuente (por definición). Si se añade un ítem *A* al itemset *I*, el itemset que resulta no puede tener más ocurrencias en el conjunto de transacciones que el original, es decir, $I \cup A$ tampoco es frecuente, o lo que es lo mismo P($I \cup A$) < min_sup.

$$P(I) < min\_sup \Rightarrow P(I \cup A) < min\_sup$$

Si se tiene un itemset como {Habitación_vista_mar, Consumo_Bar} que se encuentra contenido en un número limitado de transacciones, el itemset {Habitación_vista_mar, Consumo_Bar, Menú5} se encontrará como mucho en esas mismas transacciones, pero no más.

A este tipo de propiedades se las llama *antimonotonía*, ya que si un conjunto no pasa un test, todos sus superconjuntos fallarán también ese mismo test.

5.1.3.1.- Funcionamiento del algoritmo

El algoritmo encuentra itemsets frecuentes generando sucesivamente conjuntos L_k a partir de los L_{k-1}. Para ello sigue un proceso de dos pasos, donde se hace uso de la propiedad A priori.

- **Paso de unión (join)**

  Para encontrar L_k, se genera un conjunto de k-itemsets candidatos generado por la unión (join) de L_{k-1} con sí mismo. Este conjunto de candidatos se denota por C_k.

  Se tiene l_1 y l_2 dos itemsets en L_{k-1}, donde $l_i[j]$ hace referencia al j-ésimo item en l_i (con lo que $l_i[k]$ denotaría el último elemento del itemset). Cabe destacar que el algoritmo A priori asume que los ítems dentro de una transacción o un itemset están ***ordenados lexicográficamente***. Entonces se lleva a cabo la unión L_{k-1} L_{k-1}, para lo cual los miembros L_{k-1} deben cumplir que sus (k-2) ítems sean comunes (hay que darse cuenta de que se tiene (k-1)-itemsets, y por tanto k-1 es el último elemento de los l_i). Esto quiere decir que l_1 y l_2 se unirán si:

  $$l_1[1] = l_2[1] \wedge l_1[2] = l_2[2] \wedge \ldots \wedge l_1[\text{k-2}] = l_2[\text{k-2}] \wedge l_1[\text{k-1}] < l_2[\text{k-1}]$$

  La última condición asegura que no se generarán duplicados. El resultado de esta unión será $l_1[1]l_1[2]\ l_1[3]\ldots l_1[\text{k-1}]\ l_2[\text{k-1}]$.

- **Paso de poda**

  C_k es un superconjunto de L_k, es decir, sus miembros pueden o no ser frecuentes, pero todos los k-itemsets frecuentes están incluidos en C_k. Un examen de la base de datos para determinar la frecuencia de ocurrencia o recuento de soporte de cada candidato en C_k (es decir, el número de transacciones que contienen a ese itemset) determinará L_k. Para entrar en este conjunto el candidato debe tener una frecuencia mayor o igual al recuento de soporte mínimo. Pero este conjunto de candidatos puede ser muy grande, además del enorme tamaño de la base de datos, por lo cual llevaría un gran trabajo computacional. Por ello se usa la propiedad A priori para reducir el tamaño de C_k. Como ningún (k-1)-itemset que no sea frecuente puede ser subconjunto de un k-itemset frecuente, entonces se miran los subconjuntos de (k-1) ítems (osea, formamos (k-1)-itemsets) que se puedan formar a partir de cada k-itemset candidato, y si no están en L_{k-1}, que contiene a los que son frecuentes, entonces ese candidato tampoco puede ser frecuente y se elimina de C_k. Este test de subconjuntos se puede realizar rápidamente si se mantiene una tabla hash de todos los itemsets frecuentes.

  A continuación se presenta un ejemplo de lo explicado anteriormente. Se tendrá una base de datos con nueve transacciones, es decir |D| = 9, representada por la tabla de la Figura 6.

TID	Lista de ítems
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I3
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1,I2,I3

Figura 6. Tabla de transacciones

El algoritmo realizará el siguiente proceso iterativo:

1. En la primera iteración cada ítem es un miembro del conjunto 1-itemsets candidato C_1. Simplemente se examinan todas las transacciones para contar el número de ocurrencias de cada item.

   Se supone que el recuento de soporte mínimo es 2 (con lo que se tendría soporte mínimo min_sup = 2/9 = 22%). Se comparan las frecuencias o recuentos de soporte de cada 1-itemset candidato y los que satisfagan ese mínimo pasarán a formar parte de L_1 (Figura 7).

C_1	
Itemset	**Recuento de soporte**
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1	
Itemset	**Recuento de soporte**
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Figura 7. Itemsets candidatos C_1 y frecuentes L_1, primera iteración

Todos los itemsets satisfacen el mínimo, así que pasan al conjunto de frecuentes.

2. Se realiza la unión L_1 L_1, para generar los candidatos C_2 que ayuden a encontrar L_2. C_2 contendrá una combinación de $|L_1|$ elementos cogidos de 2 en 2, cada uno de ellos un 2-itemset.

   Se examinan estos nuevos itemsets en la base de datos para comprobar su recuento de soporte y aquellos candidatos que tengan el soporte mínimo pasan a L_2 (Figura8).

C_2	
Itemset	**Recuento de soporte**
{I1,I2}	4
{I1,I3}	4
{I1,I4}	1
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2
{I3,I4}	0
{I3,I5}	1
{I4,I5}	0

L_2	
Itemset	**Recuento de soporte**
{I1,I2}	4
{I1,I3}	4
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2

Figura 8. Itemsets candidatos C_2 y frecuentes L_2, segunda iteración

3. Se continúa con la generación de los 3-itemsets candidatos. Para ello se realiza la unión L_2 L_2. El resultado es el siguiente conjunto:

   L_2 L_2 = {{I1,I2,I3},{I1,I2,I5},{I1,I3,I5},{I2,I3,I4},{I2,I3,I5}}.

   Éstas son todas las combinaciones posibles de elementos que se pueden unir. No se pueden unir elementos como {I1,I2} con {I2,I5} porque difieren en el primer ítem.

   Ahora se debería realizar el examen sobre la tabla de transacciones para calcular la frecuencia de cada itemset, sin embargo se puede reducir este proceso podando el conjunto de candidatos gracias a la propiedad A priori.

Este es el procedimiento a seguir:

- Se forman los subconjuntos 2-itemset del elemento {I1,I2,I3}, que son {I1,I2}, {I2,I3} e {I1,I3}. Se comprueban que todos son miembros de L_2, por tanto son todos frecuentes, y su superconjunto puede ser también frecuente, por lo tanto se mantiene en C_3 como candidato.
- Los subconjuntos 2-itemset del elemento {I1,I2,I5} son {I1,I2}, {I2,I5} e {I1,I5}. Todos miembros de L_2, así que se mantiene en C_3.
- Los subconjuntos 2-itemset del elemento {I1,I3,I5} son {I1,I3}, {I3,I5} e {I1,I5}. Aquí se ve que {I3,I5} no forma parte de L_2, así que no es frecuente y consecuentemente su superconjunto tampoco puede serlo, de manera que se elimina de C_3.
- Los subconjuntos 2-itemset del elemento {I2,I3,I4} son {I2,I3}, {I3,I4} e {I2,I4}. {I3,I4} no es miembro de de L_2, así que se elimina de C_3.
- Los subconjuntos 2-itemset del elemento {I2,I3,I5} son {I2,I3}, {I3,I5} e {I2,I5}. {I3,I5} no es miembro de de L_2, así que se elimina de C_3.
- Los subconjuntos 2-itemset del elemento {I2,I4,I5} son {I2,I4}, {I4,I5} e {I2,I5}. {I4,I5} no es miembro de de L_2, así que se elimina de C_3.

De esta manera se ha reducido el número de candidatos de 6 a 2 itemsets. Ahora se hace el recuento y se determina L_3 (Figura 9).

C_3	
Itemset	**Recuento de soporte**
{I1,I2,I3}	2
{I1,I2,I5}	2

L_3	
Itemset	**Recuento de soporte**
{I1,I2,I3}	2
{I1,I2,I5}	2

Figura 9. Itemsets candidatos C_3 y frecuentes L_3, tercera iteración

4. Finalmente se realiza la unión L_3 L_3, dando como resultado el 4-itemset {I1,I2,I3,I5}. De aquí se obtiene el subconjunto {I2,I3,I5} que no es frecuente al no encontrarse en L_2, por lo tanto se poda, y se queda sin candidatos $C_4 = \varnothing$, con lo que el algoritmo termina.

Algoritmo A priori*: encuentra itemsets frecuentes usando una aproximación iterativa level-wise basada en la generación de candidatos.*

Entrada: Base de datos, *D,* de transacciones; umbral de soporte mínimo, min_sup.

Salida: *L*, itemsets frecuentes en *D*.

Método:

```
L1 = encontrar_1-itemsets_frecuentes(D);

for(k = 2; Lk-1 ≠ Ø; k++){
  Ck = a priori_gen(Lk-1, min_sup);
  para cada transaction t ∈ D {                  // examinar D para recuento
    Ct = subconjunto(Ck, t);                     // obtiene los subconjuntos de t
                                                 // que son candidatos
    para cada candidato c ∈ Ct
      c.recuento++;
  }
  Lk = { c ∈ Ck | c.recuento ≥ min_sup }
}
return L = ∪k Lk;


procedimiento apriori_gen(Lk-1: (k-1)-itemsets frecuentes, min_sup: umbral de soporte mínimo){

  para cada itemset l1 ∈ Lk-1
    para cada itemset l2 ∈ Lk-1
      si(l1[1]=l2[1] ∧ l1[2]=l2[2] ∧…∧ l1[k-2]=l2[k-2] ∧ l1[k-1]<l2[k-1])
            entonces{
        c = l1      l2;  // paso de unión: genera candidatos
        si tiene_subconjunto_infrecuente(c, Lk-1) entonces
          eliminar c;            // paso de poda: elimina candidatos infructuosos
        sino
          agregar c a Ck;
      }
  return Ck;
}


procedimiento tiene_subconjunto_infrecuente(c: k-itemset candidato; Lk-1: (k-1)itemsets frecuentes){
            // usa el conocimiento previo
  para cada (k-1)-subconjunto s de c
    si s ∉ Lk-1 entonces
      return TRUE;
  return FALSE;
}
```

Figura 10. Algoritmo A priori

La figura 10 muestra el algoritmo A priori mediante pseudo-código. Comienza encontrando los 1-itemsets frecuentes, que forman L_1. A partir de ahí usa los L_{k-1} para generar los candidatos C_k. El procedimiento apriori_gen es el encargado de generarlos, para lo cual realiza primero el paso de unión, y luego, usando la propiedad A priori, poda aquellos k-itemsets que no pueden ser frecuentes. Una vez tenemos los candidatos finales se examina la base de datos. De cada transacción se obtienen los subconjuntos que coinciden con los candidatos, y por cada uno de ellos se acumula en uno el recuento del candidato.

- **La generación de reglas de asociación**

Una vez encontrados los itemsets frecuentes, la generación de reglas de asociación fuertes a partir de ellos es inmediata. Para ello se usa la expresión de la confianza basada en probabilidades:

$$\text{Confianza } (A \Rightarrow B) = P(B|A) = \frac{\text{Recuento\_de\_soporte}(A \cup B)}{\text{Recuento\_de\_soporte}(A)}$$

Donde los recuentos de soportes son el número de transacciones que contienen esos itemsets. Para generar las reglas se debe hacer:

- Para cada itemset frecuente *l*, generar todos los subconjuntos no vacíos de *l*.
- Para cada subconjunto no vacío *s* de *l*, crear la regla "$s \Rightarrow (l - s)$" si recuento_de_soporte(*l*) / recuento_de_soporte(*s*) ≥ min_conf, donde min_conf es el umbral mínimo de confianza.

Por el método de generación de los itemsets, las reglas automáticamente satisfacen el soporte mínimo.

Volviendo al ejemplo anterior se generarían así las correspondientes reglas a partir de uno de los itemsets obtenidos *l* = {I1,I2,I5}. Los subconjuntos no vacíos de *l* son: {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2} e {I5}. Y las reglas que se crean a partir de ellos:

$I1 \wedge I2 \Rightarrow I5$ confianza = 2/4 = 50%
$I1 \wedge I5 \Rightarrow I2$ confianza = 2/2 = 100%
$I2 \wedge I5 \Rightarrow I1$ confianza = 2/2 = 100%
$I1 \Rightarrow I2 \wedge I5$ confianza = 2/6 = 33%
$I2 \Rightarrow I1 \wedge I5$ confianza = 2/7 = 29%

$I5 \Rightarrow I1 \wedge I2$ confianza = 2/6 = 100%

Si se establece el umbral mínimo de confianza como el 70%, solo 3 de esas reglas serían fuertes, y son las que se obtendrían finalmente.

5.2.- Clasificación y Predicción

5.2.1.- Algoritmo de predicción: C 4.5

Se usará la técnica conocida como árbol de clasificación, en éste se representa el conocimiento basándose en un particionamiento recursivo del dominio de definición de las variables predictoras, en el caso del sistema que se está desarrollando éstas serán los campos de la base de datos. Esta técnica se utilizará para generar modelos que permitan predecir el comportamiento futuro de los clientes a partir de comportamientos anteriores.

5.2.1.1.- Algoritmo básico

La figura 11 presenta el algoritmo denominado TDIDT (Top Down Induction of decision Trees) el cual puede ser contemplado como uniformizador de la mayoría de los algoritmos de árboles de clasificación.

La idea del algoritmo TDIDT es que mientras que todos los patrones que se correspondan con una determinada rama del árbol de clasificación no pertenezcan a una misma clase, se seleccione la variable que de entre las no seleccionadas en esa rama sea la más informativa o la más idónea con respecto de un criterio previamente establecido. La elección de esta variable sirve para expandir el árbol en tantas ramas como posibles valores toma dicha variable.

Algoritmo TIDT

Entrada: R: Un conjunto de atributos no categóricos
C: El atributo categórico
S: Un conjunto de entrenamiento
Salida: Un árbol de decisión;

Método:
Si la S consta de registros todo con el mismo valor para
El atributo categórico **entonces**
Devuelva un solo nodo con ese valor;
Si R está vacío, **entonces** devuelva un solo nodo con el valor
más frecuente de los valores del atributo categórico
encontrado en registros de S; (nótese que entonces allí
se producirán errores, eso es, estos registros serán
clasificados erróneamente);
Sea D el atributo que mejor clasifica el conjunto de entrenamiento
Entre los atributos en R;
Sea { dj j = 1,2.., M } los valores del atributo D;
Sea { Sj J = 1,2.., M } los subconjuntos de S
Asociados con los registros D del atributo D;
Devuelva un árbol con raíz D etiquetada y arcos etiquetados
d1, d2.., dm enlazando estos a los respectivos árboles:

TIDT(R-{ D }, C, S1), TIDT(R-{ D }, C, S2).., TIDT (R-{ D }, C, Sm);

Figura 11. Algoritmo TIDT

5.2.1.2.- Algoritmo ID3

En la figura 12 se muestra el algoritmo ID3, debido a que el Algoritmo elegido (C4.5) es una extensión de éste. Se eligió el algoritmo C4.5 puesto que presenta una ventaja sobre el algoritmo ID3 ya que utiliza un criterio para elegir el atributo más influyente en el resultado obtenido, que no se ve afectado por la cantidad de posibles valores de los atributos, lo cuál si sucede con el algoritmo ID3.

El algoritmo ID3 selecciona la variable más informativa en base a la mayor ganancia de información (information gain).

Algoritmo ID3

Entrada: R: Un conjunto de atributos no categóricos
C: El atributo categórico,
S: Un conjunto de entrenamiento
Salida: Un árbol de decisión;

Método:
Si la S consta de registros todo con el mismo valor para
El atributo categórico **entonces**
Devuelva un solo nodo con ese valor;
Si R está vacío, **entonces** devuelva un solo nodo con el valor
más frecuente de los valores del atributo categórico
encontrado en registros de S; (nótese que entonces allí
Se producirán errores, eso es, estos registros serán
clasificados erróneamente);
Sea D el con mayor ganancia (D,S);
Entre los atributos en R;
Sea { dj j = 1,2.., M } los valores del atributo D;
Sea { Sj J = 1,2.., M } los subconjuntos de S
Asociados con los registros D del atributo D;
Devuelva un árbol con raíz D etiquetada y arcos etiquetados
d1, d2.., dm enlazando estos a los respectivos árboles:

TIDT(R-{ D }, C, S1), TIDT(R-{ D }, C, S2).., TIDT (R-{ D }, C, Sm);

Figura 12. Algoritmo ID3

5.2.2.- Funcionamiento del algoritmo

Quinlan (1993) [Qui93] propone una mejora del algoritmo ID3, al que denomina C4.5. El Algoritmo C4.5 se basa en la utilización del criterio ratio de ganancia (gain ratio), definido como:

$$\frac{ganacia(D,S)}{div\inf(D,S)}$$

Donde divinf(D,S) es la in formación de la división de S en base a el valor del atributo categórico D. así divinf(D,S) queda como:

$$Entropia(\left|\frac{S_1}{S}\right|,\left|\frac{S_2}{S}\right|,...,\left|\frac{S_m}{S}\right|)$$

Donde ($S_{1,}S_{2,...,}S_m$) es la partición de S causada por el valor de D.

De esta manera se consigue evitar que las variables con mayor número de posibles valores salgan beneficiadas en la selección. Además el algoritmo C4.5 incorpora una poda del árbol de clasificación una vez que este ha sido inducido. La poda está basada en la aplicación de un test de hipótesis que trata de responder a la pregunta de si merece la pena expandir o no una determinada rama, esto se hace con el objetivo de simplificar el árbol, para que no se produzca el fenómeno de sobreajuste.

Este test se realiza de la siguiente manera:

Se consideran los siguientes términos:

- N(t): número de ejemplos en el nodo t, en el que se está testando la expansión.
- e(t) número de ejemplos mal clasificados en el nodo t.
- n'(t) corrección por continuidad de e(t), la cual se efectúa sumando ½ a e(t). Es decir, n'(t) = e(t) + ½ .
- h(Tt) denota el número de hojas del subárbol Tt.
- n'(Tt) se obtiene a partir del número de errores existentes en las hojas terminales del sub_arbol Tt,

y se define como:

$$n'(T_t) = \sum_{i=1}^{h(T_t)} e(i) + h(T_t)/2$$

- S(n'(Tt)) definido como la desviación de n'(Tt) a partir de la siguiente fórmula:

$$S(n'(T_t)) = \sqrt{\frac{n'(T_t)\left[N(t) - n'(T_t)\right]}{N(t)}}$$

La decisión acerca de expandir el nodo t, y contemplar el subárbol Tt se toma en base a la siguiente regla:
El nodo t se expande si y sólo si: $n'(T_t) + S(n'(T_t)) \prec n'(t)$

Los conceptos descritos anteriormente se ejemplifican mejor en el siguiente caso.

Considerando que se tiene la tabla de base de datos del hotel de la Figura 13.

Nacionalidad	Sexo	Temporada
Norteamericano	Masculino	Alta
Norteamericano	Masculino	Alta
Norteamericano	Masculino	Alta
Norteamericano	Masculino	Alta
Chilena	Masculino	Alta
Norteamericano	Masculino	Alta
Chilena	Femenino	Alta
Chilena	Masculino	Media
Chilena	Femenino	Baja
Chilena	Masculino	Baja
Chilena	Masculino	Baja
Norteamericano	Masculino	Alta
Chilena	Masculino	Alta
Norteamericano	Femenino	Media
Chilena	Masculino	Media
Norteamericano	Masculino	Alta
Norteamericano	Masculino	Alta
Norteamericano	Masculino	Alta
Norteamericano	Masculino	Alta
Chilena	Femenino	Media
Norteamericano	Masculino	Alta
Norteamericano	Masculino	Media
Norteamericano	Masculino	Alta
Norteamericano	Masculino	Alta
Norteamericano	Masculino	Media
Norteamericano	Masculino	Alta
Norteamericano	Masculino	Alta
Norteamericano	Femenino	Media
Norteamericano	Femenino	Media
Chilena	Femenino	Baja
Chilena	Masculino	Baja
Argentina	Femenino	Alta
Argentina	Masculino	Alta

Argentina	Femenino	Alta
Argentina	Masculino	Alta

Figura 13. Tabla de nacionalidad, sexo y temporada de visita del cliente

5.2.2.1.- Se busca el atributo que mejor clasifique la información

- Se calcula la entropía de cada atributo:

$$Entropia(D) = -\sum p_i \log(p_i)$$

Nacionalidad: existen 20 norteamericanos, 11 chilenos y 4 argentinos.

$$entropía(20,11,4) = -(20/35)\log(20/35) - (11/35)\log(1/35) - (4/35)\log(4/35)$$
$$= 1{,}3438$$

Sexo: hay 26 Hombres y 9 mujeres.

$$entropía(9,26) = -(9/35)\log(9/35) - (26/35)\log(26/35) = 0{,}8224$$

- Se calcula la ganancia de información para cada atributo:

$$ganancia\,(D,S) = Entropia\,(D) - \sum_{v \in val(S)} (|D_V|/|D|) Entropia\,(D_v)$$

Nacionalidad:

$$ganancia\,(Nacionalidad) =$$
$$1{,}3438 - (\frac{20}{35} entropia\,(15,5,0) + \frac{11}{35} entropia\,(3,3,5) + \frac{4}{35} entropia\,(2,2,0)) = 0{,}28209$$

Sexo:

$$ganancia\ (Sexo) = 1{,}3438 - (\frac{26}{35} entropia\ (,5,0) + \frac{9}{35} entropia\ (3,3,5)) = -0{,}31375$$

- Se calcula divinf(D,S)

Donde divinf(D,S) es:

$$Entropia(\left|\frac{S_1}{S}\right|, \left|\frac{S_2}{S}\right|, ..., \left|\frac{S_m}{S}\right|)$$

Nacionalidad:

$$entropía(20,11,4) = -(20/35)\log(20/35) - (11/35)\log(1/35) - (4/35)\log(4/35) = 1{,}3438$$

Sexo:

$$entropía(9,26) = -(9/35)\log(9/35) - (26/35)\log(26/35) = 0{,}8224$$

- Ahora se puede calcular el gain ratio:

$$\frac{ganacia(D,S)}{div\inf(D,S)}$$

Nacionalidad:

$$\frac{0{,}28209}{1{,}3438} = 0{,}20991$$

Sexo:

$$\frac{-0{,}31375}{0{,}8224} = -0{,}38150$$

Con lo cual se determina que nacionalidad es el primer atributo que mejor clasifica los ejemplos.

5.2.2.3.- Ahora se realiza el mismo procedimiento con la primera partición y el atributo que queda, y se obtiene el árbol de la Figura 14

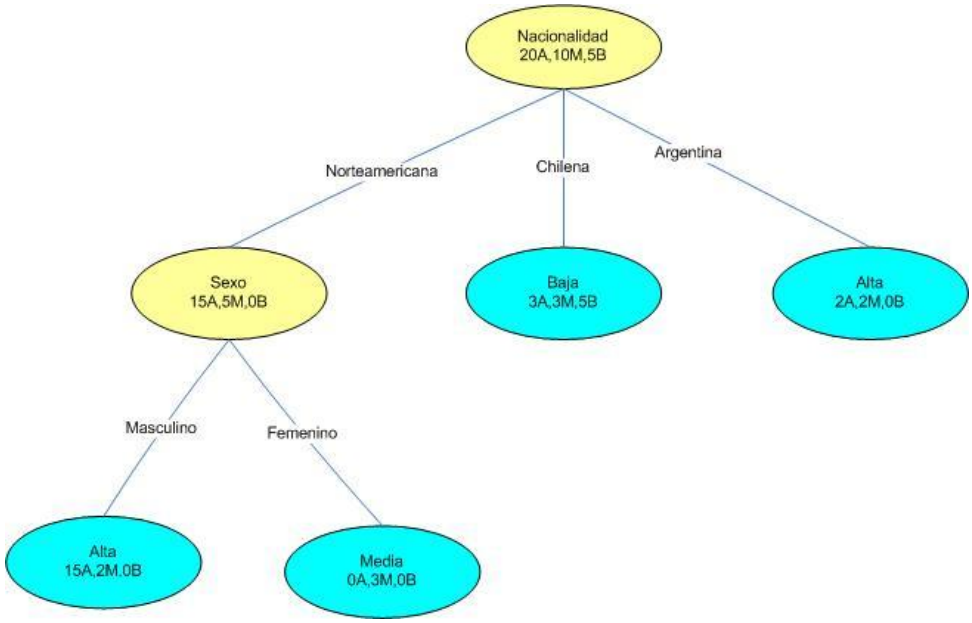


Figura 14. Árbol generado mediante algoritmo C45

5.2.2.4.- Ahora se verá si se debía expandir el árbol o no

Se considerarán los siguientes términos:

- N(t): número de ejemplos en el nodo t, en el que se está testando la expansión En el ejemplo **t=Nacionalidad, N(t) = 35.**

- e(t) número de ejemplos mal clasificados en el nodo t. En el ejemplo,
  e(t) =10+5 =15.
- n'(t) corrección por continuidad de e(t), la cual se efectúa sumando ½ a e(t). Es decir, n'(t) = e(t) + ½ .
- h(Tt) denota el número de hojas del subárbol Tt. En el ejemplo: h(Tt)=4.
- n'(Tt) se obtiene a partir del número de errores existentes en las hojas terminales del sub_arbol Tt,

y se define como:

$$n'(T_t) = \sum_{i=1}^{h(T_t)} e(i) + h(T_t)/2$$

En el ejemplo:

2+0+6+2+4/2 = 12

- S(n'(Tt)) definido como la desviación de n'(Tt) a partir de la siguiente fórmula:

$$S(n'(T_t)) = \sqrt{\frac{n'(T_t)\left[N(t) - n'(T_t)\right]}{N(t)}}$$

En el ejemplo:

$$S(n'(T_t)) = \sqrt{\frac{12\left[35-12\right]}{35}} \cong 2{,}8$$

La decisión acerca de expandir el nodo t, y contemplar el subárbol Tt se toma en base a la siguiente regla:

El nodo t se expande si y sólo si: $n'(T_t) + S(n'(T_t)) \prec n'(t)$

En el ejemplo:

$$12 + 2{,}8 \prec 15{,}5 = n'(t)$$

Por lo tanto el nodo nacionalidad se expande.

5.3.- Clustering (Agrupamiento)

5.3.1.- Algoritmo de agrupamiento: K-medias

Se usará el algoritmo K-medias para realizar las tareas de agrupamiento que debe realizar el sistema. Un algoritmo de agrupamiento es definido como una técnica diseñada para realizar una clasificación asignando patrones a grupos de tal forma que cada grupo sea más o menos homogéneo y distinto de los demás. Esta técnica podrá ser utilizada por el usuario para generar grupos de clientes con ciertos gustos y preferencias; para así por ejemplo dirigir una cierta publicidad a un grupo específico, reduciendo así los gastos en publicidad no efectiva.

5.3.1.1.- Funcionamiento del algoritmo

Este algoritmo requiere un único parámetro, K, el número de agrupamientos que debe encontrar. Puede plantearse en tres pasos:

1. Inicialización.

Consiste en inicializar arbitrariamente los centros de los K grupos.

2. Asignación y actualización de los centros.

En este paso se asigna cada patrón al grupo más cercano y se recalculan los centros en base a esta asignación.

3. ¿Convergencia?

En el paso anterior algunos patrones pueden cambiar de agrupamiento y en consecuencia, los centros de éstos. Si esto ocurre, se trata de repetir el paso 2 hasta que no se modifiquen los centros. Cuando no hayan modificaciones se considera que se ha encontrado una buena partición y se termina el agrupamiento.

EL algoritmo de agrupamiento se muestra en la Figura 15.

Algoritmo K-medias

Entrada: X un conjunto de patrones
K número de agrupamientos

Salida: devuelve: S1…Sn conjunto de patrones
Z1..Zn centros de los conjuntos

Método:

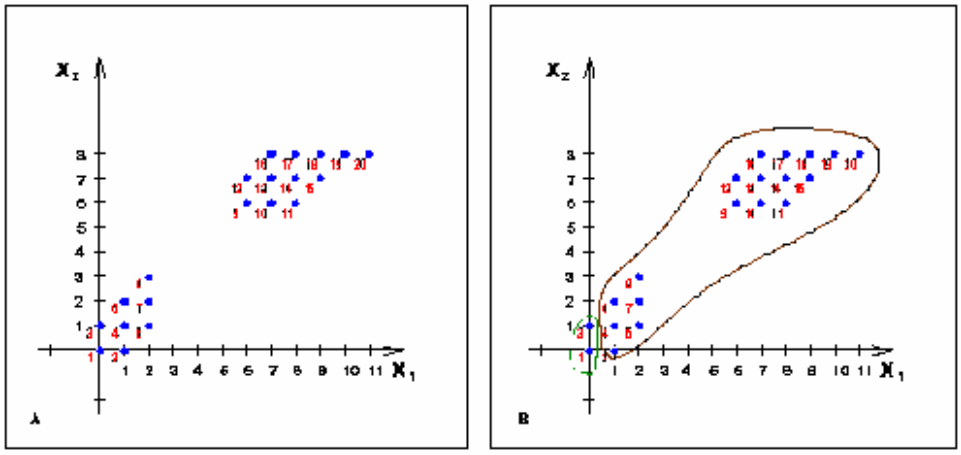
```
//paso 1inicializacion
T<- 0
Seleccionar Zi(t), i=1,2,3..,K

//paso 2 asignacion y actualizacion de centros
Para cada X =X1,X2,…,Xn
        m<- GrupoMasCercano(X)
        Sm(t)<-S(t-1)U{X}
Fin para

Para cada i=,,,K
        Zi(t)<-recalcular centro(i)
Fin para

//paso 3 convergencia?
Si Zi(t)=Zi(t-1),para i=1,2,...,K
        Terminar
Si no //Algún Zi(t-1) !=Zi(t)
        Ir al paso 2
```

Figura 15. Algoritmo K-medias

Como inconvenientes en la generación de los grupos, hay que destacar que el resultado final depende del valor de K, aunque el mayor problema está relacionado con la inicialización de los centros. Aunque los centros iniciales pueden seleccionarse aleatoriamente, si éstos se seleccionan de manera que se dispersen uniformemente sobre el espacio de representación, es más fácil que el resultado final sea mejor, además de acelerar la convergencia.

Ejemplo.

En la Figura 16 se muestra el conjunto de patrones sobre el que se aplicará el algoritmo de las *K-medias* con $K = 2$.



A) Situación inicial. B) Resultado de la primera asignación

Figura 16. Funcionamiento del algoritmo K-medias, primera asignación

Paso 1. Inicialización

Suponer que los centros se inicializan por el orden en que están los patrones en ***X***:

$$S_1(0) = \{X_1\} \quad Z_1(0) = (0, 0)$$

$$S_2(0) = \{X_2\} \quad Z_2(0) = (1, 0)$$

Paso 2. Asignación y actualización de centros

Si se asigna en base a la menor distancia entre $\delta(X, Z_1(0))$ y $\delta(X, Z_2(0))$ y finalmente se calcula $Z_1(1)$ y $Z_2(1)$

$$S_1(1) = \{X_1, X_3\} \qquad Z_1(1) = (0, 0.5)$$

$$S_2(1) = \{X_2, X_4, X_5, \ldots, X_{20}\} \quad Z_2(1) = (5.8, 5.3)$$

Paso 3. ¿Convergencia?

Como $Z_1(1) <> Z_1(0)$ y $Z_2(1) <> Z_2(0)$ se vuelve al paso 2.

Paso 2. Asignación y actualización de centros (2)

Si se asigna en base a la menor distancia entre $\delta(X, Z_1(1))$ y $\delta(X, Z_2(1))$ y finalmente se calcula $Z_1(2)$ y $Z_2(2)$

$$S_1(2) = \{X_1, X_2, ..., X_8\} \qquad Z_1(2) = (1.1, 1.3)$$

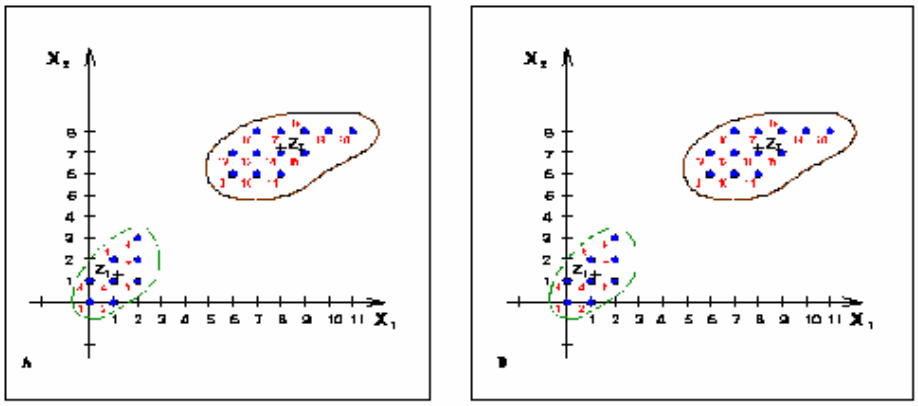
$$S_2(2) = \{X_9, X_{10}, ..., X_{20}\} \qquad Z_2(2) = (8.0, 7.2)$$

Paso 3. ¿Convergencia? (2)

Como $Z_1(2) <> Z_1(1)$ y $Z_2(2) <> Z_2(1)$ se vuelve al paso 2.

Paso 2. Asignación y actualización de centros (3)

Si se asigna en base a la menor distancia entre $\delta(X, Z_1(2))$ y $\delta(X, Z_2(2))$ y finalmente se calcula $\underline{Z_1}(3)$ y $Z_2(3)$. Lo anterior se muestra en la Figura 17.



A) Segunda asignación. B) Tercera (y última) asignación

Figura 17. Funcionamiento del algoritmo K-medias, segunda y tercera asignación

Paso 3. ¿Convergencia? (3)

Finalmente, como $Z_1(3) = Z_1(2)$ y $Z_2(3) = Z_2(2)$ se considera que la partición obtenida es estable y se termina.

6.- Diseño del sistema

Para abordar los pasos del proceso KDD, se ha tomado la decisión de separar el diseño del sistema en módulos, debido a que sus funciones son bastante distintas, y en general la información que se necesita en un módulo es la salida del anterior, se harán dos modelos de datos, uno con los atributos seleccionados, limpios y transformados para aplicar los algoritmos de Data Mining, y otro para almacenar las operaciones realizadas anteriormente por el usuario.

6.1.- Arquitectura del sistema

La arquitectura que aquí se verá, describe los elementos que se requieren para realizar el proceso de descubrimiento. El usuario interactúa con la aplicación a través de los elementos de interacción que ofrece la "vista" que es un mecanismo para visualizar los resultados actuales de un proceso de descubrimiento de conocimiento, y para que el usuario interactúe con éstos resultados la "vista" entrega las entradas del usuario al programa principal, éste encapsula los elementos ingresados y los entrega al módulo de minería. El programa principal también verifica la última actualización llevada a cabo a la base de datos del hotel desde Access. Para que si el usuario así lo desea, el módulo de limpieza extraiga los datos, los limpie, y se los entregue al módulo de trasformación el cual una vez que los haya transformado los guardará en la base de datos limpios y transformados. Una vez hecho todo esto el módulo de minería comienza a trabajar. Luego de haber realizado el proceso, el módulo de minería entrega los resultados al módulo generador de vista para que este muestre los resultados al usuario.

La Figura 18 presenta el diagrama que describe la arquitectura del sistema.

Arquitectura del Sistema

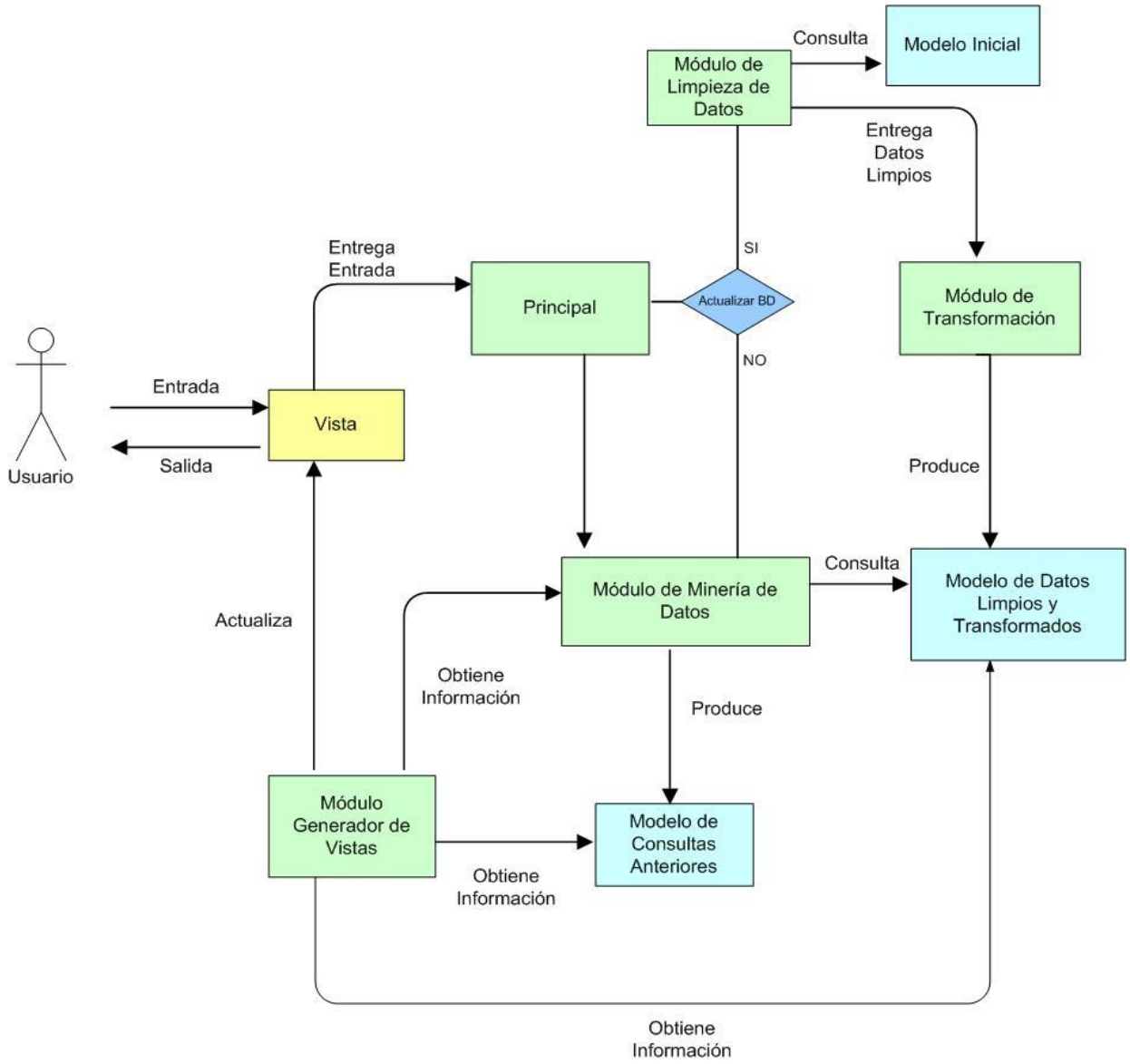


Figura 18. Arquitectura del sistema

6.1.2.- Módulos del sistema

6.1.2.1.- Módulo Principal

Este módulo recibe la entrada desde una vista de usuario, los encapsula y los entrega al Módulo de Minería de Datos. Este módulo al comienzo verificará si es que la base de datos "hotel" ha sido actualizada. Posteriormente si el usuario así lo desea, se notificará al Módulo de Limpieza de Datos para que haga su labor correspondiente. La entrada ingresada por el usuario corresponderá a la técnica de Minería de Datos a utilizar, los atributos objetivo desde dónde se extraerán los datos, y los parámetros propios del algoritmo. Este módulo se crea por la necesidad de tener una entidad que controle el orden de los procesos.

6.1.2.2.- Módulo de Limpieza de Datos

En este módulo se recuperan los datos desde la base de datos origen, se "limpian" los datos objetivo es decir eliminan los datos que puedan tener valores que dificulten o hagan menos confiable la operación de Data Mining (valores nulos, valores atípicos). Posteriormente los datos se pasan al módulo de transformación que se encargará de modificarlos para que los algoritmos puedan trabajar con ellos.

6.1.2.3.- Módulo de Transformación

En este módulo se aplican diferentes técnicas, con el fin de modificar los datos para que los algoritmos puedan trabajar con ellos, esto es discretizando los datos continuos y asignándole valores numéricos a datos nominales. Se mantendrán algunos datos duplicados de esta manera y se consultará uno u otro dependiendo de que tipo de dato necesite el algoritmo a utilizar.

6.1.2.4.- Módulo de Minería

Este módulo se encarga de realizar las operaciones solicitadas, las que recibirá conjuntamente con los atributos desde el Programa Principal. El módulo se encargará de extraer la información que necesite desde la "Base de Datos Limpios y Transformados" y ejecutará el algoritmo seleccionado (A priori, C4.5, K-medias), con los parámetros fijados por el usuario. Los resultados seran interpretados, y enviados al generador de vistas para que los muestre por pantalla. Los resultados obtenidos, también serán almacenados en la "Base de Datos de Consultas Anteriores", para que puedan ser recuperados posteriormente si el usuario así lo desea.

6.1.2.5.- Módulo Generador de Vistas

Este módulo se encarga de la interacción con el usuario. Será quien dé las opciones de selección al usuario, reciba las órdenes por parte de éste, y muestre los resultados. Este módulo se encargará de crear y manejar una serie de ventanas, en las cuales se le mostrarán al usuario las tareas que puede seleccionar, los atributos objetivo y los parámetros necesarios por el algoritmo. También mostrará los resultados de los algoritmos, estos pueden ser los obtenidos en el momento, o resultados anteriores que el usuario desee consultar.

6.2.- Diagramas del sistema

Los diagramas que se realizarán tienen como objetivo describir el sistema más detalladamente. La notación utilizada para los diagramas está descrita detalladamente en [Sch99].

6.2.1.- Diagramas de casos de uso

Los diagramas de casos de uso son utilizados para especificar los requerimientos del sistema desde el punto de vista del usuario. Cada caso de uso corresponde a un requerimiento del usuario sobre el sistema.

6.2.1.1.- Diagrama de casos de uso: *Realizar Minería de Datos*

El diagrama UML correspondiente al caso de uso general “Realizar Minería de Datos” se muestra a continuación en la Figura 19.

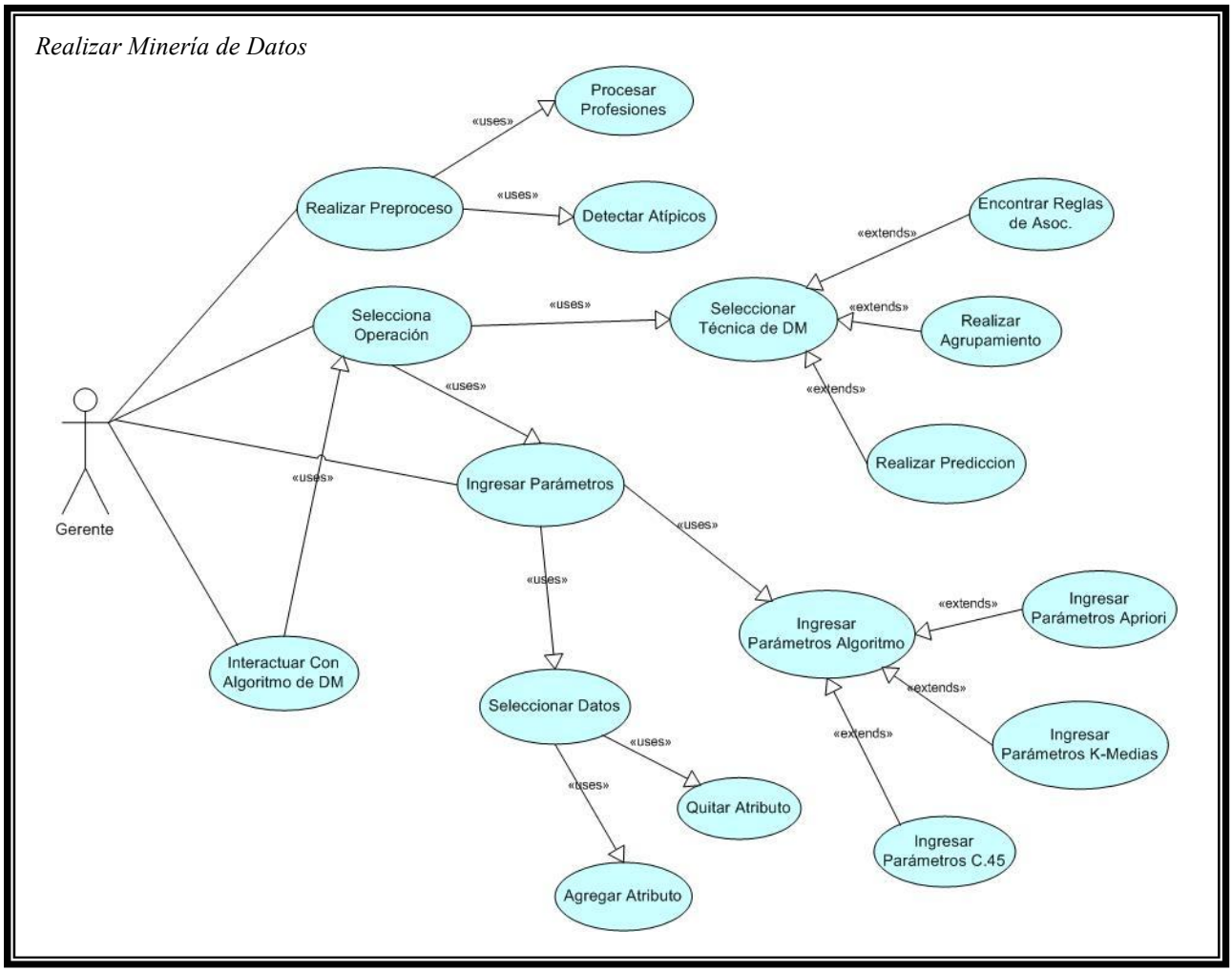


Figura 19. Caso de uso Realizar Minería de Datos

6.2.1.1.1.- Caso de uso narrativo: *Realizar Minería de Datos*

Caso de uso	Realizar Minería de Datos
Actor primario	Gerente
Propósito	Realizar una de las tres opciones de Minería de Datos disponibles en el sistema.
Resumen	Este caso de uso se inicia cuando el Gerente selecciona de entre las operaciones disponibles

	por el sistema, alguna que esté relacionada con Minería de Datos.
Escenario principal	1.- El Gerente realiza el preproceso de los datos a consultar. 2.- El Gerente selecciona la operación de Data Mining que desea realizar. 3.- El Gerente ingresa los parámetros que son necesarios como entrada para el algoritmo de Data Mining. 4.- El Gerente selecciona los atributos que van a contener los datos que van a ser utilizados por la minería 5.- El sistema valida las opciones ingresadas. 6.- El sistema envía las opciones ingresadas por el Gerente, al controlador, el cual se encarga de notificar al módulo de limpieza para que recupere y limpie los datos correspondientes a los atributos ingresados. 7.- El módulo de limpieza luego de limpiar los datos, los envía al módulo de transformación, para que les dé un formato adecuado para los algoritmos de Data Mining y los almacene en la Base de Datos Limpios y Transformados. 8.- El controlador luego de notificar al módulo de limpieza, entrega las opciones ingresadas al módulo de Minería de Datos. 9.- El módulo de Minería de Datos recupera los datos limpios y transformados de la BD y aplica la técnica correspondiente para obtener los resultados deseados. 10.-Los resultados son interpretados en un formato entendible por el Gerente, y mostrados por pantalla. 11.- El sistema da la opción de interactuar con el algoritmo de Minería de Datos, o bien regresar al

	menú principal.
Extensiones	2a.- El Gerente ingresa parámetros no válidos para el algoritmo de Data Mining a utilizar. 1) El sistema le pide que ingrese nuevamente el parámetro que fue mal ingresado. 5a.- El módulo de limpieza no pudo establecer conexión con la BD inicial. 1) Se envía mensaje de error en la conexión con la BD. 2) El sistema vuelve a la pantalla inicial. 8a.- El módulo de Minería de Datos no pudo establecer conexión con la Base de Datos Limpios y Transformados. 1) Se envía mensaje de error en la conexión con la BD. 2) El sistema vuelve a la pantalla inicial. 10a.- El Gerente desea interactuar con el algoritmo. 1) El sistema pide al Gerente que ingrese los parámetros necesarios para el algoritmo utilizado 10b.- El Gerente desea salir de la técnica de Minería de Datos utilizada. 1) El sistema vuelve al menú principal.

6.2.1.2.- Diagrama de casos de uso: *Ver Consultas Anteriores*

El segundo caso de uso identificado es “Ver Consultas Anteriores”, del cual se presenta el diagrama correspondiente en la Figura 20.

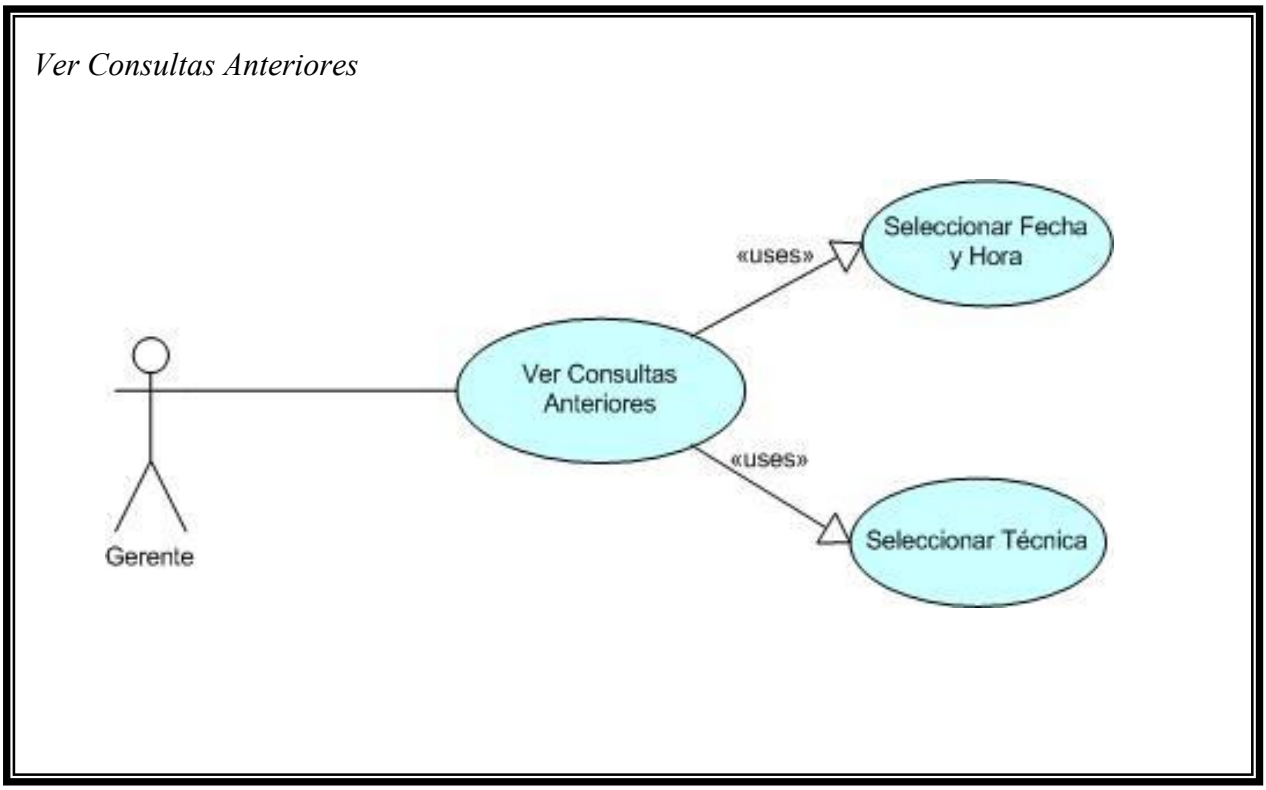


Figura 20. Caso de uso Ver Consultas Anteriores

6.2.1.2.1.- Caso de uso narrativo: *Ver Consultas Anteriores*

Caso de uso	Ver Consultas anteriores
Actor primario	Gerente
Propósito	Visualizar los resultados de las operaciones que se hayan realizado anteriormente.
Resumen	Este caso de uso se inicia cuando el gerente, elige en el menú principal la opción de Ver Consultas Anteriores. Esta opción le permite consultar operaciones de Minería de Datos realizadas anteriormente.
Escenario principal	1.- El sistema presenta un listado con las técnicas de Data Mining. 2.- El gerente selecciona una técnica a consultar.

	3.- El sistema presenta las fechas y horas de las consultas realizadas anteriormente para la técnica elegida. 4.-El gerente selecciona una fecha y hora y presiona el botón "Ir". 5.-El sistema muestra toda la información relativa a la consulta realizada. Ésta información corresponde al tipo de consulta, atributos seleccionados y respuestas obtenidas.
Extensiones	----------

6.2.2.- Diagrama de Clases Conceptual

En el modelo conceptual se descompone el problema en un conjunto de conceptos individuales, mostrando las relaciones entre dichos conceptos. Estos conceptos son las denominadas clases, que en definitiva serán los componentes del sistema final.
En la Figura 21 se presenta el diagrama de clases conceptual del sistema en desarrollo.

6.2.3.- Diagrama de Estados

Un diagrama de estados es una manera para caracterizar un cambio en un sistema, es decir que los objetos que lo componen modificaron su estado como respuestas a los sucesos y al tiempo [Sch99]. En la Figura 22 se presenta el diagrama de estados relacionado con las diferentes funcionalidades del sistema.

6.2.4.- Diagrama de Secuencias

El diagrama de secuencias consta de objetos y una línea de tiempo que representa su duración (tiempo de vida); este diagrama muestra la interacción de los objetos en el tiempo. En la Figura 23 se presenta la interacción de los objetos del sistema mediante el diagrama de secuencias general, con éste es posible representar la interacción de los tres tipos de análisis realizados, debido a la similaridad de éstos.

Diagrama de Clases Conceptual

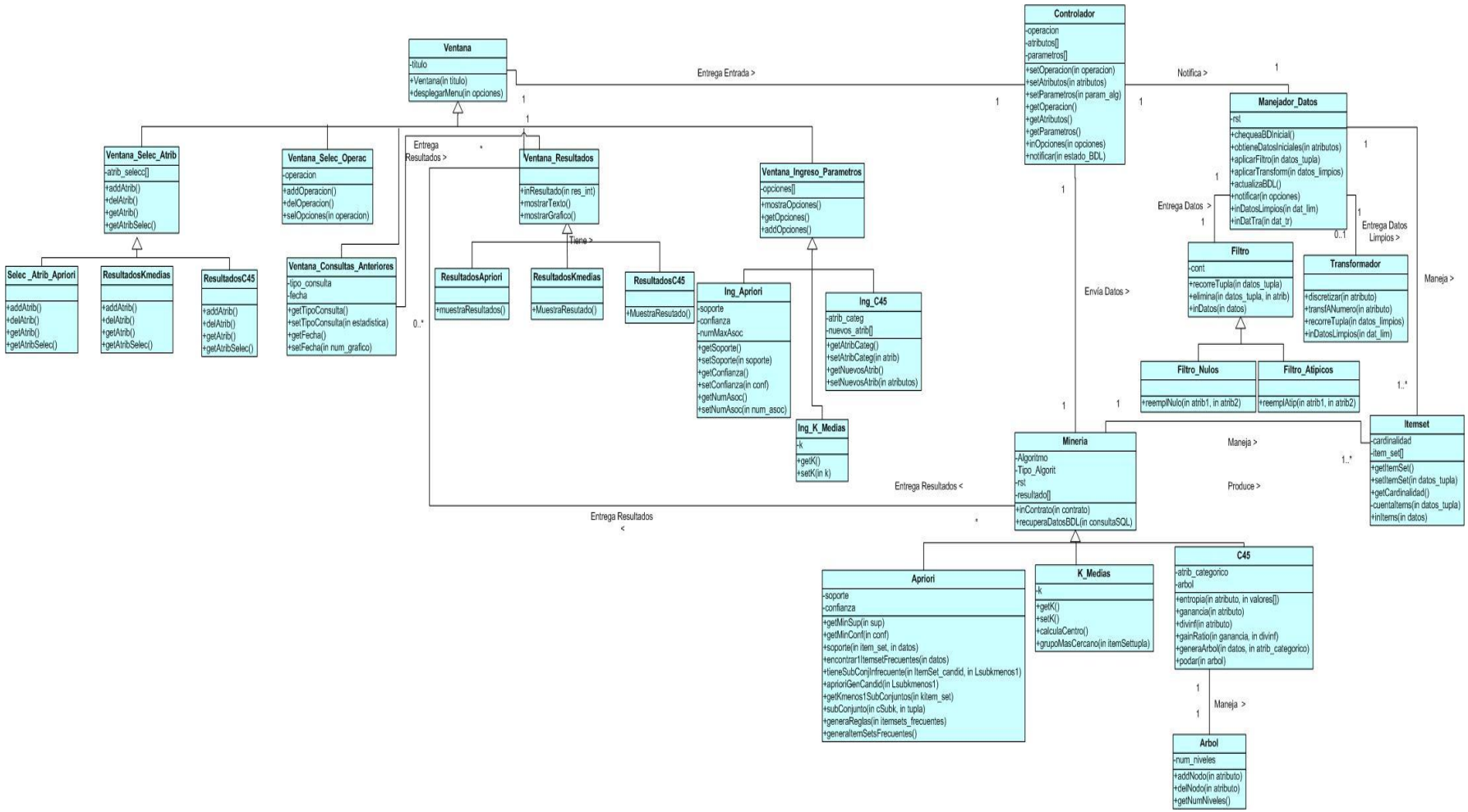


Figura 21. Diagrama de Clases Conceptual

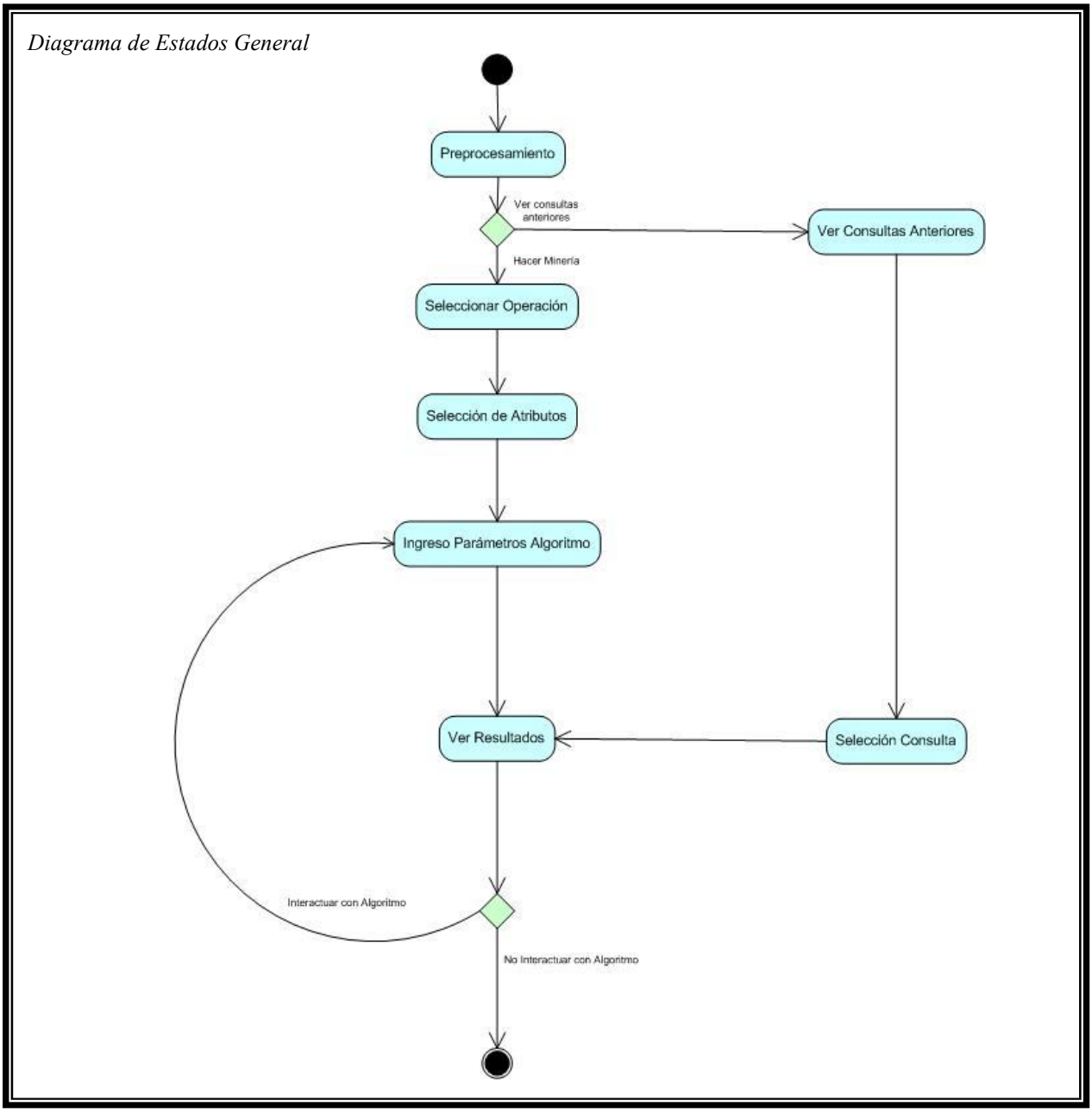


Figura 22. Diagrama de Estados General

Diagrama de Estados Realizar Preproceso

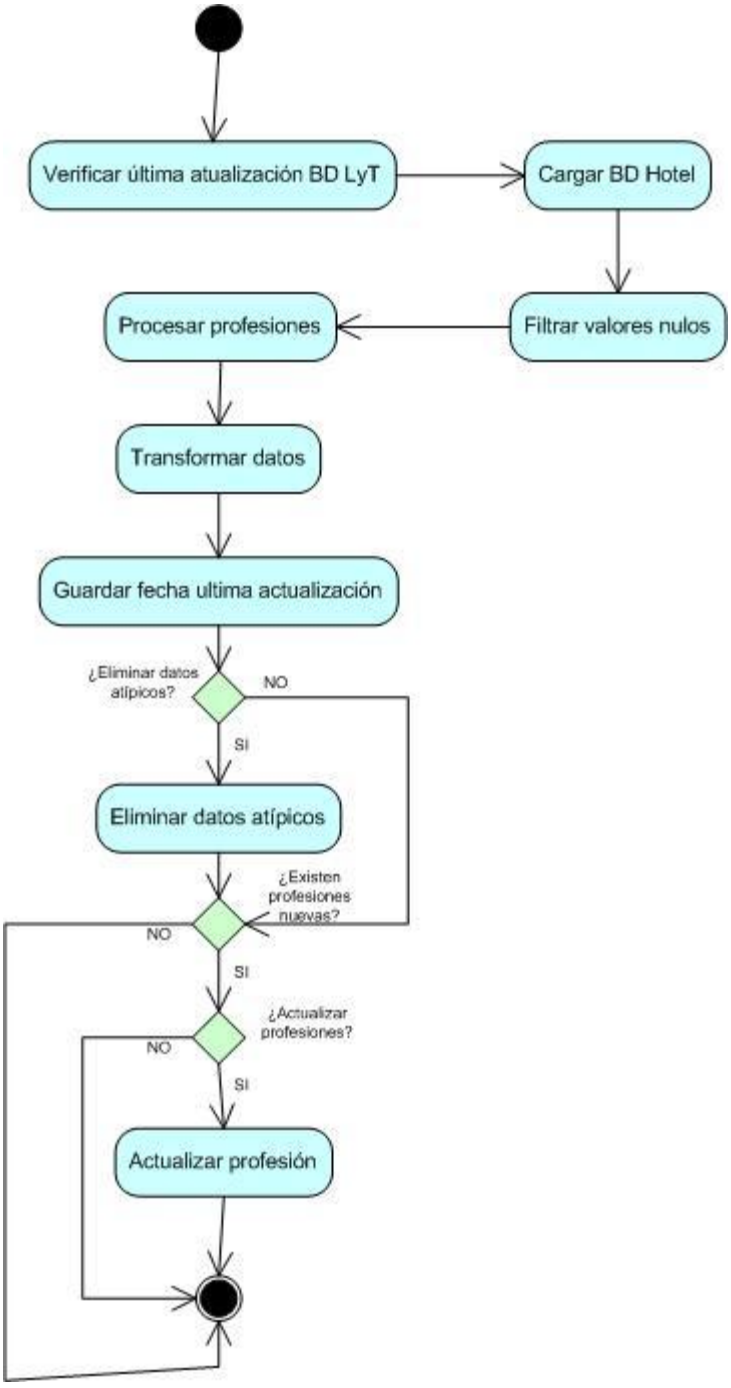


Figura 22.1. Diagrama de Estados Realizar Preproceso

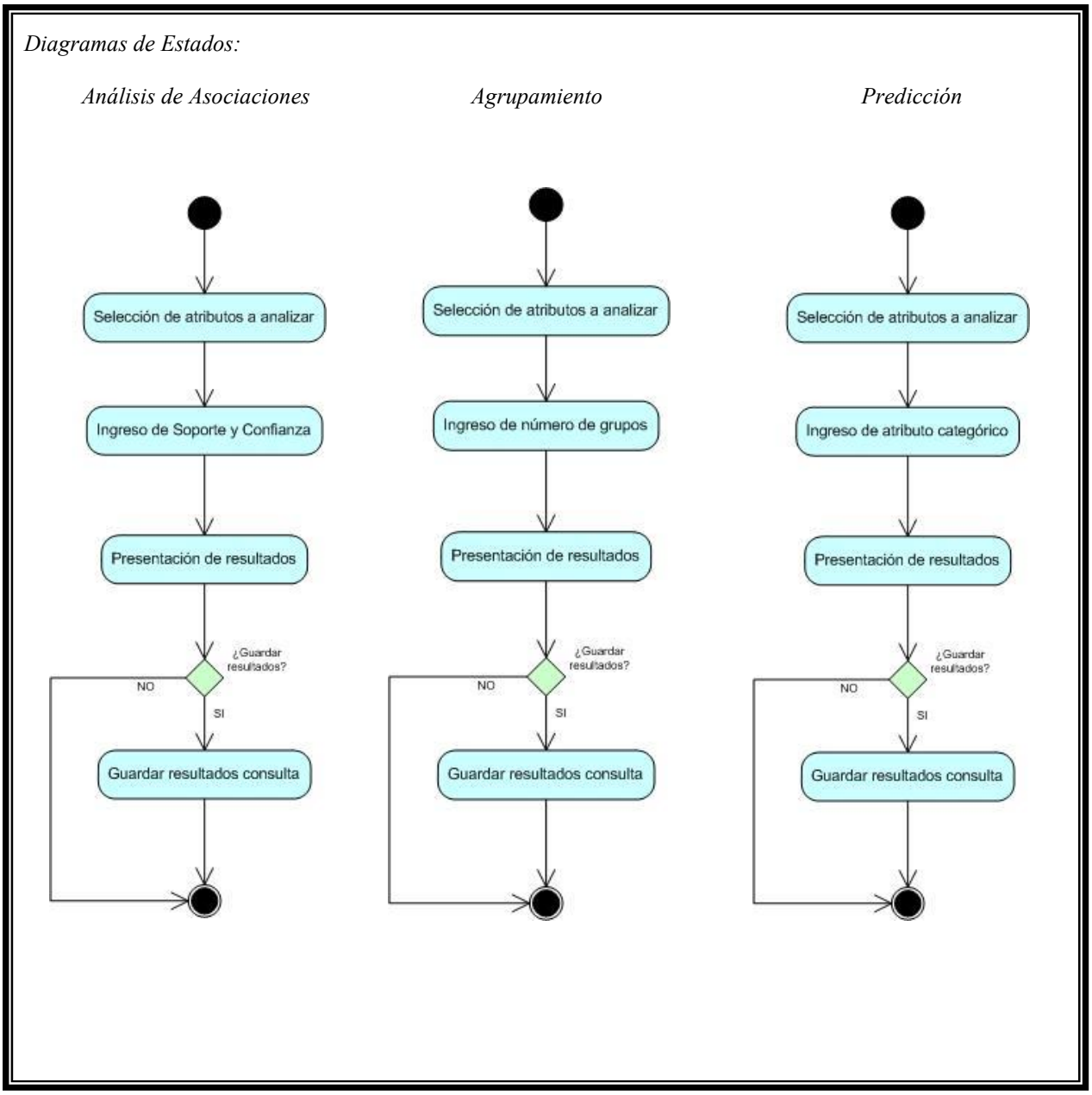


Figura 22.2. Diagramas de Estados Análisis de Asociaciones, Agrupamiento y Predicción

Diagrama de Estados Ver Consultas Anteriores

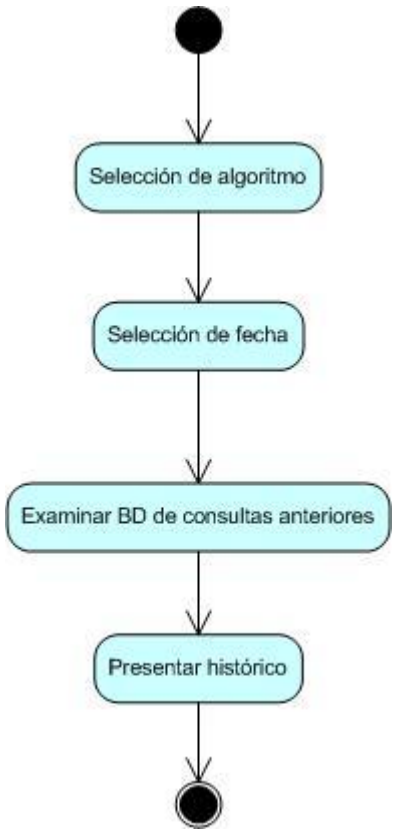


Figura 22.3. Diagramas de Estados Ver Consultas Anteriores

Diagrama de Secuencias General

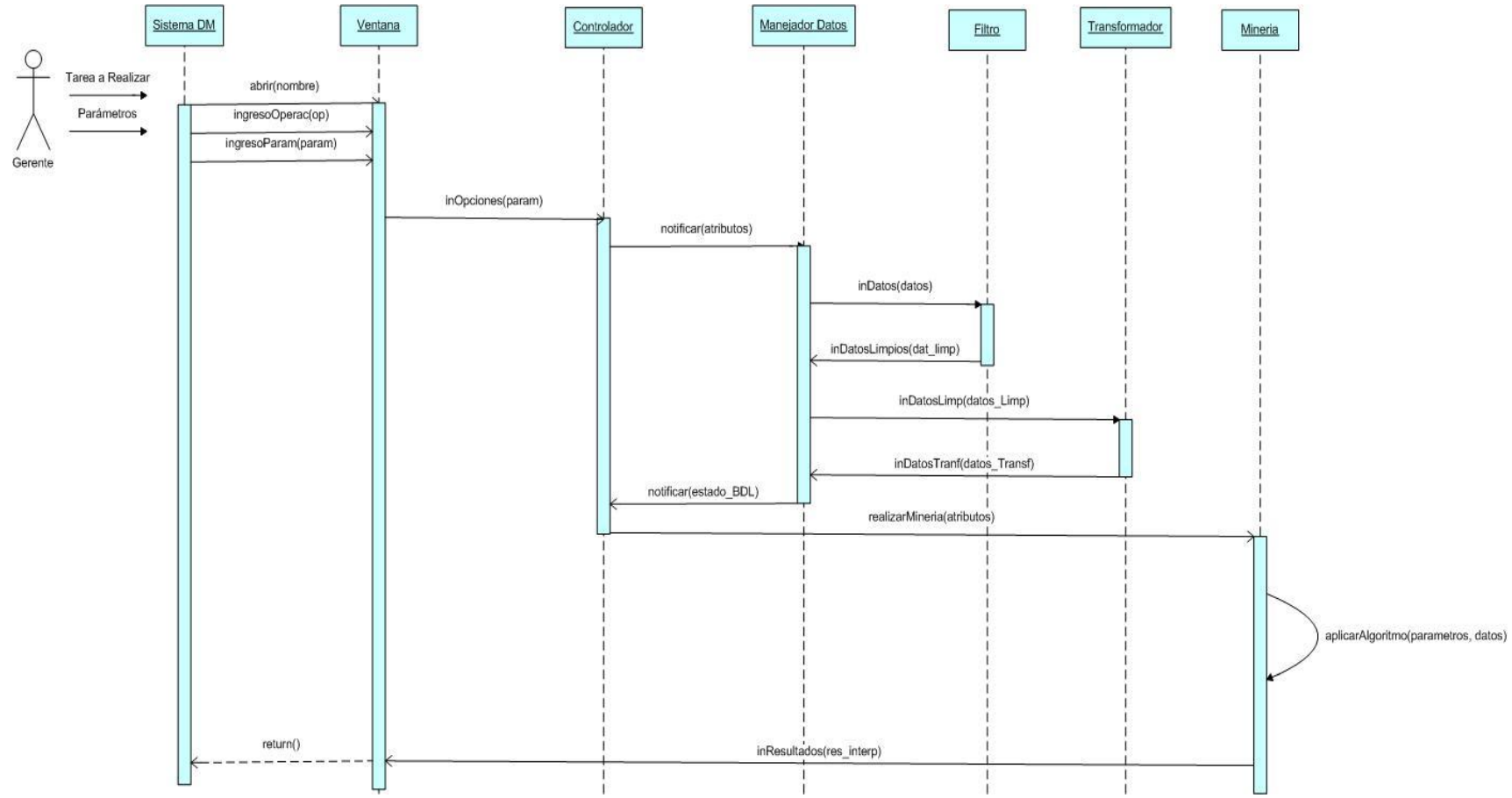


Figura 23. Diagrama de Secuencias

6.2.5.- Diagrama de Colaboración

El diagrama de colaboración muestra la forma en que los objetos colaboran entre sí, tal como sucede con un diagrama de secuencias. Este tipo de diagramas muestra los objetos junto con los mensajes que se envían entre ellos. Su diferencia está dada porque el diagrama de secuencias se organiza de acuerdo al tiempo, y el de colaboración de acuerdo al espacio [Sch99]. A continuación en la Figura 24 se muestra la colaboración de los objetos mediante el diagrama antes descrito.

Diagrama de Colaboración

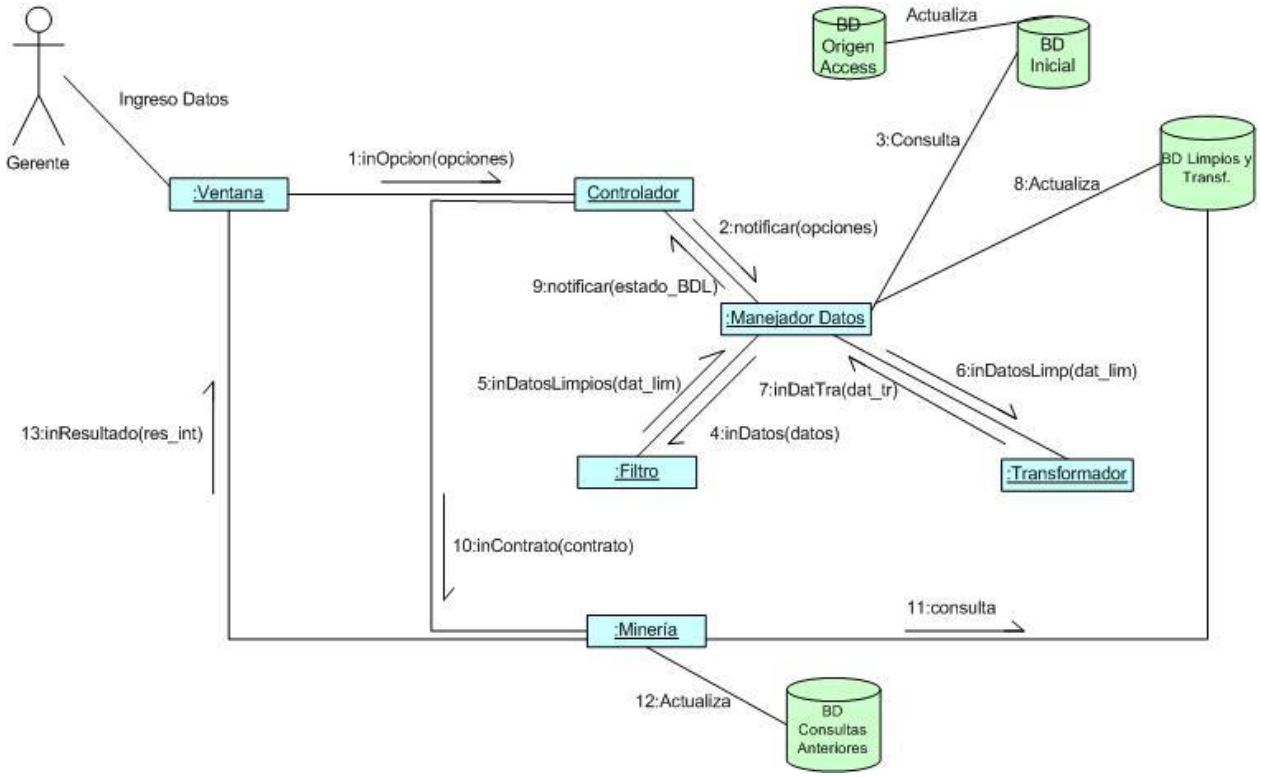


Figura 24. Diagrama de Colaboración

6.2.6.- Modelo de Datos Limpios y Transformados

El Modelo de Datos Limpios y Transformados corresponde a la base de datos que será consultada por el Módulo de Minería para obtener los datos que serán utilizados como entrada para los algoritmos de Data Mining que se ocuparán en la consulta realizada por el usuario. Este modelo es poblado obteniendo datos desde la base de datos operacional, a los cuales se le aplicarán técnicas de limpieza de datos tales como eliminación de valores atípicos, y tratamiento de valores nulos, para posteriormente ser aplicadas transformaciones de los mismos con el fin de preparar los datos para la aplicación de la técnica de Data Mining deseada. En la Figura 25 se presenta el Modelo de Datos Limpios y Transformados.

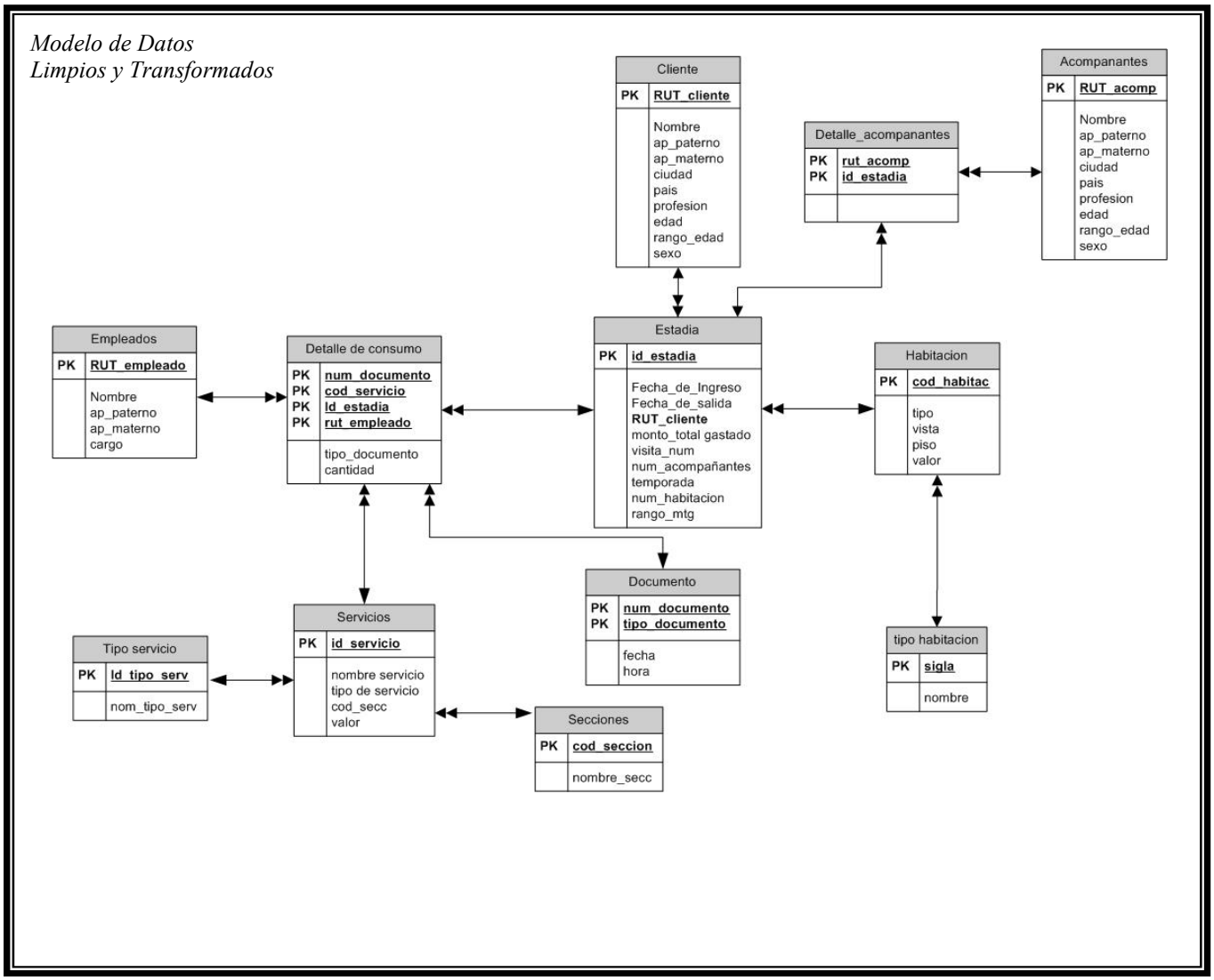


Figura 25. Modelo de Datos Limpios y Transformados

6.2.7.- Modelo de Consultas Anteriores

En este modelo de datos se almacenarán los resultados de consultas hechas anteriormente, con el objetivo de visualizar las variaciones en el comportamiento de los clientes durante el tiempo. La Figura 26 muestra el modelo de Consultas Anteriores.

Modelo de Consultas Anteriores

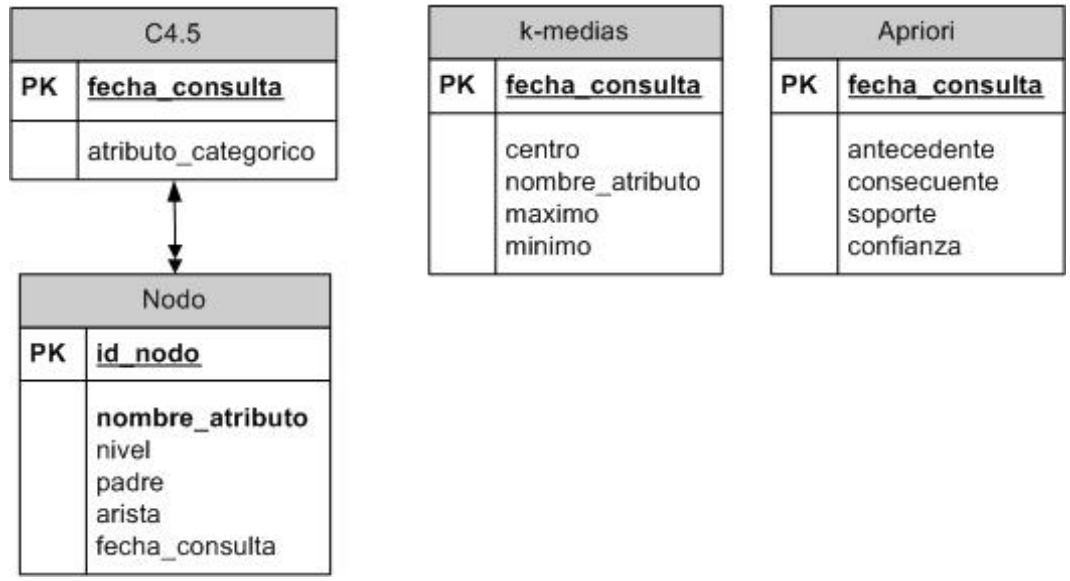


Figura 26. Modelo de Consultas Anteriores

6.3.- Selección de los datos

Esta es la primera etapa del proceso de descubrimiento de conocimiento (KDD). Aquí se ha examinado la base de datos del Hotel con el fin de determinar cuales son los atributos más relevantes al momento de realizar un análisis del comportamiento del cliente.

El modelo obtenido de la base de datos del Hotel, desde la cual se seleccionarán los datos objetivo se presenta a continuación en la Figura 27.

Base de Datos del Hotel

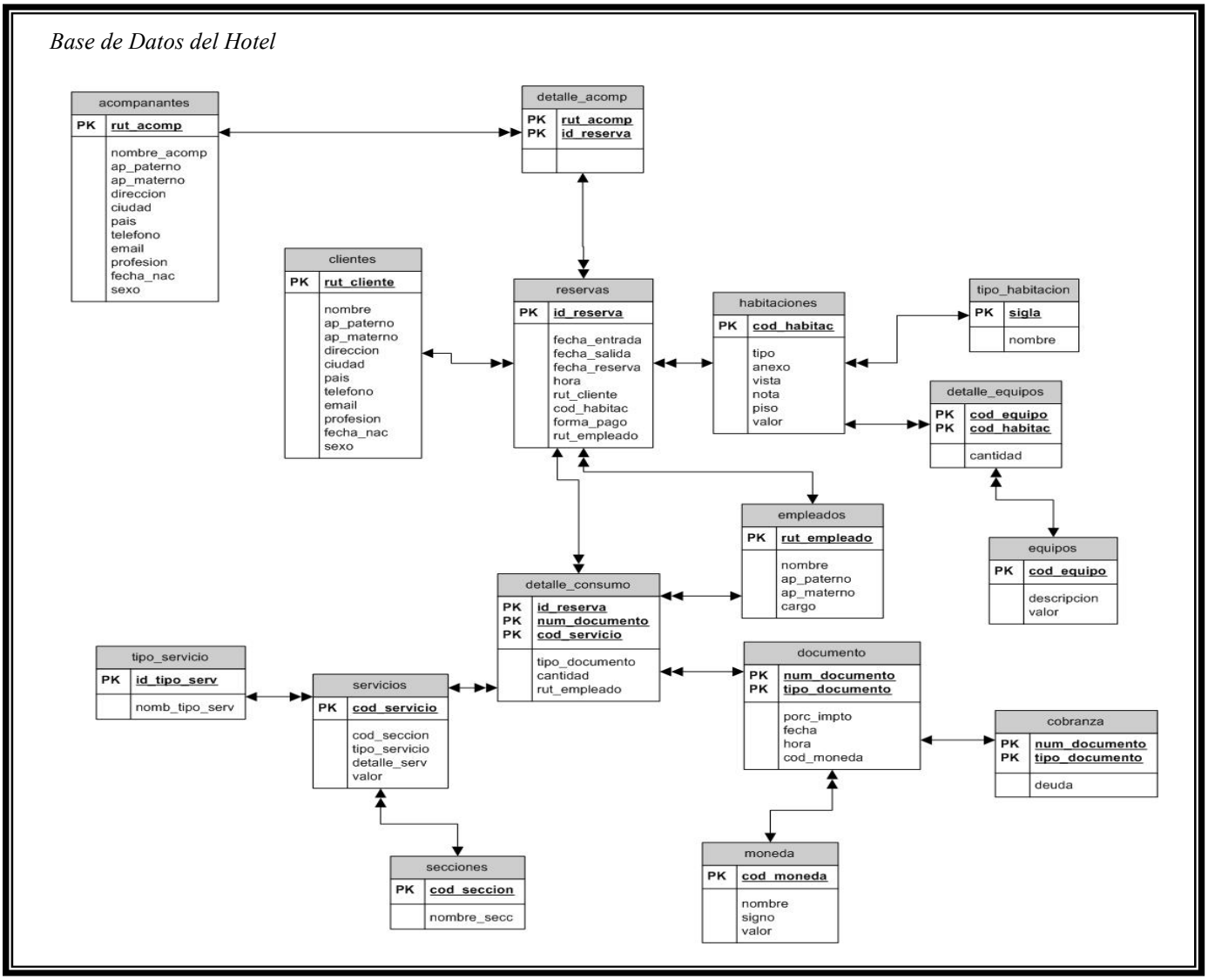


Figura 27. Base de Datos Inicial

Del modelo presentado es necesario centrarse en las tablas y atributos que serán relevantes para el análisis.
En la Figura 28 aparece un listado de las tablas seleccionadas del modelo inicial, y las tablas del modelo de Datos Limpios y Transformados que son pobladas con dicha información.

Tabla Modelo Inicial	Tabla Modelo Datos Limpios y Transformados
clientes	*clientes*
acompanantes	*estadia*
reservas	*estadia*
habitaciones	*habitacion*
tipo_habitación	*tipo_habitacion*
empleados	*personal*
detalle_consumo	*detalle_consumo*
servicios	*servicios*
tipo_servicio	*tipo_servicio*
secciones	*secciones*
acompanantes	*acompanantes*
documento	*documento*
empleados	*empleados*

Figura 28. Tablas del modelo inicial y su correspondencia con el modelo de datos limpios y transformados

6.4.- Limpieza de datos

Esta es la segunda etapa del desarrollo del sistema, en esta se construirá el módulo de limpieza de datos, el cual se ocupa de extraer los datos de la base da datos del hotel, "limpiar" los valores nulos y atípicos que puedan influir negativamente en el proceso de descubrimiento de conocimiento. Luego estos datos deberán pasar por el módulo de Transformación antes de estar listos para el trabajo de minería.

6.4.1.- Valores nulos

Existen diferentes métodos para tratar estos valores:

Se analizarán los atributos en la base de datos original que permiten valores nulos y se tratarán de una u otra manera dependiendo de cada caso.

Tabla clientes

Nombre: si el valor de este campo es nulo, no será relevante pues este valor no es considerado para realizar operaciones de Data Mining.

ap_paterno: si el valor de este campo es nulo, no será relevante pues este valor no es considerado para realizar operaciones de Data Mining.

ap_materno: si el valor de este campo es nulo, no será relevante pues este valor no es considerado para realizar operaciones de Data Mining.

direccion: si el valor de este campo es nulo, no será relevante pues este valor no es considerado para realizar operaciones de Data Mining.

ciudad: si el valor de este campo es nulo, se eliminará la tupla.

pais: si el valor de este campo es nulo, se eliminará la tupla.

telefono: si el valor de este campo es nulo, no será relevante pues este valor no es considerado para realizar operaciones de Data Mining

email: si el valor de este campo es nulo, no será relevante pues este valor no es considerado para realizar operaciones de Data Mining

profesion: si el valor de este campo es nulo, se eliminará la tupla.

fecha_nac: si el valor de este campo es nulo, se eliminará la tupla.

Tabla Acompanantes

Nombre: si el valor de este campo es nulo, no será relevante pues este valor no es considerado para realizar operaciones de Data Mining.

ap_paterno: si el valor de este campo es nulo, no será relevante pues este valor no es considerado para realizar operaciones de Data Mining.

ap_materno: si el valor de este campo es nulo, no será relevante pues este valor no es considerado para realizar operaciones de Data Mining.

direccion: si el valor de este campo es nulo, no será relevante pues este valor no es considerado para realizar operaciones de Data Mining.

ciudad: si el valor de este campo es nulo, se eliminará la tupla.

pais: si el valor de este campo es nulo, se eliminará la tupla.

telefono: si el valor de este campo es nulo, no será relevante pues este valor no es considerado para realizar operaciones de Data Mining

email: si el valor de este campo es nulo, no será relevante pues este valor no es considerado para realizar operaciones de Data Mining

profesion: si el valor de este campo es nulo, se eliminará la tupla.

6.4.2.- Valores atípicos

Antes de procesar los datos, es necesario asegurar que no existan errores en éstos, pues si existiesen, podrían afectar la veracidad de las respuestas entregadas por el sistema. Es por esto que se deben analizar los datos en busca de posibles errores en el ingreso. Estos errores se manifiestan como datos "atípicos", es decir, que sus valores son anormalmente extremos y por lo tanto están muy alejados del resto de los datos.

Para el tratamiento de los datos atípicos se usa la técnica del ***Rango Intercuartílico***. Para tal efecto se necesita conocer la medida estadística Cuartil (Q1, Q2, Q3).

Se calcula el recorrido intercuartilico IQR = Q3 – Q1
Entre estos dos cuartiles se encuentra el 50% de los datos más centrales.

Vallas interiores.

Valla interna inferior: VII = Q1 – 1.5*IQR
Valla interna superior: VIS = Q2 – 1.5*IQR

Los datos que caen fuera del intervalo [VII,VIS] se consideraran valores atípicos.
Lo antes mencionado, puede ser representado mediante el diagrama de caja y bigotes de la Figura 29 [Web-4].

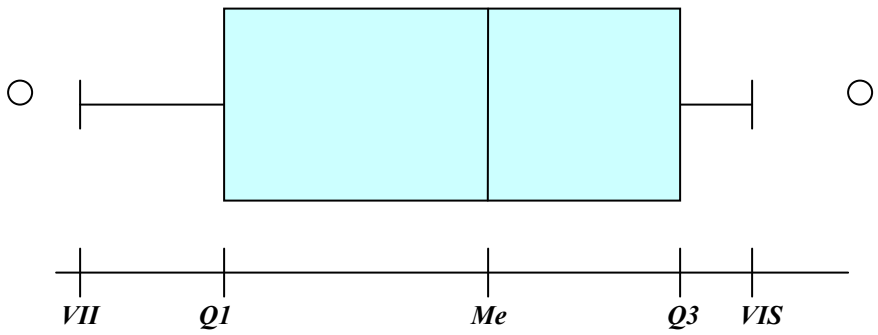


Figura 29. Diagrama de caja y bigotes

Este algoritmo es aplicado por el módulo de Limpieza a los atributos cuyos valores sean numéricos y que sean relevantes para el análisis, en este caso el atributo "cantidad" en la tabla **detalle_consumo,** que corresponde a la cantidad de servicios consumidos del mismo tipo.

6.5.- Transformación de los datos

En la etapa de transformación y reducción de los datos se realizan las siguientes labores para lograr tales objetivos.

Tabla Clientes.

En esta tabla se encuentran atributos que necesitan ser transformados para poder aplicar los algoritmos de Data Mining.

Para efectos del análisis, es necesario conocer la edad del cliente la cual se obtiene a partir de la fecha de nacimiento ingresada en el atributo "fecha_nac". A la edad obtenida se le asigna un intervalo de 10 años que corresponde al rango de la edad, almacenando así en la tabla clientes de la base de datos Limpios y Transformados estos dos atributos (edad, rango_edad).

Existe un caso especial en relación al atributo "profesion" de la tabla clientes, puesto que el ingreso de la profesión no está controlado en el sistema hotelero en utilización, pudiendo ingresarse distintas entradas para una misma profesión. Por ejemplo una de las entradas típicas detectadas está en relación al sexo, por ejemplo: ABOGADO y ABOGADA deben referenciar a la misma profesión, la cual sería ABOGADO. Entradas como éstas, como así también letras faltantes en profesiones mal ingresadas deben ser solucionadas de alguna manera. Para tal efecto se crean tres tablas: profesiones, sinonimos, y newprof.

- ***Tabla profesiones:*** en esta tabla se cargarán inicialmente las profesiones válidas.
- ***Tabla sinonimos:*** en esta tabla se almacenarán las profesiones que se corresponden con profesiones válidas de la tabla "profesiones", mediante el id de profesión.
- ***Tabla newprof:*** aquí se almacenarán las nuevas entradas no clasificadas en ninguna de las tablas anteriores.

El sistema al momento de filtrar la profesión del cliente, verifica que ésta exista en la tabla profesiones. Si la profesión no está considerada en la tabla profesiones, se consulta la tabla sinónimos para ver si está contemplada como sinónimo de una profesión válida, si éste es el caso, se toma el id de la profesión a la que corresponde dentro de la tabla profesiones, y es considerada la profesión correspondiente.

Si la profesión filtrada no está considerada en ninguna de las tablas anteriores, es almacenada en la tabla "newprof", para ser mostrada al usuario como una profesión nueva.

Tabla Acompañantes.

Se realiza el mismo procedimiento que en la tabla clientes.

Tabla Reservas.

En esta tabla se encuentran datos que también son importantes, y que deben ser tratados para poder trabajar con ellos.

Atributos Necesarios en la base de datos Limpios y Transformados:

dias_estadia: este atributo consiste en la cantidad de días que el cliente alojó en el hotel. Es obtenido mediante los atributos: fecha_entrada y fecha_salida de la tabla "reservas" de la base de datos "hotel".
monto_tg: este atributo se refiere a la cantidad total de dinero gastada en servicios durante la estadía del cliente en el hotel. Se obtiene mediante consultas sql sobre la tabla "detalle_consumo" de la base de datos "hotel".
rango_mtg: atributo obtenido asignando un rango de $100.000 al valor dado por el atributo monto_tg visto anteriormente.
num_visita: cantidad de visitas al hotel realizadas por el cliente. Obtenida por la cantidad de reservas realizadas.
num_acompanantes: cantidad de acompañantes que el cliente tuvo durante su estadía. Está dada por la cantidad de registros en la tabla "detalle_acompanantes" correspondiente a la reserva.
temporada: atributo que contiene la temporada de la reserva, puede ser: alta, media o baja. Es obtenida por la fecha de la reserva.

Los datos de las tablas que no son tratadas en la transformación de datos, son traspasados directamente mediante paquetes DTS, entre las bases de datos "hotel" y "lyt" (limpios y transformados).

7.- Construcción del sistema

En esta etapa del proceso de desarrollo de software, se alcanzará la capacidad operacional del producto. Se logrará este objetivo a través de sucesivas iteraciones e incrementos, con lo cual se obtendrá un producto de software listo para operar.

El sistema está dividido en módulos que representan el modo de trabajar del modelo KDD. Este modelo requiere que la primera actividad a realizar en la etapa de desarrollo del sistema sea la etapa de selección de datos, de la cual a continuación se presentan las labores realizadas para cumplir con tal objetivo.

7.1.- Recuperación de los datos desde base de datos Access a SQL Server 7

Para poder trabajar con los datos en forma eficiente así como contar con herramientas que dan más posibilidades al desarrollo del proyecto, los datos almacenados inicialmente por el Sistema Hotelero en bases de datos Access serán exportados a tablas de base de datos SQL Server. La selección de los datos a recuperar está dada por lo especificado en el diseño.

7.1.2.- Herramienta para el traspaso de datos: DTS

El servicio DTS de SQL Server 7.0 permite la transferencia de datos entre fuentes de datos relacionales y no relacionales. El asistente DTS permite seleccionar una fuente origen y una fuente final para los datos, y el tipo de conversiones que desea aplicar a cada columna. En el Anexo B se presenta la secuencia de pasos llevadas a cabo para tales efectos.

7.2.- Implementación de los algoritmos de Data Mining

7.2.1.- Algoritmo A priori

Este algoritmo fue utilizado para encontrar reglas de asociación entre los servicios que presta el hotel.

La idea de este algoritmo es encontrar todos los subconjuntos de servicios existentes, y al mismo tiempo examinar cuales de estos subconjuntos fueron utilizados por clientes, si el número de consumos de un subconjunto supera el soporte mínimo, se considerará este subconjunto para la creación de una regla.

Es lógico que esta idea no se pueda implementar tratando de generar todos los subconjuntos, debido a que ningún computador podría procesar tal cantidad de datos.

La cantidad de subconjuntos está dada por:

$$\sum_{k=1}^{n} C_n^k$$

Donde n es la cantidad de servicios ofrecidos por el hotel. En el proyecto se considera n=184 servicios. Por lo que la cantidad de subconjuntos sería:

$$\sum_{k=1}^{184} C_{184}^k = C_{184}^1 + C_{184}^2 + ... + C_{184}^{184} = 2{,}45199\text{E} + 55$$

Tomando en cuenta que el análisis de un subconjunto, en un computador relativamente potente, tomaría a lo menos: 0,000000001 segundos (cosa que difícilmente ocurriría), el tiempo de generación de todos los subconjuntos estaría dado por:

2,45199E+55*0,000000001/60/60/24/365=7,77522E+38 años

El ejemplo mostrado anteriormente deja en claro que es prácticamente imposible implementar la solución generando todos los subconjuntos de servicios en un tiempo de respuesta razonable.

Es por esto que el algoritmo propone generar los subconjuntos partiendo desde los subconjuntos con menos elementos (que tienen más probabilidad de superar el soporte), y luego buscar los subconjuntos más grandes considerando el conocimiento "a priori" de los subconjuntos que ya han sido generados.

Sin embargo para utilizar este conocimiento a priori al momento de implementar la solución, es necesario realizar varios procesos, entre los cuales están el descomponer cada conjunto creado en todos sus subconjuntos con cardinalidad inferior, y hacer comparaciones de estos subconjuntos, para verificar si estos superan o no el soporte mínimo. Estos procesos se deben hacer por cada subconjunto que se genere y consumen bastante tiempo, por lo tanto se ha propuesto realizar algunas optimizaciones al algoritmo con el fin de disminuir el tiempo de respuesta del sistema.

7.2.1.1.- Implementación propuesta

Para implementar el algoritmo A priori, se crea primero una matriz de transacciones donde cada fila es una transacción y cada columna un servicio que proporciona el hotel, para esto se consideró como una transacción los consumos incluidos en un documento (boleta, factura). Esta matriz de transacciones estará compuesta de ceros y unos que indicaran la presencia o ausencia de un servicio en una transacción.

Para crear los subconjuntos de servicios se creó una estructura de árbol en que cada nodo es un servicio y sus hijos son todos los demás servicios con los cuales pude formar subconjuntos nuevos.

A modo de ejemplo, suponiendo un conjunto de 3 servicios se construiría el árbol de la Figura 30.

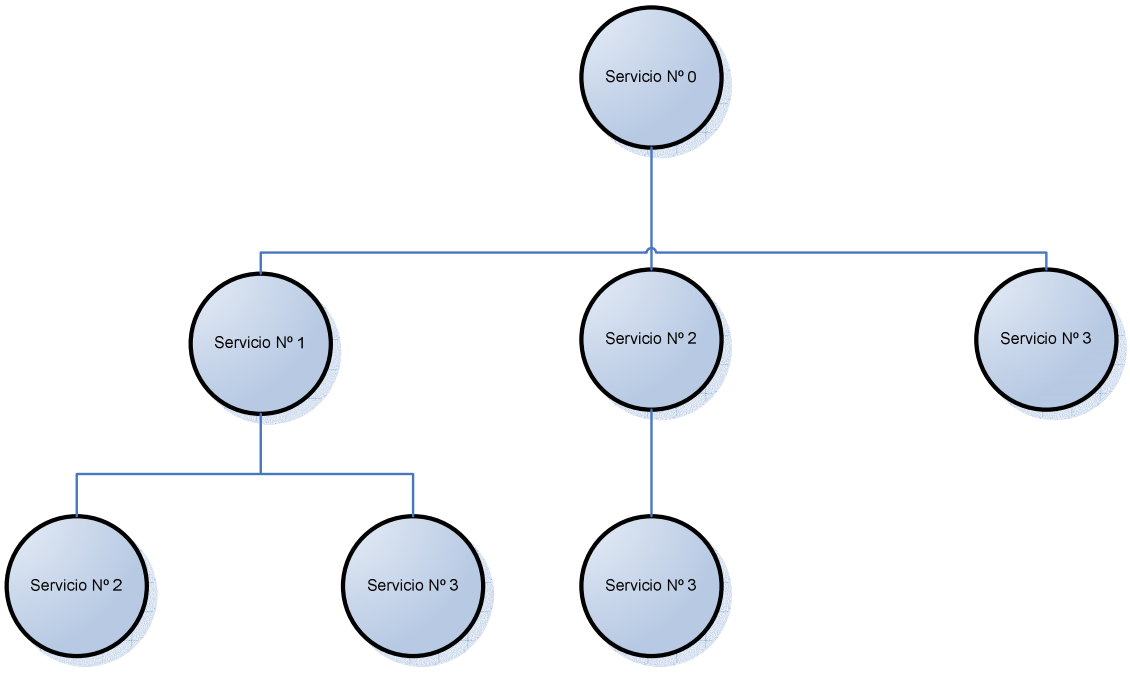


Figura 30. Árbol de servicios

El servicio Nº 0 es un servicio virtual que, se crea con el objetivo de que sea el padre de todos los otros servicios.

Al final se tendrá un árbol desde el cual se podrán obtener todos los subconjuntos de servicios fácilmente. Sin embargo es necesario decidir si es conveniente añadir un nodo al árbol o no, pues si existe una gran cantidad de servicios el árbol podría crecer demasiado ocupando toda la memoria del equipo.

Para decidir si conviene añadir un nodo al árbol, es necesario calcular si el subconjunto que se generará añadiendo ese nodo, superará el soporte mínimo, para esto se debe contar la cantidad de ocurrencias en el total de transacciones del subconjunto que se generará. Contar cuantas veces se encuentra un subconjunto en la matriz de transacciones requiere muchas comparaciones (cantidad de filas de la matriz * la cardinalidad del subconjunto), por este motivo, en este paso es necesario realizar una optimización.

Para hacer esta optimización se decidió agregar a cada nodo del árbol una lista enlazada con los índices de las filas de la matriz de transacciones donde se encontró el subconjunto compuesto por el nodo y todos sus predecesores. De esta manera, para decidir si corresponde agregar un nodo, se debe buscar la ocurrencia del servicio relativo al nodo a agregar, en los índices contenidos en la lista enlazada que posee el nodo padre. Si la cantidad de veces que el servicio está contenido en las filas de la matriz, dadas estas por los índices almacenados en la lista enlazada del nodo padre, supera el umbral de soporte mínimo, el servicio será agregado al árbol en el nodo correspondiente.

7.2.2.- Algoritmo K-medias

Este algoritmo fue utilizado para encontrar grupos de clientes dentro del hotel con características mas o menos homogéneas entre si, y distintos unos de otros.

7.2.2.1.- Implementación propuesta

Para efectos de implementar el algoritmo K-medias, los atributos numéricos elegidos por el usuario son consultados a la base de datos y guardados en una matriz. De esta matriz se eligen al azar tantas filas como grupos haya que formar (parámetro "k" ingresado por el usuario) las cuales corresponden a los centros iniciales provisorios de los grupos a formar; estos centros son almacenados en un vector. Para almacenar los grupos también se utiliza un vector que contiene k elementos correspondientes a cada grupo. Estos elementos a su vez consisten en listas enlazadas que contienen los índices de las filas de la matriz, que corresponden a los registros pertenecientes al grupo.

Para asignar las filas de la matriz a un grupo, se considera la distancia de cada una de éstas con los centros definidos anteriormente. La asignación se realiza de acuerdo a las distancias mínimas entre las filas y cada centro de grupo.
A continuación se procede a recalcular los centros mediante el promedio de las filas contenidas en cada grupo. Estando ya calculados los nuevos centros de grupos, se procede a realizar nuevamente la asignación de filas a cada grupo según las distancias mínimas, y se vuelven a calcular los centros. Este procedimiento se repite hasta que los centros recalculados coinciden con los calculados anteriormente.

Ejemplo.

Cantidad de grupos: 2

índice	edad	Monto gastado	Días estadía
1	23	54.000	5
2	28	62.000	2
3	41	114.000	12
4	52	105.000	21

Centro 1: (23,54.000, 5)
Centro 2: (28,62.000, 2) } Centros elegidos al azar

Grupo 1: 1
Grupo 2: 2, 3,4

En la segunda iteración:

Centro 1: (23,54.000, 5)
Centro 2: (40,3; 93666,7; 11,7)

Grupo 1: 1, 2
Grupo 2: 3,4

Después de la siguiente iteración los centros coinciden, por lo que finaliza el algoritmo.

7.2.3.- Algoritmo C4.5

Este algoritmo es utilizado para encontrar modelos predictivos del comportamiento de los clientes.

El algoritmo, construye un árbol de decisión de manera descendente comenzando por preguntarse: ¿Que atributo debería ser colocado en la raíz del árbol? Para responder esta pregunta, cada atributo es evaluado usando un test estadístico para determinar que tan bien clasifica él solo los ejemplos de entrenamiento. El mejor atributo es seleccionado y colocado en la raíz del árbol. Una rama y su nodo correspondiente es entonces creada para cada valor posible del atributo en cuestión. Luego para cada rama se aplica el mismo procedimiento con la tabla particionada por el valor del atributo especificado en la rama del árbol, sin incluir el atributo que clasificó anteriormente (nodo padre).

VII.- PRUEBAS

Las pruebas del sistema están orientadas a verificar el correcto funcionamiento de los algoritmos además de comprobar que el tiempo de respuesta del sistema corresponda a lo esperado por el cliente.

1.-Pruebas de funcionalidad

En estas pruebas se ejecuto cada algoritmo y se verificó que el resultado entregado por estos represente lo más fielmente posible a la realidad.

1.1.- Algoritmo A priori

De las reglas generadas, se escogieron de forma aleatoria 20 de éstas, para ser verificadas con la información de las transacciones que se encuentran en la base de datos, con el fin de probar si éstas reglas se debieron haber generado o no, y si el soporte y la confianza calculados por el sistema son los correctos.

.

A continuación se presenta un ejemplo con una de estas reglas:

La regla escogida es la siguiente, la cual fue generada mediante el análisis de todos los servicios de las secciones de Bar y Restaurant:

Cebiche; Corvina a la Plancha => Sauvignon Blanc Cousiño Macul
Soporte = 3% Confianza = 90%

Se verifican los siguientes datos en la base de datos:

1.- Total de transacciones: 6429
2.- Transacciones que incluyen Cebiche, Corvina a la Plancha: 214
3.- Transacciones que incluyen todos los servicios de la regla: 496

Soporte:

193/6429 = .03

Confianza:

193/214 = 0.9

El resultado obtenido coincidió con los valores entregados por el sistema, como así también el resultado del análisis de las 19 reglas restantes. Con lo cual se infiere que el algoritmo esta realizando su labor en forma correcta.

1.2.- Algoritmo C 4.5

Para validar el correcto funcionamiento de este algoritmo, se consideró el ejemplo dado en este informe en el capítulo VI Sección 5.2.2. Se cargaron los mismos datos presentados en este ejemplo en la base de datos, y se comprobó que el resultado, entregado por el sistema fuese el mismo que se muestra en el ejemplo.

En la figura 31 se aprecia el resultado de la prueba.

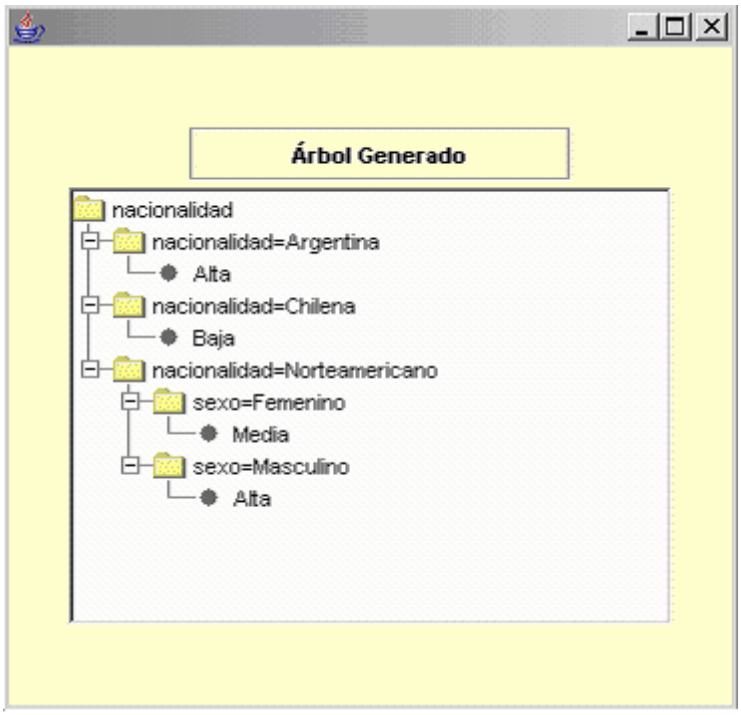


Figura 31. Resultado algoritmo C4.5

El árbol generado corresponde al obtenido en el ejemplo, y con lo cual se verificó el correcto funcionamiento del algoritmo.

1.3.- Algoritmo K-medias

La validación del correcto funcionamiento de este algoritmo es algo compleja, puesto que los grupos que formará dependen de los centros que se elijan en la primera iteración, y estos corresponden a centros aleatorios.
Por éste motivo, lo que se realizó para testear los resultados de este algoritmo fue crear dos grupos bien diferenciados, y luego verificar si el algoritmo logró separar estos grupos en forma correcta, sin ningún antecedente entregado anteriormente.

Se insertaron en la base de datos los siguientes valores:

- 100 clientes con edades entre 30 y 40 que hospedan en el hotel en temporada Baja
- 100 clientes con edades entre 60 y 70 que hospedan en el hotel en temporada Media

Luego se ejecutó el algoritmo con el parámetro K = 2, el resultado de éste se puede apreciar en la figura 32.

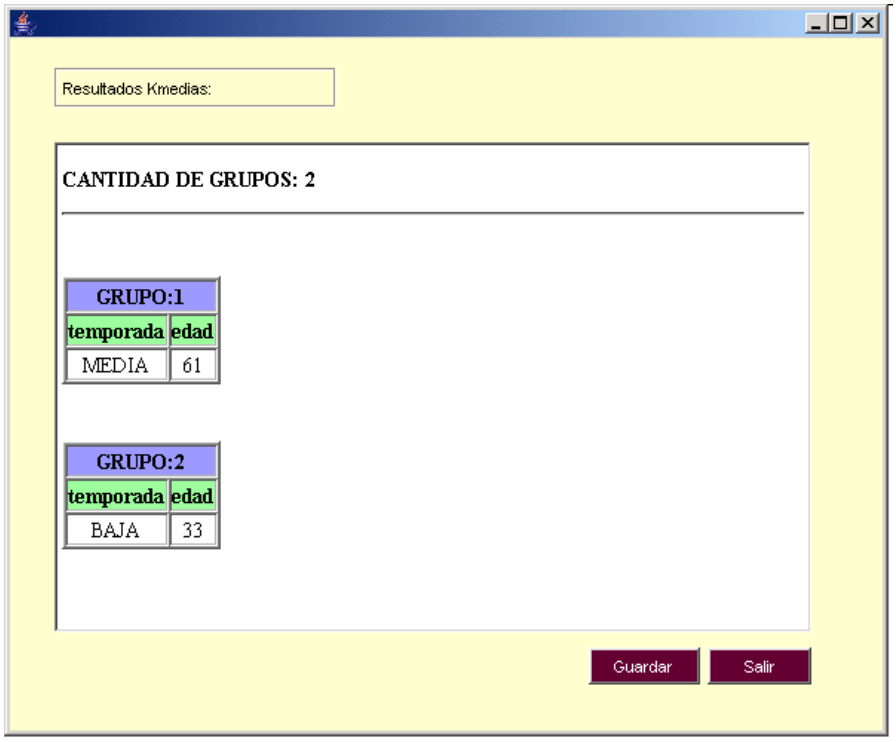


Figura 32. Resultado algoritmo K-medias

Lo que comprueba que el algoritmo tiene la capacidad de diferenciar clientes y crear grupos de acuerdo a estas diferencias.

2.- Pruebas de tiempo de respuesta

Estas pruebas están orientadas a medir el tiempo de respuesta de cada una las operaciones que realiza el sistema, y verificar que este tiempo sea aceptable para el usuario.

2.1.- Tiempo de traspaso de datos desde Access a SQL Server

Esta operación incluye el proceso de selección, limpieza y transformación de datos.
Los datos tratados en esta prueba son los que se muestran en la tabla de la figura 33.

Tabla	**Cantidad de registros**
clientes	1117
acompanantes	2428
reservas	1458
habitaciones	165
tipo_habitación	4
empleados	124
detalle_consumo	16524
servicios	184
tipo_servicio	10
secciones	3
documento	6429

Figura 33. Datos utilizados en la prueba

El tiempo en procesar estos datos fue en promedio 24 segundos en un computador de 1.47 GHZ. Y 512 MB de RAM. Este es un tiempo aceptable pues esta operación no se debe realizar siempre, solamente cuando el usuario decida actualizar los datos desde la base de datos operacional.

2.2.- Tiempo en generar reglas de asociación

Se ejecutaron 30 pruebas con el fin de medir el tiempo que demora el sistema en generar las reglas, y las variaciones de tiempo que se presentan al modificar los datos utilizados para generarlas.

Cantidad de transacciones: 6429

Cantidad de servicios analizados	Soporte	Confianza	Tiempo en generar las reglas (seg.)
2	10%	90%	0
2	2%	40%	1
20	10%	90%	3
20	2%	40%	4
60	10%	90%	8
60	2%	40%	10
120	10%	90%	14
120	2%	40%	26
184	10%	90%	24
184	2%	40%	31

Cantidad de transacciones: 3000

Cantidad de servicios analizados	Soporte	Confianza	Tiempo en generar las reglas (seg.)
2	10%	90%	0
2	2%	40%	0
20	10%	90%	1
20	2%	40%	2
60	10%	90%	3
60	2%	40%	5
120	10%	90%	7
120	2%	40%	9
184	10%	90%	12
184	2%	40%	15

Cantidad de transacciones: 500

Cantidad de servicios analizados	Soporte	Confianza	Tiempo en generar las reglas (seg.)
2	10%	90%	0
2	2%	40%	0
20	10%	90%	1
20	2%	40%	1
60	10%	90%	2
60	2%	40%	3
120	10%	90%	4
120	2%	40%	4
184	10%	90%	6
184	2%	40%	7

Los tiempos de respuesta alcanzan los 30 segundos solamente al analizar todos los servicios (184) con 6549 transacciones (BD Hotel) y con parámetros de soporte y confianza bajos, este tipo de consulta que debe generar una gran cantidad de subconjuntos, hecho por el cual se considera que el tiempo de respuesta es aceptable.

2.3.- Tiempo en generar árbol de decisión

Se ejecutaron 15 pruebas con el fin de medir el tiempo que demora el sistema en generar el árbol, y las variaciones de tiempo que presenta esta operación al modificar los datos utilizados para generarlo.

Cantidad de reservas analizadas: 1458

Cantidad características analizadas	Tiempo en generar el árbol (seg.)
2	4
4	5
6	7
8	10
9	16

Cantidad de reservas analizadas: 600

Cantidad características analizadas	Tiempo en generar el árbol (seg.)
2	2
4	3
6	4
8	7
9	10

Cantidad de reservas analizadas: 100

Cantidad características analizadas	Tiempo en generar el árbol (seg.)
2	0
4	1
6	2
8	5
9	7

Los tiempos de respuesta no superaron los 17 segundos, la variación de estos dependen de la cantidad de "características del cliente" analizadas y de la cantidad de reservas. Estos tiempos son aceptables para el usuario.

2.4.- Tiempo en generar grupos

Se ejecutaron 24 pruebas con el fin de medir el tiempo que demora el sistema en generar los grupos, y las variaciones de tiempo que presenta esta operación al modificar los datos utilizados para generarlos.

Cantidad de reservas analizadas: 1458

Cantidad características analizadas	Cantidad de grupos	Tiempo en generar los grupos (seg.)
1	2	1
1	4	1
2	2	1
2	4	2
3	2	1
3	4	2
4	2	2
4	4	2
5	2	5
5	4	6
6	2	7
6	4	8

Cantidad de reservas analizadas: 500

Cantidad características analizadas	Cantidad de grupos	Tiempo en generar los grupos (seg.)
1	2	0
1	4	0
2	2	1
2	4	1
3	2	2
3	4	2
4	2	2
4	4	3
5	2	3
5	4	4
6	2	5
6	4	5

Cantidad de reservas analizadas: 100

Cantidad características analizadas	Cantidad de grupos	Tiempo en generar los grupos (seg.)
1	2	0
1	4	0
2	2	0
2	4	0
3	2	0
3	4	1
4	2	1
4	4	1
5	2	2
5	4	2
6	2	2
6	4	3

Los tiempos no superaron los 10 segundos, éstos corresponden a tiempos de respuesta aceptables para el usuario, por lo que se considera ésta como una prueba exitosa.

VIII.- CONCLUSIONES

El disponer de información valiosa que ayude a fijar estrategias y tomar decisiones resulta fundamental para favorecer el desarrollo de cualquier tipo de organización. Esta información valiosa a menudo se encuentra "oculta" dentro de grandes cantidades de datos acumulados por años. La tecnología de Data Mining permite obtener este conocimiento o información útil mediante la automatización del proceso de extracción y análisis de los datos acumulados.

Los resultados obtenidos por el sistema, permitirán al Hotel San Martín hacer un uso práctico y eficiente de los datos de los clientes, obteniendo mayores ventajas que las que le ofrecen las herramientas disponibles actualmente.

Debido a que la tecnología de Data Mining abarca una gran gama de conocimientos sobre diferentes áreas, no es de extrañar que se piense que para utilizar este tipo de sistemas se requiera de algún conocimiento avanzado sobre el tema. Sin embargo con el desarrollo del proyecto se logró el objetivo de acercar al usuario hacia esta interesante y novedosa tecnología, de una manera amigable.

El Hotel San Martín, al pertenecer al ámbito turístico, le implica estar en un ambiente de alta competencia por la calidad de los servicios ofrecidos, es por esto que las organizaciones de este estilo requieren transformar los datos con los que cuentan a proyectos, ideas, etc., para obtener los objetivos que ellas mismas se plantean, y emprender campañas de marketing que en verdad los beneficien, y no solo a las compañías, sino que esto se traduzca en un servicio de calidad para el cliente.

El sistema desarrollado ayudará a tomar decisiones estratégicas a la empresa pues proveerá información sobre los gustos y preferencias de los clientes con la que no se contaba, la cual podrá ser utilizada para apoyar la toma de decisiones en este negocio.

En lo que se refiere al desarrollo del sistema en si, cabe destacar que una de las etapas en la que se centró el mayor esfuerzo fue la limpieza y transformación de los datos, puesto que dependerá de ésta la calidad de los datos con los cuales se esté trabajando.

En el proceso descrito en este informe se ha modelado completamente el sistema, y se han analizado y solucionado todos los problemas que surgieron en la realización de este proyecto, logrando un producto de software que cumple completamente los objetivos propuestos.

Glosario de términos

Antecedente: En una regla de asociación corresponde al término que aparece antes de la implicancia.

Clúster: Término utilizado para nombrar a cada grupo generado por un algoritmo de agrupamiento.

Confianza: Relativo al algoritmo A priori, dada una regla de asociación, corresponde al porcentaje de transacciones que conteniendo al antecedente, también contienen al consecuente.

Consecuente: En una regla de asociación corresponde al término que aparece después de la implicancia.

CRM: (Customer Relationship Management), CRM es una filisofía corporativa en la que se busca entender y anticipar las necesidades de los clientes existentes y también de los potenciales, que actualmente se apoya en soluciones tecnológicas que facilitan su aplicación, desarrollo y aprovechamiento. En pocas palabras, se trata de una estrategia de negocios enfocada en el cliente y sus necesidades.

Data Mining: Se refiere a la extracción o minería de conocimiento desde grandes cantidades de datos, mediante técnicas estadísticas y de inteligencia artificial.

Datawarehouse: Repositorio completo de datos de la empresa, donde se almacenan datos estratégicos, tácticos y operativos, con el objetivo de obtener información estratégica y táctica.

Entropía: Medida que sirve para calcular la ganancia de información en un árbol de clasificación.

KDD: (Knowledge Discovery in Databases) El Descubrimiento de Conocimiento en Bases de Datos es el proceso no trivial de identificación de patrones válidos, potencialmente útiles y comprensibles en los datos. El objetivo es la extracción de conocimiento de los datos, en el contexto de las bases de datos de gran tamaño.

Soporte: Relativo al algoritmo A priori, cantidad de ocurrencias de un itemset en la totalidad de transacciones registradas en la base de datos.

Referencias Bibliográficas

[HK01] Jiawei Han y Micheline Kamber, Data Mining concepts and techniques, AcademicPress 2001.

[Ag93] Agrawal R., Imielinski T., Swami A. Mining association rules between sets of items in large databases SIGMOD 1993.

[Qui93] Quinlan, Programs for Machine Learning, Morgan Kaufmann 1993.

[Sch99] Joseph Schmuller, Aprendiendo UML en 24 Horas, Prentice Hall 1999.

Referencias Web

[Web-1] http://www.cs.waikato.ac.nz/~ml/weka

[Web-2] http://pluton.ls.fi.upm.es/BDdeductivas/BasesDeductivas/

[Web-3] http://www.netbeans.org

[Web-4] http://www.informatica.us.es/~calvo/alumnos/t02es2inf02/tema3.4.html

[Web-5] http://www.infor.uva.es/~jmrr/TAD2003/Sesiones/TADONJava/JAVA.html

[Web-6] http://es.wikipedia.org/wiki/RUP