MATE-KDD: UNA HERRAMIENTA GENÉRICA PARA EL DESCUBRIMIENTO DE REGLAS DE CLASIFICACIÓN MEDIANAMENTE ACOPLADA AL SGBD POSTGRESQL

CLAUDIA MILENA CASTRO RODRIGUEZ
MARI ALEYDA CABRERA CABRERA

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA DE SISTEMAS
SAN JUAN DE PASTO
2007

MATE-KDD: UNA HERRAMIENTA GENÉRICA PARA EL DESCUBRIMIENTO DE REGLAS DE CLASIFICACIÓN MEDIANAMENTE ACOPLADA AL SGBD POSTGRESQL

CLAUDIA MILENA CASTRO RODRIGUEZ
MARI ALEYDA CABRERA CABRERA

Trabajo de Grado presentado como requisito parcial para optar al título de Ingeniero de Sistemas

RICARDO TIMARAN PEREIRA
Ph.D
Director del Proyecto

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA DE SISTEMAS
SAN JUAN DE PASTO
2007

“Las ideas y conclusiones aportadas en el trabajo de grado son responsabilidad exclusiva del autor”

Articulo 1° del acuerdo No. 224 de Octubre 11 de 1966, emanado del Honorable Consejo Directivo de la Universidad de Nariño.

Nota de aceptación

_____
_____
_____
_____
_____
_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

_____
Firma del gerente del proyecto

San Juan de Pasto, 16 de Febrero de 2007

A nuestras familias y amigos
por su afecto y apoyo incondicional

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

ABSTRACT

The investigations in Discovery of Knowledge in databases (DCBD), they were centered initially in to define models of discovery of patterns and to develop algorithms for these, one of these models but acquaintances are the " Classification " and inside this the managed technique is that of Trees of Decision". later Investigations focused is had in the problem of integrating DCBD with systems of databases, the development of systems and tools of Discovery of Knowledge whose architectures can be classified in three categories taking place as a result: weakly, fairly and strongly coupled with a System of Administration of databases (SGBD).

In this article the process of implementation of a tool is presented to discover rules of Classification fairly coupled the SGBD PostgreSQL by means of functions FDU and the evaluation of its yield.

Key words: System Agent of databases, Discovery of Knowledge in databases, Task of Classification, Defined Functions for the User FDU.

RESUMEN

Las investigaciones en Descubrimiento de Conocimiento en Bases de Datos (DCBD), se centraron inicialmente en definir modelos de descubrimiento de patrones y desarrollar algoritmos para éstos, uno de estos modelos mas conocidos es la “Clasificación” y dentro de esta la técnica más manejada es la de “Árboles de Decisión”. Investigaciones posteriores se han focalizado en el problema de integrar DCBD con sistemas de bases de datos, produciendo como resultado el desarrollo de sistemas y herramientas de Descubrimiento de Conocimiento cuyas arquitecturas se pueden clasificar en tres categorías: débilmente, medianamente y fuertemente acopladas con un Sistema de Gestión de Bases de Datos (SGBD).

En este artículo se presenta el proceso de implementación de una herramienta para descubrir reglas de Clasificación medianamente acoplada al SGBD PostgreSQL por medio de funciones FDU y la evaluación de su rendimiento.

Palabras claves: Sistema Gestor de Bases de Datos, Descubrimiento de Conocimiento en Bases de Datos, Tarea de Clasificación, Funciones Definidas por el Usuario FDU.

1. INTRODUCCIÓN

El desarrollo de una herramienta para el descubrimiento de reglas de clasificación medianamente acoplado al Sistema Gestor de Bases de Datos (SGBD) PostgreSQL, se presento como propuesta para proyecto de grado.

En éste capítulo se presenta las generalidades del problema que se identificó como objeto del estudio: La introducción al trabajo de grado realizado, la descripción del problema, los objetivos propuestos los cuales reflejan los alcances del proyecto y la organización del presente documento.

1.1 INTRODUCCIÓN

A través de los años, se ha dado un gran crecimiento en las capacidades de generar y colectar datos, debido básicamente a tres factores: primero está el gran poder de procesamiento de cómputo a bajo costo; segundo la acumulación de datos a una razón creciente, resultado del bajo costo de almacenamiento y tercero, la introducción de nuevos conjuntos de métodos desarrollados para el procesamiento de datos. De esta manera el crecimiento desmesurado de datos almacenados en las bases de datos excede las habilidades que tienen los seres humanos para analizar y obtener conocimiento a partir de ellas. Para tratar de descubrir conocimiento no explicito en grandes cantidades de datos, expertos en diversas áreas como Estadística, Inteligencia Artificial y Sistemas de Bases de Datos han propuesto un proceso cuyo objetivo es descubrir relaciones no evidentes entre los datos.

Los investigadores en DCBD, se centraron inicialmente en definir nuevas operaciones de descubrimiento de patrones y desarrollar algoritmos para estas. Investigaciones posteriores se han focalizado en el problema de integrar DCBD con Sistemas Gestores de Bases de Datos (SGBD), ofreciendo como resultado el desarrollo de herramientas DCBD cuyas arquitecturas se pueden clasificar en una de tres categorías: débilmente acopladas, medianamente acopladas y fuertemente acopladas con el SGBD[35], [41] .

Una herramienta DCBD debe integrar una variedad de componentes (técnicas de minería de datos, consultas, métodos de visualización, interfaces, etc.), que juntos puedan eficientemente identificar y extraer patrones interesantes y útiles de los datos almacenados en las bases de datos. De acuerdo a las tareas que desarrollen, las herramientas DCBD se clasifican en tres grupos: herramientas genéricas de tareas sencillas, herramientas genéricas de tareas múltiples y herramientas de dominio específico [17], [32], [35].

Las herramientas de Data Mining predicen futuras tendencias y comportamientos, permitiendo en los negocios tomar decisiones proactivas y conducidas por un conocimiento acabado de la información (***knowledge-driven***). Estas herramientas exploran las bases de datos en busca de patrones ocultos, encontrando información predecible que un experto no puede llegar a encontrar porque se encuentra fuera de sus expectativas.

Las técnicas de Data Mining pueden ser implementadas rápidamente en plataformas ya existentes de software y hardware para acrecentar el valor de las fuentes de información existente y pueden ser integradas con nuevos productos y sistemas pues son traídas en línea (***on-line***).

1.2 DEFINICIÓN DEL PROBLEMA

Muchos investigadores [8], [22], [36] han reconocido la necesidad de integrar los sistemas de descubrimiento de conocimiento y bases de datos, haciendo de esta un área activa de investigación. La gran mayoría de herramientas de DCBD tiene una arquitectura débilmente acoplada con un Sistema Gestor de Bases de Datos.

Algunas herramientas como ***Alice, C5.0_RuleQuest, Qyield, CoverStory*** ofrecen soporte únicamente en la etapa de minería de datos y requieren un pre y un post procesamiento de los datos. Hay una gran cantidad de este tipo de herramientas, especialmente para clasificación apoyadas en árboles de decisión, redes neuronales y aprendizaje basado en ejemplos. El usuario de este tipo de herramientas puede integrarlas a otros módulos como parte de una aplicación completa.

Otros ofrecen soporte en mas de una etapa del proceso de DCBD y una variedad de tareas de descubrimiento, típicamente, combinando clasificación, visualización, consulta y clustering, entre otras. En este grupo están ***Clementine, DBMiner, DBLearn, Data Mine, Intelligent Miner, Quest*** entre otras.

Todas estas herramientas necesitan de la adquisición de costosas licencias para su utilización. Este hecho limita a las pequeñas y medianas empresas u organizaciones, al acceso de herramientas DCBD para la toma de decisiones, que inciden directamente en la obtención de mayores ganancias y en el aumento de su competitividad.

Además, debido a que cada vez los volúmenes de datos son más grandes, las herramientas de minería de datos desarrollados se hacen ineficientes. De continuarse esta situación toda esa cantidad de datos obtenidos no harán mas que seguir ocupando espacio subutilizado, desechando la información que puede brindar apoyo para la toma de decisiones, lo que es una perdida de oportunidades de negocios.

Como respuesta al problema de la ineficiencia en la aplicación de los diferentes algoritmos en grandes volúmenes de datos, se han desarrollado “nuevos operadores del álgebra

relacional que soportan las nuevas primitivas con las que se extiende el lenguaje SQL para expresar eficientemente al interior de un SGBD, tareas de asociación y clasificación" [34].
Una alternativa de solución visible, es el desarrollo de una herramienta genérica DCBD, medianamente acoplada, bajo software libre, que permita el acceso a este tipo de herramientas sin ningún tipo de restricciones, a las pequeñas y medianas empresas y organizaciones de nuestro país o de cualquier parte del mundo y se ajuste a los requerimientos del entorno como son que sea intuitivo y de fácil accesibilidad.

1.3 OBJETIVOS

1.3.1. Objetivo general. Desarrollar una herramienta genérica para el Descubrimiento de reglas de Clasificación en bases de datos medianamente acoplada con el Sistema Gestor de Bases de Datos PosgresSQL, bajo los lineamientos del Software Libre.

1.3.2. Objetivos específicos

- Estudiar y analizar diferentes herramientas de DCBD para clasificación.
- Analizar, diseñar y desarrollar programas que permitan la selección, preprocesamiento y transformación de datos.
- Analizar, diseñar y desarrollar programas que implementen los operadores algebraicos y primitivas SQL para la tarea de Clasificación de datos.
- Evaluar la eficacia del nuevo algoritmo propuesto MATE-TREE.
- Realizar pruebas y comparar el rendimiento computacional y eficiencia de los algoritmos tratados (SLIQ, C4.5 y MATE-TREE), con bases de datos sintéticas que manejen grandes volúmenes de datos.
- Diseñar e implementar una interfaz de aplicación intuitiva y amigable al usuario para la visualización y manejo de los datos y resultados.
- Evaluar la calidad de la información generada.
- Documentar el software implementado, para facilitar su manejo.

1.4. ORGANIZACIÓN DEL DOCUMENTO

El resto de este documento esta organizado de la siguiente manera: En el **capítulo II,** se presenta un estudio sobre el Descubrimiento de Conocimiento en Bases de Datos, especificando las fases del proceso de DCBS, las diferentes arquitecturas y técnicas de Data Mining, y de esta forma entrar en materia y tener una noción clara acerca de la minería de datos. Una vez explicado el proceso de Data Mining, en el **capítulo III,** se presenta un

estudio detallado sobre el modelo de “Clasificación”, que es una de las técnicas de la minería la cual se tiene en cuenta para desarrollar este proyecto especificando sus principales características y aplicaciones, también se profundiza en la técnica de Árboles de Decisión y se detallan teóricamente los algoritmos que finalmente se implementan en nuestra herramienta; Mate-Tree, Slip y C4.5. En el **capítulo IV**, se describe el análisis y diseño de la herramienta de software para descubrimiento de reglas de clasificación en minería de datos. En el **capítulo V** se muestra de manera detallada la implementación de las funciones definidas por el usuario FDU en el Sistema Gestor de Bases de Datos Postgres. El **capítulo VI**, contiene las pruebas realizadas y el análisis de los resultados obtenidos. Finalmente, el **capítulo VII**, presenta las conclusiones y recomendaciones a las que se llegó con el desarrollo de este proyecto.

2. PROCESO DE DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS

En este capítulo se describe el marco teórico de la investigación en todo lo relacionado con el proceso de DCBD. En la primera sección se introducen los conceptos básicos, se define formalmente y se describe las fases del proceso KDD, en la segunda sección se describe las técnicas más conocidas de Data Mining, en la tercera sección se describe las Arquitecturas de Descubrimiento de Conocimiento en Bases de Datos y finalmente se hace una introducción al estándar CRISP-DM.

2.1 DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS

2.1.1. Introducción a KDD. La necesidad de conocimiento es una característica inherente al ser humano y un camino para obtenerlo es a través de la acumulación e interpretación de datos.

El descubrir de Conocimiento en Bases de Datos (DCBD) es básicamente un proceso automático en el que se combinan descubrimiento y análisis. El proceso consiste en extraer patrones en forma de reglas o funciones, a partir de los datos, para que el usuario los analice. Esta tarea implica generalmente preprocesar los datos, hacer minería de datos (*data mining*) y presenta resultados [8], [11], [15], [19], [20], [22], [32].

Este proceso, al cual se le ha dado entre otros los nombres de Descubrimiento de Conocimiento en Bases de Datos, *Data Mining*, Extracción del Conocimiento, Descubrimiento de Información, Arqueología de Datos y Procesamiento de Patrones de Datos, ofrece una alternativa para su extracción.

El término *KDD* (*Knowledge Discovery in Databases*, descubrimiento de conocimiento en Bases de Datos), apareció por primera vez a finales de los años 80 para enfatizar que el conocimiento es el producto de un descubrimiento guiado por los datos, y ha servido como punto de unión para las diferentes áreas de investigación que se dedicaban a estudiar el proceso de análisis de datos y extracción de conocimiento a partir de ellos desde distintos puntos de vista (*Data Mining*).

Data Mining ha sido comúnmente usado por estadísticos, analistas de datos y por la comunidad MIS (Sistema de Administración de Información), mientras que *KDD* ha sido principalmente usada por investigadores de inteligencia artificial y Aprendizaje Automático. En este documento, el término *KDD* se refiere al proceso global de descubrir conocimiento útil a partir de los datos, mientras que el de *Data Mining* se refiere a la etapa en la cual se aplican algoritmos para extraer patrones a partir de los datos [24], [30].

2.1.2. Definición de KDD. El proceso de descubrimiento de conocimiento en bases de datos (KDD) se define como ***"****El proceso no trivial de identificación de patrones válidos, nuevos, potencialmente útiles y comprensibles en los datos****"*** (Fallad et al., 1996), y consiste en el conjunto de técnicas avanzadas para la extracción de información escondida en grandes bases de datos. El proceso *KDD* tiene como metas principales procesar automáticamente grandes cantidades de datos crudos para identificar los patrones más significativos y relevantes, y presentarlos como conocimiento apropiado para satisfacer las metas del usuario.

En esta definición los datos hacen referencia a un conjunto de hechos (casos de una base de datos). Un patrón hace referencia a una expresión en algún lenguaje, que sirve para describir un subconjunto de datos o un modelo aplicable a esos datos. Es decir, un patrón es una instancia de un determinado modelo. Por ello, se entiende la extracción de patrones como la extracción de un modelo para unos datos, esto es, cualquier descripción de alto nivel de los datos.

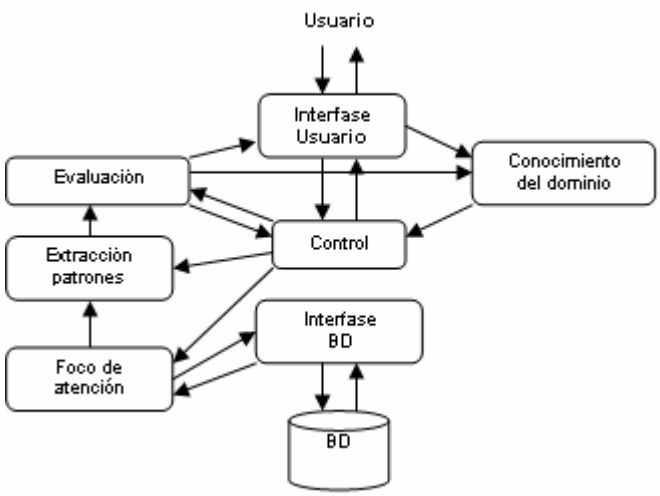
El término proceso implica que *KDD* es la conjunción de muchos pasos repetidos en múltiples iteraciones. Se dice por otra parte que es no trivial, porque se supone que hay que realizar algún tipo de proceso complejo. Los patrones deben ser válidos, con algún grado de certidumbre, y novedosos, por lo menos para el sistema y, preferiblemente, para el usuario, al que deberán reportar alguna clase de beneficio (útil). Por último los patrones deben ser comprensibles, si no de manera inmediata, sí después de alguna clase de preprocesado.

Todo lo indicado con anterioridad implica que se pueden definir medidas cuantitativas para evaluar los patrones obtenidos. Estas medidas pueden servir, por ejemplo, para evaluar la bondad, utilidad, simplicidad y certidumbre de los patrones. Las medidas de interés se pueden definir para ordenar los patrones obtenidos por un cierto sistema de *KDD*. El proceso de *KDD*, según lo expuesto anteriormente, es el proceso de aplicar a una determinada base de datos, las operaciones requeridas de selección, preprocesado, muestreo, transformación y métodos de Data Mining para extraer los patrones y posteriormente evaluarlos para identificar el modelo que representará al conocimiento. La fase de Data Mining del proceso de *KDD* tiene que ver tan sólo con la aplicación de los algoritmos de extracción de patrones. El proceso total incluye, tanto la preparación previa, como las posibles fases de interpretación de los patrones obtenidos [5], [14], [28].

2.1.3. Componentes del KDD. Dentro de los componentes que integran el Descubrimiento de Conocimiento en Bases de Datos se puede rescatar [24], [28]:

- **Conocimiento del dominio y preferencias del usuario:** incluye el diccionario de datos, información adicional de las estructuras de los datos, restricciones entre campos, metas o preferencias del usuario, campos relevantes, listas de clases, jerarquías de generalización, modelos causales o funciones, entre otros.

Figura 1. Componentes del KDD



El objetivo del conocimiento del dominio es orientar a ayudar en la búsqueda de patrones interesantes, aunque a veces esto pueda causar resultados contraproducentes.

- **Control del descubrimiento:** toma el conocimiento del domino, lo interpreta y decide qué hacer. En la mayoría de los sistemas el control lo realiza el usuario.

- **Interfaces:** Entre la base de datos y con el usuario.

- **Foco de atención:** específica a qué tablas, campos y registros acceder. Tiene que tener mecanismos de selección aleatoria de registros tomando muestras estadísticamente significativas, puede usar predicados para seleccionar un subconjunto de los registros que comparten cierta característica, etc. Algunas técnicas para enfocar la atención incluyen:

    - Agregación: consiste en juntar valores

    - Partición de datos: se particionan los datos en base a los valores que suman los atributos.
    - Proyección: consiste en ignorar algún o algunos atributos.

    - Extracción de patrones.

    - Evaluación.

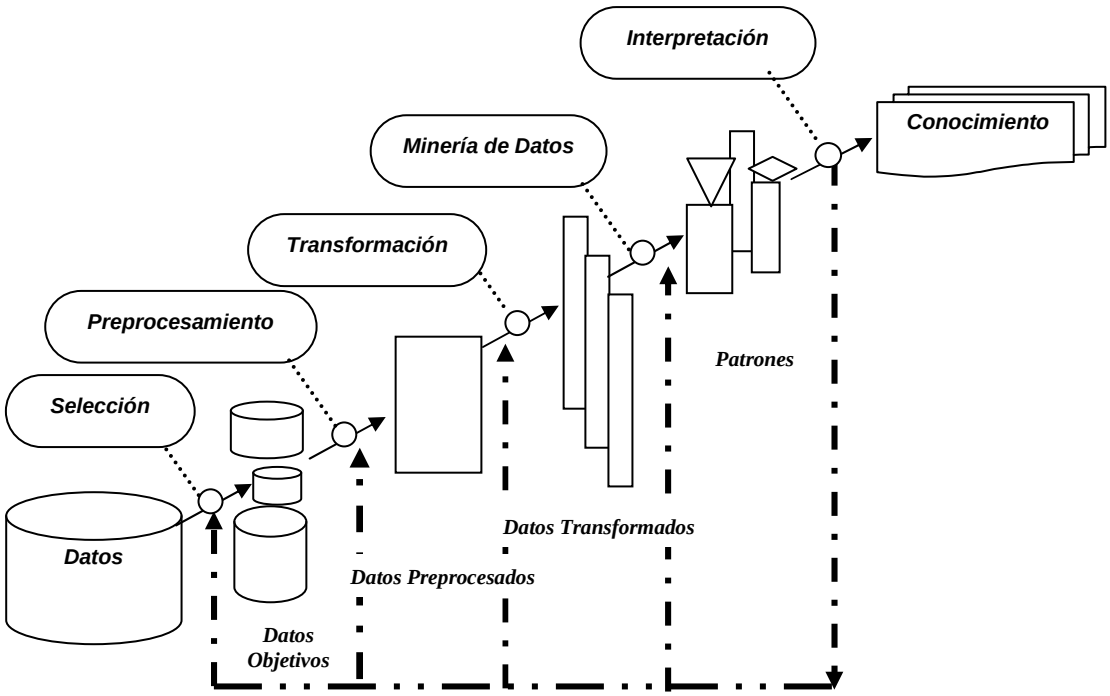
  La partición y proyección implican menos dimensiones. Y agregación y proyección implican menos dispersión. Los patrones son cualquier relación entre los elementos de la base de datos. Pueden incluir medidas de incertidumbre.

2.1.4 Fases del proceso KDD. El proceso de KDD es interactivo e iterativo, involucra numerosos pasos con muchas decisiones realizadas por el usuario que debe proporcionar los atributos que mejor modelizan los datos a estudiar. El proceso de descubrimiento del conocimiento toma los resultados tal como vienen de los datos, en forma cuidadosa y con precisión, y los transforma en información útil y entendible. No es información que habitualmente se pueda recuperar mediante técnicas normales, pero se descubre mediante el uso de técnicas de Inteligencia Artificial. [14].

El proceso de descubrimiento de conocimiento en Bases de Datos consta de las siguientes fases [9], [12], [21]: (Figura 2).

- **Definición de los objetivos.** En esta fase hay que estudiar el problema y decidir cuál es la meta del proyecto. Esta fase es similar a la de cualquier otro proyecto, y suele requerir la colaboración de un experto. Si se hace bien el planteamiento del problema, se descubren fácilmente las fuentes de datos y los algoritmos de Data Mining que se aplicarán. Un mal planteamiento del problema nos puede llevar a resultados erróneos.

Figura 2. Fases del proceso KDD



- **Selección de los datos.** Se identifican las fuentes de datos internas o externas y se selecciona el subconjunto de datos necesarios para la aplicación de Data Mining. Generalmente, los datos que se necesitan no están en una única base de datos, sino que están dispersos por varias fuentes distintas, cada una de ellas con sus propias tablas y sus propios atributos. Teniendo en cuenta la gran cantidad de tablas que

puede haber en las distintas bases de datos, no es difícil darse cuenta de lo costoso en tiempo que puede ser esta elección.

- **Preprocesamiento de los datos (limpieza).** Una vez identificados los datos a utilizar hay que estudiarlos para, por un lado entender el significado de los atributos y, por otro lado, para detectar errores de integración, como puede ser el hecho de que haya datos repetidos con distinto nombre o datos que significan lo mismo en diferente formato. Estos problemas surgen porque los datos pueden provenir de fuentes diferentes, y no todas almacenan la misma información de la misma manera. Con el preproceso de los datos lo que se consigue es tener un conjunto de datos adecuado para el correcto funcionamiento de las fases posteriores del proceso de KDD.

  Se deben eliminar el mayor número posible de datos erróneos o inconsistentes (limpieza) e irrelevantes (criba).

  Existe una enorme cantidad de técnicas que pueden aplicarse para optimizar la calidad de los datos con vista a la fase de minería de datos. Algunas técnicas son:

  - Tratamiento de valores ausentes, hay una gran cantidad de técnicas para realizar esta tarea: reemplazo el valor faltante con una constante global, reemplazo del valor faltante con la media, con la media de la clase e incluso técnicas completas que pretenden predecir el valor.

  - Reducción del volumen de datos, por ejemplo eliminando campos con bajo potencial de predicción o redundantes.

  - Histogramas: detección de datos anómalos.

  - Selección de datos: muestreo, ya sea verticalmente, eliminando atributos, u horizontalmente, eliminando tuplas.

  - Redefinición de atributos: agrupación o separación.

- **Transformación de los datos (reducción).** Se selecciona el algoritmo de Data Mining a aplicar así como los parámetros del algoritmo. Como cada algoritmo requiere un formato diferente en los datos de entrada, en esta fase se transforman los datos para que se ajusten al formato de entrada del algoritmo seleccionado. Permite llevar a cabo estrategias para representar los datos dependiendo de los requerimientos del usuario. Se utiliza reducción dimensional tanto vertical (eliminando atributos) como horizontal (eliminando filas duplicadas) para reducir el número de variables y la representación del conocimiento.

Transformación del Esquema:

- Esquema Original:
  Ventajas: Las R.I. se mantienen (no hay que reaprenderlas, no despistan)
  Inconvenientes: Muchas técnicas no se pueden utilizar.

- Tabla Universal: Cualquier Esquema Relacional se puede convertir (en una correspondencia 1 a 1) a una tabla universal.
  Ventajas: Modelos de aprendizaje más simples (preposicionales).
  Desventajas: Muchísima redundancia (tamaños ingentes).

- Desnormalizado tipo estrella o copo de nieve (*datamarts*):
  Ventajas: Se pueden buscar reglas sobre información sumariada y si resultan factibles se pueden comprobar con la información detallada.
  Desventajas: Orientadas a extraer un tipo de información (granjeros).

Intercambio de dimensiones: (filas por columnas)

- Es muy costoso: hay que mirar al menos la raíz cuadrada de todas las filas (cestas). Y puede haber millones en una semana...

Transformación de los campos:

- Numerización / Etiquetado
  Ventajas: Se reduce espacio. Ej.: apellido Þentero. Se pueden utilizar técnicas más simples.
  Desventajas: Se necesita meta-información para distinguir los datos inicialmente no numéricos (la cantidad no es relevante) de los inicialmente numéricos (la cantidad es relevante: precios, unidades, etc.)

- Normalización de los datos, por ejemplo, de una escala decimal al rango [0,1]

- Discretización: el proceso de discretización convierte valores de atributos continuos en otros discretos, mediante la especificación de un número de valores, los cuales se conocen como puntos de corte. Algunas de las técnicas de discretización son: Intervalo de igual amplitud, intervalo de igual frecuencia, métodos basados en entropía y métodos basados en error.
  Ventajas: Se reduce espacio. Ej. 0..10 Þ(pequeño, mediano, grande). Se pueden utilizar árboles de decisión y construir reglas discretas.
  Desventajas: Una mala discretización puede invalidar los resultados.

Pick & Mix:

- En minería de datos, generalmente los datos sugieren la creación de nuevos campos (columnas) por pick & mix.
  Ejemplos:

  - height^2/weight (índice de obesidad)
  - debt/earnings
  - passengers * miles
  - credit limit – balance
  - population / area

- **Minería de datos.** Es la etapa principal porque es en ella donde se aplican los diferentes algoritmos de análisis de datos sobre los datos transformados y preparados en las etapas anteriores. Durante la etapa de Data Mining se buscan los patrones presentes en los datos. En función del algoritmo seleccionado se obtendrá un formato diferente en la salida. En esta fase es posible aplicar varias veces el mismo algoritmo o utilizar conjuntamente varios algoritmos.

- **Análisis de los resultados.** En esta etapa es cuando se interpretan y evalúan los resultados obtenidos en la etapa de Data Mining. Se suele utilizar técnicas de visualización para ver los resultados obtenidos. Una vez presentados los resultados el usuario debe interpretar los resultados y si no está de acuerdo debe volver a aplicar los algoritmos con otros parámetros, e incluso ejecutar otro algoritmo de Data Mining, para obtener unos resultados más reales. Todo esto hace que el proceso de KDD sea iterativo. En esta fase también se debe determinar como utilizar los resultados obtenidos. Los resultados se pueden integrar en un sistema experto o como procedimientos almacenados en un gestor de base de datos.

Los pasos a seguir para la realización de un proyecto de minería de datos son siempre los mismos, independientemente de la técnica específica de extracción de conocimiento usada

2.2. DATA MINING

Data Mining es un proceso sobre datos que tiende hacia la entrega de información prospectiva y proactiva, está listo para su aplicación en la comunidad de negocios porque está soportado por tres tecnologías que ya están suficientemente maduras: Recolección masiva de datos, Potentes computadoras con multiprocesadores y Algoritmos o técnicas de Data Mining [5].

La minería de datos, colecciona los datos y espera que de ellos emerjan hipótesis. Se busca que los datos describan o indiquen por qué son como son. Luego entonces, se valida esa hipótesis inspirada por los datos en los datos mismos, será numéricamente significativa,

pero experimentalmente inválida. De ahí que la minería de datos debe presentar un enfoque exploratorio, y no confirmador.

Cuando se habla de la etapa de Data Mining, hace referencia *"al proceso de búsqueda de patrones, tendencias o relaciones, en los datos"*. *Data Mining* consiste en trabajar con un conjunto de algoritmos específicos inteligentes (técnicas), bajo limitaciones de eficiencia computacional para producir una serie de patrones sobre datos. Se parte de un cúmulo de datos y se llega a descubrir relaciones ocultas y complejas a partir de diversas operaciones, por lo general el espacio de patrones es grande, por lo tanto, la aplicación de los algoritmos se restringe a un subespacio de datos [2], [30].

Las dos metas u objetivos principales de los algoritmos de Data Mining son:

- **La verificación:** Donde el sistema se limita a verificar las hipótesis del usuario.

- **El descubrimiento:** El sistema autónomamente encuentra patrones. La información encontrada es novedosa, útil, comprensible para el usuario. De este modo a su vez incluye dos submetas: ***Descripción:*** el sistema hace un manejo de la información, análisis dimensional y encuentra patrones para presentarlos al usuario en una forma comprensible. ***Predicción:*** el sistema genera modelos y encuentra patrones para predecir comportamiento futuros de determinados objetos.

Las tareas primitivas que se pueden realizar, según el tipo de conocimiento que se requiera extraer, son [4]:

- **Predicción:** Es una tarea de aprendizaje de patrones por medio de ejemplos, utilizando un modelo de desarrollo para predecir valores futuros de una variable designada.

- **Caracterización:** Es la tarea de abstraer las características comunes de un conjunto de datos para poder describir cada grupo o clase. Por ejemplo, se pueden extraer patrones de llamadas en una empresa de telecomunicaciones o describir los síntomas de una enfermedad. Existen muchos métodos para extraer reglas de este tipo, entre los que se destacan la inducción orientada a atributos y la teoría de los Rough Sets.

- **Discriminación:** Consiste en extraer las características comunes de un grupo de datos, a los que se denominarán clase objetivo, que los distinguen con un factor de error de los miembros de otros grupos, que se denominarán clases discriminantes. Por ejemplo, se pueden encontrar los síntomas de una enfermedad que la distinguen totalmente del resto de enfermedades.

- **Clasificación:** Consiste en determinar la pertenencia de un objeto a un grupo de entre varios. Se suministra un conjunto de datos pertenecientes a diversas clases. A partir de ellos, se extraen las características que permitirán clasificar los nuevos datos en sus correspondientes clases. Algunos sistemas que permite extraer reglas de este tipo son ID3 o C4.5.

- **Asociación:** Es el proceso de descubrir relaciones o asociaciones entre los datos. Generalmente, estas reglas se buscan en las llamadas bases de datos transaccionales. Como ejemplo de aplicación de este tipo de algoritmos, se tiene el caso del análisis de la canasta familiar. En estas bases de datos, los nombres de los atributos son nombres de productos, cuyo valor es 0, si no está presente en la transacción, o 1 si está presente. De este modo, cada fila representa una "compra" determinada del cliente, y se trata de buscar relaciones entre los objetos de la transacción, es decir, siempre que se compra A, se compra B. Por ejemplo siempre que se compra Arroz se compra Aceite.

- **Clustering o Segmentación:** Consiste en dividir la población en clases o "clusters" con el criterio de minimizar la distancia entre elementos de una clase y minimizar la distancia entre clases. Una vez que todos los grupos han sido determinados, se extraen las características de cada cluster. Por ejemplo, se pueden agrupar piezas de maquinaria según su forma, y una vez que quedan determinados los clusters se pueden utilizar para describir ese conjunto de piezas.

- **Patrones Secuenciales:** Consiste en descubrir relaciones entre los datos, que se recogen durante un periodo. De este modo, se puede determinar si dado un suceso A, ocurrirá un suceso B después de n meses. Básicamente se trata de una variante de la consulta de asociación en las cuales el factor tiempo interviene de forma relevante.

- **Series Temporales:** Es el proceso descubrir similitudes entre diferentes conjuntos de datos que evolucionan en el tiempo, para tratar de identificar patrones de comportamiento. De este modo, se pueden identificar compañías con patrones de crecimiento similar, o averiguar si una partitura musical es similar a otra en algún fragmento.

2.2.1. Enfoques de Data Mining. En torno a la etapa de Data Minino se agrupan la mayor gama de algoritmos propuestos para el proceso *KDD*, como a*prendizaje automático* (**Maching Learning**), base de datos inteligentes, reconocimiento de patrones, razonamiento basado en casos, estadística y matemática, entre otros. Debido a está integración, en la etapa Data Mining se puede utilizar varios enfoques básicos, los cuales están estrechamente ligados a las áreas investigativas mencionadas anteriormente. Estos enfoques son [1]:

- **Enfoque matemático:** En estos enfoques se construye un modelo matemático para posteriormente extraer patrones del modelo. (Ejemplo: Teoría de Rouge Sets).

- **Enfoque estadístico:** Construye modelos con razonamiento estadístico para extraer los patrones y regularidades de los datos (Ejemplo: Las reglas Bayesianas).

- **Enfoque aprendizaje automático:** Utiliza modelos cognitivos para imitar el proceso de aprendizaje humano (Ejemplo: Aprendizaje a partir de ejemplos).
- **Enfoque orientado a Bases de Datos:** En este caso se utilizan técnicas y herramientas desarrolladas para diferentes tipos de bases de datos.

- **Enfoque integrados:** Constituyen técnicas unificadas que exploran las ventajas de dos o más enfoques.

2.2.2. Modelos y técnicas. Modelado es el acto de construir un modelo en una situación donde se conoce la respuesta y luego se la aplica en otra situación de la cual se desconoce la respuesta. Esto se ha estado haciendo desde hace mucho tiempo, aun antes del auge de la tecnología de Data Mining.

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

Las computadoras de cargan con mucha información acerca de una variedad de situaciones y luego el software de Data Mining debe correr a través de los datos y, mediante alguna técnica, distinguir las características de los datos que llevarán al modelo. Una vez que el modelo se construye, puede ser usado en situaciones similares donde no se conoce la respuesta.

Con Data Mining, la mejor manera para realizar esto es dejando de lado ciertos datos para aislarlos del proceso. Una vez que el proceso está completo, los resultados pueden ser comparados contra los datos excluidos para confirmar la validez del modelo. Si el modelo funciona, las observaciones deben mantenerse para los datos excluidos.

Por otro lado, la aplicación de los algoritmos de minería de datos requiere realizar una serie de actividades previas encaminadas a preparar los datos de entrada debido a que, en muchas ocasiones dichos datos proceden de fuentes heterogéneas, no tienen el formato adecuado o contienen ruido.

Luego se debe realizar el análisis preliminar de datos usando herramientas de consulta. Antes de aplicar las técnicas de Data Mining se debe aplicar una consulta SQL a un conjunto de datos, para rescatar algunos aspectos visibles. La gran mayoría de la

información, aproximadamente un 80%, puede obtenerse con SQL. El 20% restante conforma la información oculta, para la misma es donde se requiere utilizar de técnicas avanzadas.

Este primer análisis en SQL es para conocer cual es la distribución de los valores posibles de los atributos. Recién después se puede visualizar la *performance* del algoritmo correspondiente.

Los algoritmos de minería de datos se clasifican en dos grandes categorías: supervisados o predictivos y no supervisados o de descubrimiento del conocimiento [24] [29].

- **Modelos predictivos o supervisados.** Predicen el valor de un atributo de un conjunto de datos, conocidos otros atributos. A partir de datos cuyo atributo se conoce se induce una relación entre dicho atributo y otra serie de atributos. Básicamente entonces, hallan relaciones entre atributos.

Cuando una aplicación no es lo suficientemente madura y no tiene el potencial necesario para una solución predictiva, entonces hay que recurrir a los métodos no supervisados o de descubrimiento.

Las siguientes son algunas técnicas que se utilizan para crear modelos predictivos:

- **Clasificación:** se asigna los registros de datos en categorías predefinidas. Los datos son objetos caracterizados por atributos que pertenecen a diferentes clases. La meta es inducir un modelo para poder predecir una clase dados los valores de los atributos. Se utiliza:

  Algoritmos genéticos: técnicas de optimización que usan procesos tales como combinaciones genéticas, mutaciones y selección natural en un diseño basado en los conceptos de evolución. Inspirados en el principio de la supervivencia de los más aptos. La recombinación de soluciones buenas en promedio mejores soluciones. Es una analogía con la evolución natural.

  Redes neuronales artificiales: modelos predecibles no-lineales que aprenden a través del entrenamiento y semejan la estructura de una red neuronal biológica.

  Árboles de decisión: estructuras de forma de árbol que representan conjuntos de decisiones. Estas decisiones generan reglas para la clasificación de un conjunto de datos. Los métodos específicos de árboles de decisión incluyen:

    CART (Árboles de Clasificación y Regresión): Técnica utilizada para la clasificación de un conjunto de datos. Provee un conjunto de reglas

que se pueden aplicar a un nuevo conjunto de datos, sin clasificar, para predecir cuáles registros darán un cierto resultado. Segmenta un conjunto de datos creando dos divisiones. Requiere menos preparación de dato que CHAID.

CHAID (Detección de Interacción Automática de Chi cuadrado): técnica similar a la anterior, pero segmenta un conjunto de datos utilizando tests de chi cuadrado para crear múltiples divisiones.

- **Regresión o Estimación:** se predice el valor de atributos continuos partiendo de otros atributos conocidos. La meta es inducir un modelo para poder predecir el valor de la clase dados los valores de los atributos. También utiliza árboles de regresión, regresión lineal, redes neuronales.

- **Modelos de descubrimiento o no supervisados.** Descubren patrones y tendencias en los datos sin tener ningún tipo de conocimiento previo acerca de cuales son esos patrones buscados. El descubrimiento de esa información sirve para llevar a cabo acciones y obtener beneficios científicos o de negocios.
El rango de éxito de estos modelos depende de la utilidad del descubrimiento de conocimiento.

Las siguientes son algunas de las técnicas que se utilizan para crear modelos descriptivos:

- **Clustering:** agrupa los datos en determinadas categorías llamados clusters, basándose en las similitudes de los mismos. Existen diferentes técnicas de clustering y cada una de las mismas tiene sus propias aproximaciones para descubrir las aproximaciones que existen entre sus datos. Las aproximaciones las deben determinar los clusters. Las técnicas incluyen árboles de decisión, redes neuronales.

- **Análisis de enlace (Link Analysis):** describe una familia de técnicas que determinan asociaciones entre los registros de datos. El tipo de análisis de enlace más conocido es el Análisis de la canasta de mercado. El análisis de la canasta de mercados descubre la combinación de artículos que fueron comprados por diferentes consumidores, y por asociación o enlace se puede determinar que tipo de productos son comprados juntos. La canasta es un grupo de registros de datos, por lo tanto la técnica puede ser usada en cualquier situación donde haya un número grande de grupos de registros de datos. Incluye reglas de asociación. Patrones secuenciales, secuencias similares.

- **Análisis de frecuencia (Frequency Análisis):** comprende aquellas técnicas de minería de datos que son aplicadas al análisis de registros ordenados en el tiempo o cualquier conjunto de datos que puedan ser ordenado en el tiempo. Estas técnicas de minería de datos intenta detectar secuencias o subsecuencias similares en los datos ordenados. Incluye patrones secuenciales, secuencias similares.

2.2.3. Ventajas y desventajas de la Minería de Datos. Si bien la Minería de Datos se presenta como una tecnología emergente, posee ciertas ventajas [10]:

- Resulta un buen punto d encuentro entre los investigadores y las personas de negocios.

- Ahorra grandes cantidades de dinero a una empresa y abre nuevas oportunidades de negocios.

- Contribuye a la toma de decisiones tácticas y estratégicas proporcionando un sentido automatizado para identificar información clave desde volúmenes de datos generados por procesos tradicionales.

- Permite a los usuarios dar prioridad de decisiones y acciones mostrando factores que tienen un mayor valor en un objetivo.

- Generar modelos descriptivos: en un contexto de objetivos definidos en los negocios permite a empresas, sin tener en cuenta la industria o el tamaño, explorar automáticamente, visualizar y comprender los datos e identificar patrones, relaciones y dependencias que impactan en los resultados finales de la cuenta de resultados.

- Genera modelos predictivos: permite que relaciones no descubiertas e identificadas a través del proceso del Data Mining se expresen como reglas de negocio o modelos predictivos.

Existen elementos que la hacen operable, sin embargo, existen algunos factores que pueden crear un deslustre a la Minería de Datos, como son [10]:

- Algunos productos a comercializar son, en la actualidad, significativamente costosos, y los consumidores pueden hallar una relación costo/beneficio improductiva.

- Se requiere de mucha experiencia para utilizar herramientas de la tecnología, o que sea muy fácil hallar patrones equívocos, triviales o no interesantes.

- No es posible resolver los aspectos técnicos de hallar patrones en tiempo o en espacio.

Otro factor que se debe considerar es la privacidad. Antes se pensaba que la Minería de Datos no presentaba ningún peligro o riesgo para la privacidad de los clientes. Hoy en día, se piensa todo lo contrario, sin embargo, no existe un marco jurídico que haya mantenido el paso con el avance tecnológico.

2.2.4. Extensiones de Data Mining

- **Web Mining.** Consiste en aplicar las técnicas de Minería de Datos a documentos y servicios Web, con el fin de descubrir información o conocimiento potencialmente útil y previamente desconocido [24].

Todos los que visitan un sitio en Internet dejan huellas digitales como son direcciones de IP, navegador, entre otras, que los servidores automáticamente almacenan en una bitácora de accesos (log).

Las herramientas de *Web Mining* analizan y procesan estos log's para producir información significativa. Debido a que los contenidos de Internet consisten en varios tipos de datos, como texto, imagen, video, o metadatos, se utiliza el termino "minería de datos multimedia" como una instancia del *Web Mining* para tratar este tipo de datos. Los accesos totales por dominio, horarios de accesos más frecuentes y visitas por día, entre otros datos, se registran mediante herramientas estadísticas que complementan todo el proceso de análisis del *Web Mining*.

- **Text Mining.** El *Text Mining* se refiere a examinar una colección de documentos y descubrir información no contenida en ningún documento individual de la colección; en otras palabras, trata de obtener información sin haber partido de algo [24].

Dado que el ochenta por ciento de la información de una compañía esta almacenada en forma de documentos, las técnicas como la categorización de texto, el procesamiento de lenguaje natural, la extracción y recuperación de la información o el aprendizaje automático, apoya a la minería de texto.
En ocasiones se confunde el *Text Mining* con la recuperación de la información. Esta última consiste en la recuperación automática de documentos relevantes mediante indexaciones de textos, clasificación, y categorización.

- El cliente o el usuario se convierte en miembro del equipo

2.3. ARQUITECTURAS DE INTEGRACIÓN DEL PROCESO DCBD CON SGBD

Las arquitecturas de integración de herramientas DCBD con un SGBD se pueden ubicar en una de tres tipos: herramientas débilmente acoplados, medianamente acoplados y fuertemente acoplados con un SGBD.

Una arquitectura es débilmente acoplada cuando los algoritmos de Minería de Datos y demás componentes se encuentran en una capa externa al SGBD, por fuera del núcleo y su integración con este se hace a partir de una interfaz.

Una arquitectura es medianamente acoplada cuando ciertas tareas y algoritmos de descubrimiento de patrones se encuentran formando parte del SGBD mediante procedimientos almacenados o funciones definidas por el usuario.

Una arquitectura es fuertemente acoplada cuando la totalidad de las tareas y algoritmos de descubrimiento de patrones forman parte del SGBD como una operación primitiva, dotándolo de las capacidades de descubrimiento de conocimiento y posibilitándolo para desarrollar aplicaciones de este tipo.

Por otra aparte de acuerdo con las tareas que desarrollen, las herramientas DCBD se clasifican en tres grupos: herramientas genéricas de tareas sencillas, herramientas genéricas de tareas múltiples y herramientas de dominio específico.

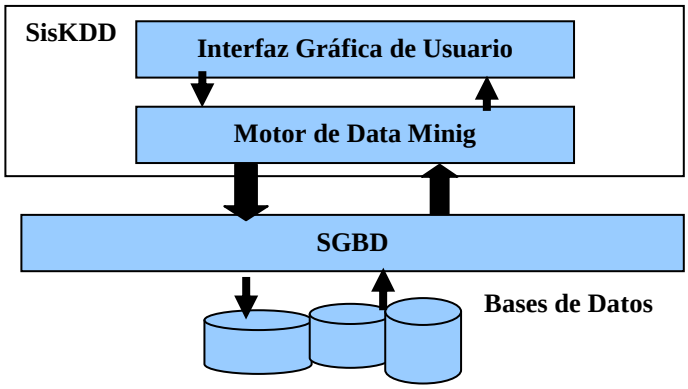
Las herramientas genéricas de tareas sencillas principalmente soportan solamente la etapa de minería de datos en el proceso de DCBD y requieren un pre y pos procesamiento de los datos. El usuario final de estas herramientas es típicamente un consultor o un desarrollador quien podrá integrarlas con otros módulos como parte de una completa aplicación.

Las herramientas genéricas de tareas múltiples realizan una variedad de tareas de descubrimiento, típicamente combinando clasificación, asociación, visualización, clustering, entre otros. Soportan diferentes etapas del proceso DCBD. El usuario final de estas herramientas es un analista quien entiende la manipulación de los datos.

Finalmente, las herramientas de dominio específico, soportan descubrimiento solamente en un dominio específico y hablan el lenguaje del usuario final, quien necesita conocer muy poco sobre el proceso de análisis [33].

2.3.1. Arquitectura DCBD débilmente acoplada. Una arquitectura es débilmente acoplada cuando los algoritmos de Minería de Datos y demás componentes se encuentran en una capa externa al SGBD, por fuera del núcleo y su integración con este se hace a partir de una interfaz cuya función, en la mayoría de los casos se limita a los comandos “leer de” y “escribir en”. En una arquitectura débilmente acoplada, los procesos de minería de datos corren en un espacio de direccionamiento diferente al del SGBD. Mientras el SGBD provee el almacenamiento persistente, la mayor parte del procesamiento de datos se realiza en herramientas y aplicaciones por fuera del motor del SGBD (ver figura 3).

Figura 3. Arquitectura DCBD débilmente acoplada



Esta arquitectura está presente en la mayoría de las herramientas de DCBD. Algunas como *Alice, C5.0_RuleQuest, Qyield, CoverStory* ofrecen soporte únicamente en la etapa de minería de datos y requieren un pre y un pos procesamiento de los datos. Hay una gran cantidad de este tipo de herramientas, especialmente para clasificación apoyadas en árboles de decisión, redes neuronales y aprendizaje basado en ejemplos. El usuario de este tipo de herramientas puede integrarlas a otros módulos como parte de una aplicación completa.

Otros ofrecen soporte en mas de una etapa del proceso de DCBD y una variedad de tareas de descubrimiento, típicamente, combinando clasificación, visualización, consulta y clustering, entre otras. En este grupo *están Clementine, DBMiner, DBLearn, Data Mine* entre otras.

La implementación de este tipo de arquitectura se hace a través de SQL embebido en el lenguaje anfitrión del motor de minería de datos. Los datos residen en el SGBD y son leídos registro por registro a través de ODBC, JDBC o de una interfaz de cursores SQL. La ventaja de esta arquitectura es su portabilidad. Sus principales desventajas son la escalabilidad y el rendimiento. El problema de escalabilidad consiste en que las herramientas y aplicaciones bajo este tipo de arquitectura, cargan todo el conjunto de datos en memoria, lo que las limita para el manejo de grandes cantidades de datos. El bajo rendimiento se debe a que los registros son copiados uno por uno del espacio de direccionamiento de la base de datos al espacio de direccionamiento de la aplicación de minería de datos y estas operaciones de entrada/salida, cuando se manejan grandes volúmenes de datos, son bastante costosas, a pesar de la optimización de lectura por bloques presente en muchos SGBD (*Oracle, DB2, Informix, etc*.) donde un bloque de tuplas puede ser leído al tiempo.

Estos sistemas, a pesar de que son fáciles de integrar a cualquier SGBD, son ineficientes en términos computacionales. El problema radica en que los costosos algoritmos de descubrimiento de conocimiento se implementan por fuera del núcleo del SGBD,

impidiendo cualquier aplicación de las técnicas de optimización de consultas por parte del optimizador del SGBD [35], [38].

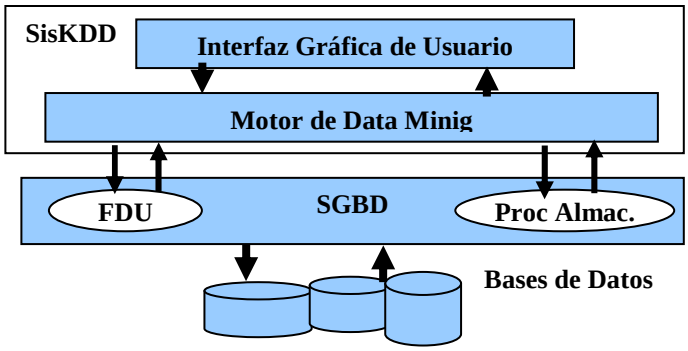
2.3.2. Arquitectura DCBD medianamente acoplada. Una arquitectura es medianamente acoplada cuando ciertas tareas y algoritmos de descubrimiento de patrones se encuentran formando parte del SGBD mediante procedimientos almacenados o funciones definidas por el usuario.

En una arquitectura medianamente acoplada las principales tareas de descubrimiento de conocimiento se descomponen en subtareas, algunas de las cuales se mueven al SGBD como consultas SQL en procedimientos almacenados o funciones definidas por el usuario (ver figura 4). La principal ventaja de esta arquitectura es que tiene en cuenta las capacidades de escalabilidad, administración y manipulación de datos del SGBD. Su mayor reto es obtener un buen rendimiento ya que el optimizador de consultas del SGBD no cuenta con nuevas estrategias de búsqueda y transformación que le permitan generar eficientes planes de ejecución, para las consultas que involucren descubrimiento de conocimiento.

Este problema se puede solucionar, en algún grado, implementando algoritmos ingeniosos, pero que incrementan la complejidad y el costo de desarrollo de estos sistemas.

En la arquitectura medianamente acoplada los algoritmos de descubrimiento son encapsulados como procedimientos[1] almacenados que corren en el mismo espacio de direccionamiento del SGBD. El motor de minería de datos invoca al procedimiento almacenado y le transmite los parámetros requeridos para hacer una tarea. La ventaja de este método es que brinda mejor desempeño, ya que ejecutan todas las conexiones, aplicaciones y la base de datos en el mismo espacio de direccionamiento, además, un procedimiento almacenado recibe igual trato que cualquier otro objeto de la base de datos y es registrado en su catálogo.

Figura 4. Arquitectura DCBD medianamente acoplada



[1] Un procedimiento almacenado es un conjunto nombrado de instrucciones y lógica de procedimientos de SQL compilado, verificado y almacenado en la base de datos.

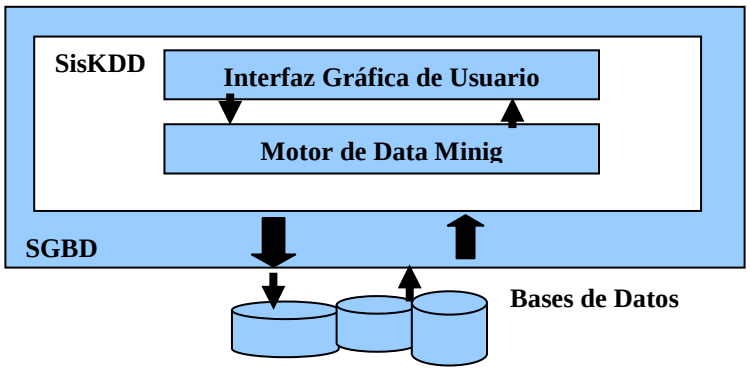
En la arquitectura medianamente acoplada por medio de funciones definidas por el usuario (FDUs), las funciones son definidas e implementadas en un lenguaje de programación de propósito general. El ejecutable de una FDU se almacena en el SGBD y este puede acceder e invocar la función en el momento que esta sea referenciada en una instrucción SQL. Los algoritmos de Minería de Datos se expresan como una colección de funciones definidas por el usuario. La mayoría del procesamiento se realiza en la FDU y el SGBD se usa principalmente para proveer tuplas hacia las FDUs.

Las FDUs, al igual que los procedimientos almacenados, corren en el mismo espacio de direccionamiento que el SGBD. La principal ventaja de las FDUs sobre los procedimientos almacenados es su tiempo de ejecución ya que pasar tuplas a un procedimiento almacenado es mas lento que hacia una FDU, debido a que en un procedimiento almacenado se deben ejecutar una serie de instrucciones SQL y en una FDU no. El resto del procesamiento es el mismo. La principal desventaja es el costo de desarrollo ya que los algoritmos de minería de datos tienen que ser escritos como FDUs [35], [38].

2.3.3. Arquitectura DCBD fuertemente acoplada. Una arquitectura es fuertemente acoplada cuando la totalidad de las tareas y algoritmos de descubrimiento de patrones forman parte del SGBD como una operación primitiva, dotándolo de las capacidades de descubrimiento de conocimiento y posibilitándolo para desarrollar aplicaciones de este tipo (ver figura 5).

En una arquitectura fuertemente acoplada, un algoritmo completo de descubrimiento de conocimiento se integra al motor de un SGBD como una primitiva. Debido a que todos los algoritmos son ejecutados conjuntamente con los datos en el SGBD, la ventaja potencial de este enfoque es que resuelve los problemas de escalabilidad y rendimiento de las otras arquitecturas. Talvez la mayor limitación de este método es que los desarrolladores de aplicaciones y herramientas DCBD no están dispuestos a poner algoritmos completos en los sistemas de bases de datos por razones de competencia. Por otro lado, muchos algoritmos cambian frecuentemente con los resultados de las nuevas investigaciones, lo que dificulta su oportuna actualización [35], [35], [38].

Figura 5. Arquitectura DCBD fuertemente acoplada



Hay propuestas de investigación que discuten la manera como tales sistemas pueden ser implementados y las extensiones del lenguaje SQL necesarias para soportar estas operaciones de descubrimiento de patrones: DMQL, M-SQL, MINE RULE y NonStop SQL/MX .

Timarán [36], [37] plantea un nuevo método de acoplar las tareas de Asociación y Clasificación a un SGBD. El método consiste en implementar inicialmente operadores algebraicos al interior del motor del SGBD que faciliten estas tareas y extender el lenguaje SQL con nuevas primitivas que los soporten. Su implementación permitirá contar verdaderamente con una arquitectura de DCBD fuertemente acoplada con un SGBD. Y el desarrollo de esta propuesta se encuentra en [1], [38].

2.4. EL ESTÁNDAR CRISP-DM

CRISP-DM es un estándar de desarrollo de soluciones de data mining. Una de las grandes ventajas del estándar es que no está ligado a ningún producto comercial ni a ningún tipo específico de aplicaciones.

El documento donde se recoge el estándar describe la metodología de data mining *Crisp-DM* como un modelo de proceso jerárquico. Este modelo se divide en cuatro niveles de abstracción, siendo el primero el más general y el último el más específico.

Dentro del primer nivel, en proceso de data mining se organiza en un cierto número de fases, constituidas por varias tareas genéricas de segundo nivel que se pretende cubran todas las posibles situaciones que se puedan encontrar. El tercer nivel describe cómo se llevarán a cabo las acciones de dichas tareas genéricas en situaciones específicas. En el cuarto nivel se tiene un registro de las acciones, decisiones y resultados del caso de data mining abordado [7].

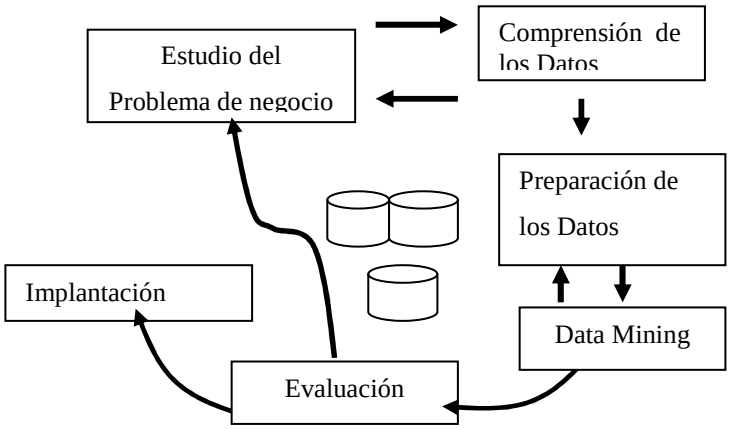
2.4.1. El modelo de referencia CRISP-DM. El modelo actual proporciona una vista general del ciclo de vida del proyecto de data mining. Contiene fases, relaciones entre ellas y sus tareas específicas. Estas relaciones dependen de los objetivos, los datos, el usuario, etc. Este ciclo de vida se compone de seis fases, dependientes entre si. Son las siguientes [7]: (Figura 6).

- **Determinación del dominio:** Se debe identificar cual es el problema que se desea resolver, aun con la suposición, o bien conociendo, que la respuesta se halla sepultada en alguna parte de los datos, aunque no se sabe exactamente donde está.

Es indispensable entonces contar con una clara descripción del problema a resolver; comprender sobre el total de los datos, cuales pueden ser los más importantes, y tener una visión lo más acabada posible de que tan lejos se está dispuesto a llegar, una vez que se conozcan los resultados. Es más, saber que tan lejos se está dispuesto a llegar permite

muchas veces clarificar cual es el problema, e inclusive determinar que datos se podrían utilizar.

Figura 6: Modelo CRISP-DM



En el momento de identificar el problema, se debe pensar en términos de patrones y relaciones, que es con lo que opera el Data Mining.

Es muy importante comprender los datos pues en esta fase se determina que las fases sucesivas sean capaces de extraer conocimiento válido y útil a partir de la información original. Se debe determinar si los datos con los que se cuenta son suficientes para hallar la solución, si son realmente útiles. Algunas veces no resulta obvio que esos datos no pueden proveer la respuesta que se está buscando, por ello la importancia de prestar total atención a este punto.

♦ **Diseñar el almacén de datos:** Se debe diseñar un esquema del almacén de datos que consiga unificar de manera operativa toda la información recogida.

La información que se quiere investigar generalmente se encuentra en bases de datos muy diversos, tanto internas como externas, que suelen utilizarse para el trabajo transaccional.

Se debe seleccionar un conjunto o subconjunto de bases de datos, enfocar la búsqueda en subconjuntos de variables, y seleccionar muestras de datos en donde realizar el proceso de descubrimiento.

Los datos y metadatos conforman lo que se denomina el Modelo de Datos. Típicamente, un modelo de datos define las fuentes de datos utilizadas los tipos de dato, su contenido, descripción, y el uso de los mismos.

La fuente de datos indica la ubicación física donde éstos fueron almacenados. El tipo define como están estructurados los mismos, por ejemplo, el formato de tiempo que utilizan. El

contenido de datos lista las tablas o archivos de datos, y los campos que éstos contienen. El uso de datos considera la pertenencia de las tablas y campos, que usuarios comprenden su contenido, y cuales lo explotan.

El modelo de datos también contiene información acerca de cuando se considera al dato como válido.

Cada registro puede contener una o más variables, donde cada variable puede derivar de un cierto número de fuentes diferentes. En la mayoría de las aplicaciones, los tipos de datos más comunes son:

- Datos transaccionales: son datos propios de las operaciones diarias realizadas. Algunas de estas transacciones incluyen cierto grado de detalle respecto a su contenido.

- Datos relacionales: son datos de contenido relativamente no volátil. Por lo general se trata de información estable sobre clientes, equipos, productos, y procesos de trabajo.

- Datos demográficos: provenientes de fuentes externas.

Cuando el modelo de datos es requerido para soportar una aplicación que contiene requerimientos específicos, entonces los datos a utilizar pueden ser definidos por los usuarios finales. Se los debe interrogar acerca de cual es la información que están necesitando, y en base a ello, realizar las agregaciones necesarias para soportar estos requerimientos.

Generar un modelo de datos para aplicaciones de Minería de Datos puede consumir mucho más tiempo del que se cree.

La tendencia es utilizar manejadores de bases de datos y almacenes de datos que están optimizados para realizar un proceso analítico.

♦ **Seleccionar y preprocesar los datos:** Una vez definido el modelo de datos que proveerá la estructura necesaria, en términos de las variables que se van a minar, es necesario suministrar los datos a utilizar.

Se debe identificar, coleccionar, filtrar y agregar los datos en el formato requerido por el modelo de datos.

El proceso que abarca desde la identificación de los datos hasta su preparado, es el que mayor tiempo consume en cualquier proceso de minería. Al identificar las fuentes se debe considerar su origen y su contenido.

La Minería de Datos, al igual que otras herramientas de análisis, requiere que los datos estén consolidados en una única tabla o archivo. Si las variables requeridas están distribuidas alrededor de un número variado de fuentes, entonces se debe realizar la consolidación de las mismas, de tal manera que se obtenga un conjunto de registros de datos consistentes.

Si los datos a utilizar no provienen de un Data Warehouse[2], entonces se deben aplicar funciones de preproceso para la limpieza, agregación, trasformación y filtrado.

Las herramientas de Minería de Datos usualmente proveen capacidades limitadas para limpiar los datos debido a que este es un proceso especial pero existen una gran variedad de productos que pueden cumplir con este propósito eficientemente.

Agregación y filtrado se pueden realizar por diferentes vías, según la estructura precisa de las fuentes de datos.

Se debe diseñar una estrategia adecuada para manejar ruido, valores faltantes, valores incompletos, valores no existentes, secuencias de tiempo, y casos extremos de ser necesario

- **Evaluar el modelo de datos:** Se realiza una evaluación inicial del modelo de datos. Para ello se lleva a cabo una serie de pasos:

El primer paso radica en la inspección visual, consiste en "curiosear" los datos de entrada con herramientas de visualización. Esto permite principalmente, detectar distribuciones inverosímiles.

El segundo paso tiene que ver con la identificación de inconsistencias y la resolución de errores. Excepcionalmente, las distribuciones halladas en el paso se originan por la mala recolección de los datos. Los valores distantes o ausentes producen resultados parciales.

El último paso consiste en la selección final de las variables para comenzar a correr el proceso de minería.

Las variables dependientes o altamente correlacionadas pueden ser halladas mediante tests estadísticos, o regresión lineal o polinómica. Luego del chequeo estadístico, no todas las variables son nominadas a permanecer, sino únicamente aquellas que posean una clara interpretación, y aquellas que tengan sentido para el usuario final.

Este paso se puede simplificar si se utiliza un modelo de datos previamente examinado.

[2] Es una técnica para consolidar y administrar datos de variadas fuentes con el propósito de responder preguntas de negocios y tomar decisiones, de una forma que no era posible hasta ahora.

♦ **Seleccionar la técnica de minería de datos:** Se debe seleccionar las técnicas de Minería de datos más satisfactoria.

Este paso no implica únicamente definir la técnica apropiada o la combinación de técnicas a utilizar, sino también la vía a través de la cual se aplicará la misma.

Existen varios tipos de técnicas o algoritmos disponibles: clasificación, asociación, clustering, predicción de valores, patrones similares, y secuencias similares.

La selección del método a utilizar a menudo resulta obvia. Dependerá del tipo de conocimiento que se desea extraer. Usualmente, el desafió no está en que técnica utilizar, sino en el camino a seguir para aplicarla.

Las técnicas de Minería de Datos requieren la selección de algunos parámetros, pero para ello hay que saber como funcionan estas técnicas, y que realiza cada uno de los parámetros. Se tiene que especificar un criterio de preferencia para seleccionar un modelo dentro de un conjunto posible de modelos, y se debe especificar la estrategia de búsqueda a utilizar, aunque generalmente se halla predeterminada en el algoritmo de minería.

♦ **Interpretar los resultados:** Los resultados que se obtienen al aplicar alguna técnica de minería de datos pueden proporcionar una gran riqueza de información. Pero esta información muchas veces resulta difícil de interpretar. Por ello, es muy importante presentar estos resultados mediante formas fáciles de comprender.

Para esta etapa es necesario disponer de un conjunto de herramientas que ayuden a visualizar los resultados y así proveer la información que se desee.

De ser necesario, es posible regresar a pasos anteriores. Esto puede involucrar repetir el proceso, quizás con otros datos, otros algoritmos, otras metas y otras estrategias.

La interpretación puede servir también para eliminar patrones redundantes o irrelevantes.

♦ **Desplegar los resultados:** Quizás esta sea la etapa más importante de todas. Si la Minería de Datos se la considera únicamente como una herramienta de análisis, desde ya se esta fracasando al realizar el potencial análisis que ésta tiene para ofrecer.

La Minería de Datos crea representaciones de los datos que son llamados modelos. Estos modelos son muy importantes, porque no solamente proveen un profundo conocimiento de la organización en sí, sino que también se puede desplegar para utilizarse en otros procesos de negocios.

El conocimiento se obtiene para realizar acciones, la idea es incorporarlo dentro de un sistema de desempeño o simplemente almacenarlos y reportarlo a las personas interesadas.

3. CLASIFICACIÓN

En este capítulo se presenta el estado del arte del modelo de *Data Mining* "Clasificación". En la primera sección se describe sus principales características y los tipos de problemas de Clasificación que existen. En la siguiente sección se describen las técnicas del modelo de Clasificación. En la tercera sección se profundiza en la técnica de Árboles de Decisión. En la cuarta sección se presentan los algoritmos más conocidos basados en árboles de decisión. En la quinta sección se describen el algoritmo C4.5. En la sexta el Algoritmo SLIQ. En la séptima el algoritmo Mate-Tree. y finalmente, se hace un paralelo entre herramientas de DM que usan el modelo de clasificación y MATE-KDD.

3.1. MODELO DE CLASIFICACIÓN

*"La **clasificación** es un problema multivariado que consiste en asignar individuos u objetos a uno de N grupos o clases basándose en mediciones de las variables x_1, x_2, ….,x_p, tomadas en ellas"*.

La clasificación de datos es el proceso por medio del cual se encuentra propiedades comunes entre un conjunto de objetos de una base de datos y se los clasifica en diferentes clases, de acuerdo al modelo de clasificación. Este proceso se realiza en dos pasos: en el primer paso se construye un modelo en el cual, cada tupla, de un conjunto de tuplas de la base de datos, tiene una identidad de clase conocida (etiqueta), determinada por uno de los atributos de la base de datos, llamado atributo clase. El conjunto de tuplas que sirve para construir el modelo se denomina conjunto de entrenamiento y se escoge randomicamente del total de tuplas de la base de datos. A cada tupla de este conjunto se denomina ejemplo de entrenamiento [18]. En el segundo paso, se usa el modelo para clasificar. Inicialmente, se estima la exactitud del modelo utilizando otro conjunto de tuplas de la base de datos, cuya clase es conocida, denominado conjunto de prueba. Este conjunto es escogido randomicamente y es independiente del conjunto de entrenamiento. A cada tupla de este conjunto se denomina ejemplo de prueba [18].

La exactitud del modelo, sobre el conjunto de prueba, es el porcentaje de ejemplos de prueba que son correctamente clasificadas por el modelo. Si la exactitud del modelo se considera aceptable, el modelo puede ser usado para clasificar futuros datos o tuplas para los cuales no se conoce a que clase pertenecen.

Los datos de entrada son instancias de las clases que se desean modelar e incluyen una serie de atributos o características. El objetivo de la clasificación es obtener una descripción precisa para cada clase utilizando los atributos de los datos de entrada. El modelo así

obtenido puede servir para clasificar casos cuyas clases se desconozcan o, simplemente, para comprender mejor la información disponible [3].

El modelo de clasificación puede construirse entrevistando a expertos. La mayor parte de los sistemas basados en conocimiento [SBC en castellano o KBS en inglés] se han construido así a pesar de la dificultad que la extracción manual del conocimiento. No obstante, si se dispone de suficiente información registrada (en una base de datos), el modelo de clasificación se puede construir generalizando a partir de ejemplos específicos mediante algún proceso inductivo.

Los casos de entrenamiento utilizados en la construcción del modelo de clasificación suelen expresarse en términos de un conjunto finito de propiedades o atributos con valores discretos o numéricos. Las categorías a las que han de asignarse los distintos casos se deben establecerse de antemano (aprendizaje supervisado). En general, estas clases serán disjuntas (aunque pueden establecerse jerarquías) y deberán ser discretas (para predecir atributos con valores continuos se suelen definir categorías discretas utilizando términos imprecisos propios del lenguaje natural).

Las técnicas inductivas de clasificación se basan en el descubrimiento de patrones en los datos de entrada, por lo que se dispone de suficientes casos de entrenamiento (bastantes más que clases diferentes) para obtener un modelo de clasificación fiable. Se necesitan bastantes datos para poder diferenciar patrones válidos de patrones debidos a irregularidades o errores. Esta diferenciación se suele realizar utilizando alguna técnica estadística.

Suponiendo que todos los patrones a reconocer son elementos potenciales de clases distintas denotadas, se denomina $\Omega = \{\omega j \mid l<=j<=J\}$ al conjunto de las clases informacionales. En ocasiones se extiende Ω con una clase de rechazo ω_o a la que se le asigna todos los patrones para los que no se tiene una certeza aceptable de ser clasificados correctamente en alguna de las clases de Ω. De este modo, $\Omega^* = \{\omega_o\} \upsilon\ \Omega$ es el conjunto extendido de clases informacionales. Un clasificador o regla de clasificación es una función $d: P \rightarrow \Omega$ definida sobre el conjunto de patrones tal que para todo patrón X, $d(X) \in \Omega^*$ [6], [27].

3.1.1. Tipos de problemas de clasificación

- ***Análisis discriminante (conocido también como reconocimiento de patrones supervisado***). En este caso se dispone de un conjunto de observaciones multivariadas, para las cuales se conoce a priori la clase a la cual pertenecen, ésta es llamada una muestra de entrenamiento. Es decir, cada caso en la muestra consiste del valor de una variable de respuesta *y* y de un vector *x=(x1,x2,…..xp)* que representa *p* características de un sujeto u objeto. Con los datos de la muestra de entrenamiento se encuentra la regla de clasificación o clasificador.

También se tiene otro conjunto de datos, con las mismas propiedades distribucionales de la muestra de entrenamiento y el cual es usado para validar el clasificador, éste es llamado la muestra de prueba. Algunas veces la misma muestra de entrenamiento (o un porcentaje de ella) es usada como muestra de prueba.

- ***Análisis por conglomerados (conocido también como reconocimiento de patrones no supervisado)***. En este caso los grupos no son conocidos de antemano, es decir no hay variable de respuesta. El objetivo del análisis es determinar los agrupamientos de datos tales que los elementos dentro del mismo grupo son más similares u homogéneos que aquellos que pertenecen a otros grupos. Algunas veces este análisis es usado como una etapa previa del análisis discriminante.

3.2 TÉCNICAS DE CLASIFICACIÓN

Los métodos de aprendizaje basados en reglas de clasificación buscan obtener reglas o árboles de decisión que particionen un grupo de datos en clases predefinidas. Para cualquier dominio real, el espacio de datos es demasiado grande como para realizar una búsqueda exhaustiva en el mismo.

En cuanto a los métodos inductivos, la elección del atributo para cada uno de los nodos se basa en la ganancia de entropía generada por cada uno de los atributos. Una vez que se ha recopilado la información acerca de la distribución de todas las clases, la ganancia en la entropía se calcula utilizando la teoría de la información o bien el índice de *Gini* [6], [27].

3.2.1. Reglas de producción. Conforme el tamaño los árboles de decisión aumenta, su inteligibilidad disminuye. Cuando el problema de clasificación es complejo, el árbol de decisión generado es tan grande que ni siquiera los expertos pueden comprender el modelo de clasificación construido (ni siquiera simplificándolo al podar el árbol).

Shapiro propuso descomponer un árbol de decisión complejo en una jerarquía de pequeños árboles de decisión para obtener un modelo más comprensible [*Structured induction*]. Sin embargo, es mucho más sencillo expresar el árbol de decisión construido como un conjunto de reglas de producción, una forma de representación del conocimiento más inteligible que los árboles.

Las reglas de producción se pueden derivar de un árbol de decisión con facilidad. El algoritmo que nos permite realizar este cambio de modelo de representación es muy sencillo: de cada camino desde la raíz del árbol hasta un nodo hoja se deriva una regla cuyo antecedente es una conjunción de literales relativos a los valores de los atributos situados en los nodos internos del árbol y cuyo consecuente es la decisión a la que hace referencia la hoja del árbol (la clasificación realizada).

También existen otros métodos de clasificación que obtienen reglas de producción directamente, sin necesidad de construir previamente un árbol de decisión. Estas técnicas,

más ineficientes, suelen emplear estrategias de búsqueda heurística como la búsqueda dirigida, una variante de la búsqueda primero el mejor.

Michalski y sus colaboradores desarrollaron, bajo el nombre de metodología STAR, un conjunto de técnicas de aprendizaje inductivo incremental basadas en la utilización de expresiones lógicas en forma normal disyuntiva (modelo de representación más expresivo que el empleado por los algoritmos de construcción de árboles de decisión). Estas expresiones lógicas describen conceptos y se pueden utilizar directamente como reglas de clasificación. Entre los algoritmos desarrollados en los años 80 en el entorno de Michalski destacan INDUCE y AQ, del que existen múltiples variantes.

Con posterioridad a los trabajos de Michalski, Peter Clark y su equipo propusieron el algoritmo CN2 en 1989. Este algoritmo intenta combinar adecuadamente las mejores cualidades de los algoritmos TDIDT y AQ: CN2 trata de combinar la eficiencia y el manejo de información con ruido que permite la familia de algoritmos TDIDT con la flexibilidad de la familia AQ en su estrategia de búsqueda de reglas.

3.2.2. Inducción en árboles de decisión. El modelo de clasificación basado en árboles de decisión es un método de aprendizaje supervisado que construye árboles de decisión a partir de un conjunto de casos o ejemplos (training set) extraídos de la base de datos. También se escoge un conjunto de prueba, cuyas características son conocidas, con el fin de evaluar el árbol. Un árbol de decisión es una estructura donde:

- Cada nodo interno o no terminal está etiquetado con un atributo.
- Cada rama que sale de un nodo está etiquetada con un valor de ese atributo.
- Cada nodo terminal u hoja representa una clase.
- El nodo inicial o superior se denomina raíz.

Uno de los algoritmos de clasificación con árboles de decisión mas utilizados es el C4.5 [3], [5]. Este algoritmo construye el árbol de decisión de arriba hacia debajo de forma recursiva. Se trata de construir el árbol de decisión más simple que sea consistente con el conjunto de entrenamiento T. Para ello hay que ordenar los atributos relevantes, desde la raíz a los nodos terminales, de mayor a menor poder de clasificación. El poder de clasificación de un atributo A es su capacidad para generar particiones del conjunto de entrenamiento que se ajuste en un grado dado a las distintas clases posibles, introduciendo de esta forma un orden en dicho conjunto. El orden o el desorden (ruido) de un conjunto de datos es medible. El poder de clasificación de un atributo se mide de acuerdo a su capacidad para reducir la incertidumbre o entropía (grado de desorden de un sistema). Esta métrica se denomina Ganancia de información. El atributo con la más alta ganancia de información se escoge como el atributo que forme un nodo en el árbol.

La construcción de árboles de decisión, también denominados árboles de clasificación o de identificación, es sin duda el método de aprendizaje automático más utilizado.

El dominio de aplicación de los árboles de decisión no está restringido a un ámbito concreto sino que pueden ser utilizados en diversas áreas (desde aplicaciones de diagnóstico médico hasta juegos como el ajedrez o sistemas de predicción meteorológica).

El conocimiento obtenido en el proceso de aprendizaje se representa mediante un árbol en el cual cada nodo interior contiene una pregunta sobre un atributo concreto (con un hijo por cada posible respuesta) y cada hoja del árbol se refiere a una decisión (una clasificación). Un árbol de decisión puede usarse para clasificar un caso comenzando desde su raíz y siguiendo el camino determinado por las respuestas a las preguntas de los nodos internos hasta que se encuentre una hoja del árbol.

La construcción de los árboles de decisión se hace recursivamente de forma descendente (se parte de conceptos generales que se van especificando conforme se desciende en el árbol), por lo que se emplea el acrónimo TDIDT [*Top-Down Induction on Decisión Trees*] para referirse a la familia completa de algoritmos de este tipo.

La familia de algoritmos TDIDT abarca desde algoritmos ya clásicos de IA como CLS [*Concept Learning System*] hasta algoritmos optimizados como SLIQ o SPRINT, dos algoritmos desarrollados en el IBM Almaden Research Center que se usan en *Data Mining*. Los árboles de decisión se especificaran en la sección 3.3.

3.2.3. Reglas de asociación. Muchas veces en la vida real no se pueden construir modelos completos que permita una clasificación perfecta de todos los casos con los que se pueda encontrar. A veces hay que conformarse con descubrir modelos aproximados, los cuales contemplan algunas características de las distintas clases sin que el modelo abarque todas las clases posibles ni todos los casos particulares de una clase determinada.

La construcción de un modelo de clasificación completo puede no ser factible cuando se ha de tratar con una gran cantidad de atributos, cuando muchos valores son desconocidos, cuando unos atributos deben modelarse en función de otros o cuando el número de casos de entrenamiento es excesivamente elevado.

Los árboles de decisión no son muy adecuados para tratar con información incompleta (valores desconocidos en atributos de los casos de entrenamiento) y resultan problemáticos cuando unos atributos son función de otros. Las redes neuronales tampoco son apropiadas cuando se tiene información incompleta y, además, su entrenamiento puede llegar a consumir demasiado tiempo. Finalmente, las técnicas empleadas en ILP [*Inductive Logic Programming*] suelen ser muy poco eficientes.

Por su parte, un modelo de clasificación parcial intenta descubrir características comunes a los distintos casos de cada clase sin la necesidad de formar un modelo predictivo completo.

La extracción de reglas de asociación puede ser útil para resolver problemas de clasificación parcial donde las técnicas de clasificación clásicas no son efectivas.

Como se vio en el capítulo dedicado al uso de las reglas de asociación, el problema de la clasificación parcial se puede resolver usando reglas de asociación de dos formas diferentes: dividiendo el conjunto de casos de entrenamiento (un subconjunto por clase) o considerando la clase como un atributo más. En cualquiera de las dos situaciones anteriores, para cada regla de asociación A=>C obtenida ha de calcularse su riesgo relativo utilizando la siguiente expresión:

$$r(A \Rightarrow C) = \frac{P(C \mid A)}{p(C \mid -A)}$$

Cuando el cociente anterior es elevado se puede considerar interesante la regla $A \Rightarrow C$. Intuitivamente se puede comprender con facilidad el sentido de la ecuación que define el riesgo relativo de una regla. Por ejemplo, carecería de interés clasificar una enfermedad atendiendo a síntomas que no siempre se manifiestan asociados a esa enfermedad y sería trascendental identificar síntomas específicos de una enfermedad (aunque éstos sean muy poco frecuentes).

Se pueden construir modelos de clasificación híbridos basados, en mayor o menor medida, en reglas de asociación. Por ejemplo, ART [*Association Rule Tree*] que intenta aprovechar las mejores cualidades de las reglas de asociación como modelo de clasificación parcial con la construcción descendente de árboles de decisión.

3.2.4. Clasificadores basados en medidas de similaridad. Una forma básica de clasificar un caso es asignarle la misma clase que a otro caso similar cuya clasificación es conocida. Entre ellos destacan los métodos de clasificación por el vecino más cercano k-NN, donde k es impar (no tiene sentido probar con valores pares de k porque el error asociado a la regla k-NN es el mismo para 2x y 2x-1).

A la hora de construir clasificadores de este tipo han de resolverse algunas cuestiones previas, entre las que destacan:

¿Cómo se realiza la clasificación?: es posible asignarle a un caso la clase del caso almacenado más similar (*1-NN*) o utilizar los grados de similaridad con distintos casos almacenados a la hora de realizar la predicción (como en el método *k-NN*).

¿Qué casos deben almacenarse?: lo ideal sería almacenar aquellos casos típicos que recojan toda la información relevante necesaria para poder realizar una buena clasificación. Almacenar todos los casos conocidos haría muy ineficiente el funcionamiento del clasificador.

Los métodos de edición y condensado se utilizan para mejorar el rendimiento de este tipo de clasificadores. Los métodos de edición (como la edición de Wilson o el *Multiedit*) intentan eliminar los patrones mal etiquetados que puedan aparecer cerca de las fronteras de decisión. Por su parte, los métodos de condensado (como el algoritmo de *Hart*) procuran reducir el número de muestras del conjunto de entrenamiento sin que esto afecte a la calidad del clasificador construido.

Para reducir la complejidad computacional del problema se pueden emplear métodos de condensado o, simplemente, utilizar algoritmos optimizados como el de *Fukunaga* y *Narendra* para la obtención del vecino más cercano.

¿Cómo se mide la similaridad entre distintos casos?: cuando los atributos son numéricos, se suele calcular la similaridad entre casos utilizando alguna métrica de distancia (que es una medida de disimilaridad), como la distancia euclídea o la distancia de *Mahalanobis*. Por ejemplo, se puede utilizar la raíz cuadrada de las suma de los cuadrados de las diferencias de los valores de los atributos (usando factores de escala para que la influencia de todos atributos sea similar). Cuando los atributos son discretos, establecer una medida de similaridad es bastante más problemático.

Además, si hay atributos irrelevantes, se corre el riesgo de considerar muy diferentes casos que sólo difieren en los valores que toman atributos irrelevantes para la clasificación.

3.2.5. Técnicas estadísticas. Existen muchas técnicas estadísticas aplicables a problemas de clasificación. Estas técnicas suelen ser paramétricas. Se asume la forma del modelo y, a partir de los datos de entrenamiento, se hallan los valores adecuados para los parámetros del modelo.

Por ejemplo, un clasificador lineal asume que la clasificación puede realizarse mediante una combinación lineal de los valores de los atributos y emplea la combinación lineal que mejor se adapte al conjunto de casos de entrenamiento a la hora de clasificar nuevos casos.
En determinadas circunstancias, un clasificador cuadrático puede obtener mejores resultados que un clasificador lineal simple. Sin embargo, el ADC [Análisis Discriminante Cuadrático] requiere muchas más muestras de entrenamiento que el ADL [*Análisis Discriminante Lineal*] para obtener resultados similares ya que es más sensible al número de muestras requeridas.

Pero no siempre es mejor un clasificador cuadrático. Para determinados conjuntos de datos el ADC ni siquiera se puede aplicar, como sucede con un conjunto estándar de datos de la ionosfera formado por 351 patrones de 34 atributos cada uno (John Hopkins *University Ionosphere Database*). En estos casos, no se dispone de suficientes muestras para estimar la matriz de covarianza de los datos adecuadamente (de hecho, para los datos de la ionosfera ni siquiera se puede calcular su inversa).

Aunque en teoría el error de Bayes decrece conforme la dimensionalidad de los datos se incrementa, en la práctica se dispone de un conjunto fijo y finito de muestras para construir el clasificador (los estimadores están sesgados por las muestras disponibles). La bondad conseguida con un clasificador aumenta con la dimensionalidad de los datos hasta cierto punto, a partir del cual decrece conforme se incorporan nuevas variables (fenómeno de Hughes).

El problema anterior podría solucionarse consiguiendo más muestras de entrenamiento (lo cual no suele ser posible) o eligiendo un subespacio del espacio de patrones (usando técnicas de selección de características).

3.2.6. Redes neuronales. Las redes neuronales representan una de las aportaciones más importantes que las ciencias biológicas han realizado al campo de la Inteligencia Artificial. Su característica más importante es su capacidad de aprender a partir de ejemplos, que les permite generalizar sin tener que formalizar el conocimiento adquirido.

Con las redes neuronales se intenta expresar la solución de problemas complejos no como una secuencia de pasos, sino como la evolución de unos sistemas de computación inspirados en el funcionamiento del cerebro humano, los cuales no son sino la combinación de una gran cantidad de elementos simples de proceso (neuronales) interconectados que operan en paralelo.

Es importante destacar que, aunque se pueden desarrollar aplicaciones mediante programas de simulación, codificando algoritmos de funcionamiento y aprendizaje, la verdadera potencia de las redes neuronales se pone de manifiesto mediante su implementación física en hardware.

Una ventaja de las redes neuronales respecto a otros modelos de clasificación es que también pueden utilizarse para predecir valores reales y no sólo clases discretas.

Sin embargo, el modelo de clasificación construido al entrenar la red suele ser totalmente incomprensible para un experto humano (lo que dificulta que el clasificador goce de la confianza de los expertos). El conjunto de métodos LVQ (de aprendizaje por cuantificación vectorial) constituye una honrosa excepción al destacar por la sencillez de las heurísticas que utiliza.

LVQ: los métodos LVQ [*Linear-Vector Quantization*] son métodos de aprendizaje adaptativo basados en los mapas auto-organizativos [*SQM: Self-Organizing Maps*] de Kohonen. Se caracterizan por utilizar un número fijo y relativamente bajo de prototipos para aproximar las funciones de densidad de probabilidad de las distintas clases.

Dada una secuencia de observaciones vectoriales (patrones), se selecciona un conjunto inicial de vectores de referencia (codebooks o prototipos). Iterativamente, se selecciona una observación X y se actualiza el conjunto de prototipos de forma que case mejor con X.

Una vez finalizado el proceso de construcción del conjunto de prototipos (es decir, el entrenamiento de la red), las observaciones se clasificarán utilizando la regla a *1-NN* (buscando el vecino más cercano en el conjunto de vectores de referencia).

3.2.7. Algoritmos genéticos. Los algoritmos genéticos están inspirados en la Naturaleza, en la teoría de la evolución descrita por Charles Darwin en su libro "sobre el origen de las especies por medio de la selección natural". La hipótesis de Darwin (y de *Wallace*, que llegó a las mismas conclusiones independientemente) es que pequeños cambios heredables en los seres vivos y la selección son los dos hechos que provocan el cambio en la Naturaleza y la generación de nuevas especies. Fue Lendel quien descubrió que los caracteres se heredaban de forma discreta, y que se tomaban del padre o de la madre, dependiendo de su carácter dominante o recesivo. A estos caracteres que podían tomar diferentes valores se les llamaron genes, y a los valores que podían tomar, alelos. En los seres vivos, los genes están en los cromosomas.

En la evolución natural, los mecanismos de cambio alteran la proporción de alelos de un tipo determinado en una población, y se dividen en dos tipos: los que disminuyen la variabilidad (la selección natural y la deriva genética), y los que la aumentan (la mutación, la poliploidía, la recombinación o cruce y el flujo genético).

A principios de los 60, en la Universidad de Michigan en Ann Arbor, las ideas de John Holland comenzaron a desarrollarse y a dar frutos. En un libro escrito por un biólogo evolucionista, R.A. Fisher, titulado "La teoría genética de la selección natural", se pone de manifiesto que la evolución era una forma de adaptación más potente que el simple aprendizaje y tomó la decisión de aplicar estas ideas para desarrollar programas bien adaptados para un fin determinado. Los objetivos de su investigación fueron dos: imitar los procesos adaptativos de los sistemas naturales y diseñar sistemas artificiales (programas) que retengan los mecanismos de los sistemas naturales.

Los algoritmos evolutivos tratan de imitar los mecanismos de la evolución para resolver problemas. La aplicación de un algoritmo genético consiste en hallar de qué parámetros depende el problema, codificarlos en un cromosoma y aplicar los métodos de la evolución (selección y reproducción sexual con intercambio de información y alteraciones que generen diversidad). La mayoría de las veces una codificación correcta es la clave de una buena resolución del problema.

Los algoritmos genéticos en sí son métodos de optimización. En el algoritmo genético va implícito el método para resolver el problema. Un algoritmo genético es independiente del problema, lo cual lo hace robusto, por ser útil para cualquier problema, pero a la vez débil, pues no está especializado en ninguno.

Como método general de resolución de problemas que son, los algoritmos genéticos también pueden utilizarse para resolver problemas de clasificación.

Al igual que las redes neuronales y los clasificadores basados en medidas de similaridad, los clasificadores construidos utilizando algoritmos genéticos suelen destacar porque su rendimiento no se ve excesivamente afectado por la aparición errores en los casos de entrenamiento (lo que sí ocurre con determinados modelos simbólicos).

3.3 ÁRBOLES DE DECISIÓN

3.3.1 Introducción. Los árboles de decisión, se destacan por su sencillez y pueden utilizarse en diversas áreas, tales como: reconocimiento de señales de radar, reconocimiento de caracteres, censores remotos, sistemas expertos, diagnóstico médico, juegos, predicción meteorológica, control de calidad, etc. Se trata de un método para aproximar funciones de valores discretos, capaz de expresar hipótesis disyuntivas y robusto al ruido en los ejemplos de entrenamiento. Puede verse como la estructura resultante de la partición recursiva del espacio de representación a partir del conjunto (numeroso) de prototipos. Esta partición recursiva se traduce en una *organización jerárquica* del espacio de representación que puede modelarse mediante una estructura de tipo árbol. Cada *nodo interior* contiene una pregunta sobre un atributo concreto (con un hijo por cada posible respuesta) y cada *nodo hoja* se refiere a una decisión (clasificación).

Una característica importante de los Árboles de Decisión es su capacidad para convertir un proceso de decisión complejo en una colección de decisiones más simples, proporcionando una solución más fácil de interpretar. Un problema fundamental en la construcción de los árboles de decisión radica especialmente en hallar árboles que clasifiquen correctamente todo el conjunto objeto[3].

El despliegue grafico y la facilidad de interpretación pueden ser, la razón de la popularidad de los Árboles de Decisión, pero sus rasgos más importante son *naturaleza jerárquica* y su *flexibilidad.*

La naturaleza jerárquica de los árboles de decisión es uno de sus rasgos básicos. Su jerarquía por niveles lo convierte en un excelente clasificador de objetos. En cada nivel del árbol se encuentra una selección minuciosa del conjunto de objetos. La flexibilidad en su estructura le permite acoplarse fácilmente a cualquier cambio presentado en el conjunto de objetos de entrada. Esto lo convierte en una opción de análisis muy atractiva, por eso, su uso se recomienda en la realización de tareas bastante complejas. De hecho, pueden existir tareas que requieran métodos más tradicionales (Redes Neuronales, Redes Bayesianas, entre otros).

[3] El término **objeto** es esta sección hace referencia a registros, tuplas, hechos en una Base de Datos (BD). Se adopta el término objeto por su uso cotidiano en muchos artículos especializados en árboles de decisión. Un objeto esta formado por los posibles valores que toman los atributos en una tabla, dentro de una BD.

A continuación se presenta un estudio de los métodos para construir árboles de decisión y los problemas existentes, también sus ventajas potenciales, así como las estrategias para seleccionar los nodos que formaran el árbol [16].

3.3.2 El algoritmo básico. A continuación se hace una introdución las ideas fundamentales del denominado algoritmo *TDIDT (Top Down Induction of Decision Trees)* el cual puede ser contemplado como uniformizador de la mayoría de los algoritmos de inducción de árboles de clasificación a partir de un conjunto de datos conteniendo patrones etiquetados. El pseudocódigo del algoritmo TDIDT puede contemplarse en la figura 7.

Figura 7. Pseudocódigo del algoritmo TDIDT

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

end
endinf
End TDIDT

La idea subyacente al 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 mías informativa o la mías 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 [25].

3.3.3. Conceptos básicos. A continuación se presentan algunas definiciones necesarias en el proceso de contracción de los Árboles de Decisión [16], [31], [33].

- Un árbol de decisión esta compuesto por una dupla ***G = (N, A)***, donde ***N*** es el conjunto de **nodos** y ***A*** es el conjunto de **arcos.**

- Un camino en un árbol, es una sucesión de arcos de la forma $(a_1, a_2), (a_2, a_3), \ldots, (a_{n-1}, a_n)$. Entonces el camino de a_1 hasta a_n es de longitud ***n***.

- Los arcos son una tripleta de forma (n_i, n_j, e_{ik}) los cuales establecen una relación entre los nodos n_i, n_j de ***N***, en un sentido $(n_i \rightarrow n_j)$, y una etiqueta asociada (e_{ik}).

- El árbol debe cumplir las siguientes propiedades:

    1. Hay solamente un nodo **raíz**, al que ningún arco llega.

    2. Cada nodo excepto la **raíz** tiene solamente un arco de llegada

    3. Hay un único camino del nodo **raíz** a cada nodo.

- Si (n_k, n_z) son nodos en un árbol, entonces n_k es el padre de n_z y n_z es el hijo de n_k. Si hay un camino de n_k a n_z ($n_k \neq n_z$), entonces n_k es el predecesor de n_z y n_z es el antecesor de n_k.

- Un nodo sin sucesor, se le llama nodo **hoja** (o terminal), todos los demás nodos (excepto la raíz) se llaman nodos **interiores.**

- La *profundidad* de un nodo n_i en un árbol es la longitud desde el nodo raíz hasta el nodo n_i.

- La *altura* de un nodo n_i en un árbol es la longitud del camino más largo desde n_i a un nodo hoja. La altura del árbol es la altura del camino mas largo desde el nodo raíz hasta un nodo hoja, mas 1.

- El *nivel* de un nodo n_i en un árbol es la altura del árbol menos la profundidad del nodo n_i. el peso de un árbol es el número de elementos que contiene.

- Un árbol binario[4] es un árbol tal que:

    1. Cada hijo es distinto como hijo izquierdo o como hijo derecho.

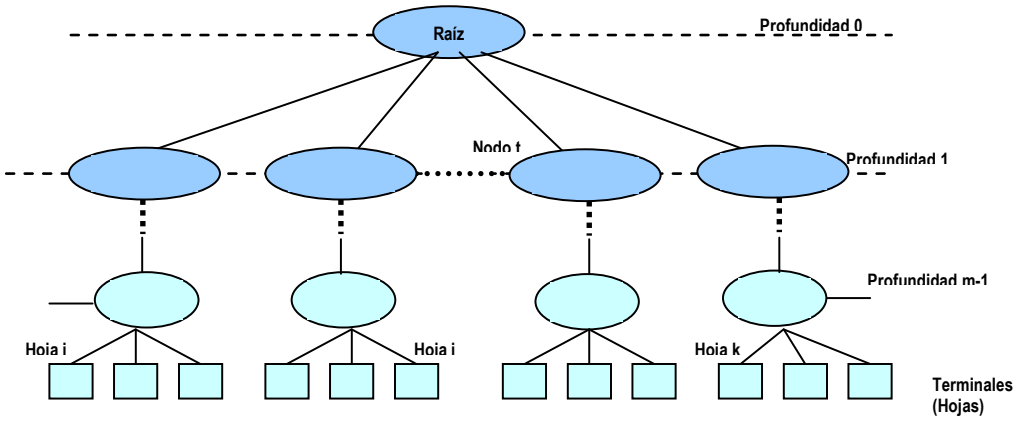
    2. Ningún nodo tiene más de un hijo izquierdo ni más de un hijo derecho.

- El balanceo de un nodo n_i en un árbol binario es $(1 + L) / (2 + L + R)$, donde L y R son el número de nodos en el subárbol derecho e izquierdo de n_i.

- Un árbol binario es ∞ - *balanceado,* $0 < \infty \leq 1$, si cada nodo esta balanceado entre ∞ y $1-\infty$. Un árbol binario ½ -*balanceado* es conocido como un árbol completo Figura 8.

[4] Aunque no vamos a hacer referencia especial a los árboles de decisión binarios, son mencionados por su especial conformación.

Figura 8. Ejemplo de un árbol de decisión



- Al número promedio de capas desde la raíz hasta un nodo *terminal* se le conoce como la **profundidad promedio** del árbol. El número promedio de nodos internos en a cada nivel del árbol, se conoce como la **extensión promedio** del árbol.

- En la tarea de formación del árbol interviene un **Conjunto de Objetos** (universo) $\boldsymbol{U} \neq 0$, este conjunto esta conformado por una dupla $S = \{T, V\}$, donde T representa un conjunto de **atributos** y V todos los posibles **valores** disjuntos que toman los atributos.

- Esta dupla se puede representar como una **tabla** cuyas *columnas* están conformadas por *atributos* y las *filas* por el conjunto de *valores* (usualmente pequeño) que toma cada atributo. Esta representación permite definir a S como una función $f: T \rightarrow V$, del conjunto de atributos al conjunto de valores.

- Los posibles valores que toman los atributos pueden clasificarse en:

    - **Nominales:** si toman valores que no tienen un orden natural, por ejemplo {soleado, nublado, lluvia}, {circulo, cuadrado, triangulo}, {rojo, verde, amarillo}.

    - **Categóricos:** si toman valores que expresan un juicio o decisión, por ejemplo {verdad, falso}, {si, no}, {oscuro, claro}.

    - **Numéricos:** el cuál se divide en:

        * **Discretos u ordinales,** por ejemplo {10, 20, 30, 40}, {pequeño, medio, grande}

        * **Continuos o intervalos** {3.1, 6.5, 7.4}

- Adicionalmente, cada **objeto** (fila en la tabla) del universo pertenece a un conjunto de **clases,** con las cuales se pueden identificar propiedades comunes de un conjunto de objetos y poderlos clasificar de acuerdo a un objetivo (*clases*) Tabla 1.

- En la tabla 1 se observa cada atributo con sus posibles valores, así como los valores de la clase *objetivo* (Si, No). El conjunto de atributos T se divide en:

  Atributos Condición: (Ej.: Dolor de Cabeza, Dolor Muscular, Temperatura), los cuales forman el conjunto objetivo a clasificar.

  Atributos Decisión o Clases: (Ej.: Gripa) el cual es usado para clasificar todo el conjunto objetivo.

Tabla 1. Conjunto de objetos (universo U)

	Atributos			*Clase*
Objetos	**Dolor de Cabeza**	**Dolor Muscular**	**Temperatura**	**Gripa**
Obj1	No	Si	Alta	Si
Obj2	Si	No	Alta	Si
Obj3	Si	Si	Media	No
Obj4	No	Si	Normal	Si

- Los nodos (***N***) de un árbol de decisión están formados por atributos de objetos y las hojas o terminales por etiqueta de la clase (*atributo decisión*). Los arcos (***A***) que unen un nodo con sus hijos están etiquetados con los posibles valores que toma el nodo (*atributos condición*) padre.

- La clasificación de un objeto mediante un árbol de decisión se hace siguiendo una ruta desde la raíz del árbol hasta un nodo hoja, tomando los arcos que corresponden a los valores que toma el objeto en el atributo que se está utilizando. La hoja donde finaliza el proceso contiene la etiqueta de la clase a la cual pertenece el objeto.

- El proceso de aprendizaje del árbol, es bueno si permite predecir satisfactoriamente las clasificaciones a las que pertenecen los objetos. En la construcción del modelo de clasificación se utilizan una buena cantidad de objetos. Este conjunto de objetos *U* es dividido en dos partes: los **datos de entrenamiento** L_1 y Los **datos de prueba** L_2, la partición del conjunto de objetos se realiza muchas veces randomicamente[5], L_1 por lo general toma 2/3 del conjunto y L_2 el resto 1/3:

[5] La palabra Random o randomicamente, expresa una escogencia al azar de algún objeto, situación, etc.

Datos de entrenamiento: estos datos son obtenidos especialmente del conjunto de objetos, sirven para realizar los ajustes de aprendizaje del algoritmo así como la verificación del conocimiento obtenido.

Datos de prueba: la metodología consiste en separar los datos de entrenamiento de los datos de prueba, para lograr de esta forma una mayor predicción y exactitud en la información final obtenida.

- El conocimiento obtenido es expresado en forma de **reglas de clasificación.** Estas **reglas** se expresan forma de un par de condición – acción {C ➔ A}, donde *C* es un conjunto de atributos condición y *A* es el atributo decisión o clase. El árbol representa una disjunción de conjunciones, donde cada camino desde la raíz hasta una hoja corresponde a una conjunción de atributos *C* y las hojas del árbol *A* a las clases. $C_1 = v_1 \wedge C_2 = v_2 \wedge C_3 = v_{3\ldots} \rightarrow A$, donde C_1, C_2, C_3…. son los atributos que se encuentran presente en cada nodo del árbol y v_1, v_2…. son los posibles valores que toma cada atributo, representados por los arcos en el árbol.

3.3.4. Potencialidades y problemas con los árboles de decisión. Los árboles de decisión son atractivos por las siguientes razones [31]:

- Las reglas de asignación son simples y legibles, por tanto la interpretación de resultados es directa e intuitiva.
- Es robusta frente a datos atípicos u observaciones mal etiquetadas.
- Es válida sea cual fuera la naturaleza de las variables explicativas: continuas, binarias nominales u ordinales.
- Es una técnica no parámetrica que tiene en cuenta las interacciones que pueden existir entre los datos.
- Es computacionalmente rápido.

Los posibles inconvenientes de un árbol de decisión son [31]:

- Las reglas de asignación son bastantes sensibles a pequeñas perturbaciones en los datos (inestabilidad)
- Dificultad para elegir el árbol óptimo
- Ausencia de una función global de las variables y como consecuencia pérdida de la representación geométrica.

- Los árboles de clasificación requieren un gran número de datos para asegurarse que la cantidad de las observaciones de los nodos hoja es significativa.

3.3.5. Diseño de un árbol de decisión. La construcción de un árbol de decisión se fundamenta principalmente en [16], [31], [33]:

1. Clasificar correctamente los datos de entrenamiento.

2. Generalizar[6] los datos de entrenamiento para que la información oculta pueda ser clasificada con una alta exactitud.

3. Facilitar la actualización del árbol con nuevos datos de entrenamiento.

4. Presentar la más simple estructura posible.

El diseño de un árbol de decisión se puede descomponer en las siguientes tareas:

1. Una apropiada elección de la estructura del árbol.

2. La elección de un subconjunto de objetos para ser usados en cada nodo interno.

3. La elección de las estrategias para ser usada en cada nodo interno.

Existen varios métodos propuestos para el diseño del árbol. Algunos de los criterios comunes para optimizar (heurística) el diseño del árbol son:

- La Tasa Mínima de Error

- Camino de Longitud Min-Max.

- Mínimo Número de Nodos en el Árbol.

- Camino de Longitud Mínima Esperada.

- Máximo Promedio de Ganancia de Información.

Un árbol óptimo es construido recursivamente a través de la aplicación de varias técnicas de programación. Los problemas básicos con algunos de los métodos de optimización sugeridos, son su ineficiencia computacional; cantidad normalmente grande de tiempo

[6] La generalización consiste en la transformación de datos a otros niveles de abstracción más altos. Los datos son organizados en forma jerárquica o en cierto orden parcial, la cual se utiliza para expresar conocimiento de alto nivel y facilitar el descubrimiento de conocimiento a múltiples niveles de abstracción.

computacional y almacenamiento requeridos. Esto en la práctica ha puesto una restricción en el tipo de árbol y en el número de atributos ha ser usados en cada nodo.

Existen varios métodos heurísticos para la contracción de árboles de decisión, los cuales se puede dividir en cuatro categorías:

Aproximación Bottom-Up, el árbol es construido usando el conjunto de entrenamiento. Usa algunas medidas de distancia, tal como *Distancia-Mahalonobis,* d*istancia PairWise*, se definen entonces unas clases, las cuales en cada etapa del árbol se unen las dos clases con la distancia más pequeña para formar un nuevo grupo. Estas son las mismas características de los métodos *Clustering*. En la construcción de un árbol de esta manera, las discriminaciones más obvias se hacen primero cerca de la raíz y más sutilmente en los niveles posteriores en el árbol.

Aproximación Top-Down, el diseño de un árbol de decisión se reduce a las siguientes tres tareas:

1. La selección de una regla de partición de un nodo.

2. La decisión a cerca de cual nodo es terminal.

3. La asignación de cada nodo terminal a una clase predefinida.

De las tres tareas anteriores, el problema de asignación de clases a cada nodo terminal es el más fácil. Para minimizar la porción de la mala clasificación, se asignan nodos terminales a las clases que tienen la más alta probabilidad. Estas probabilidades son normalmente estimadas a través del porcentaje de objetos de cada clase a ese nodo terminal, por el número total de objetos en ese específico nodo terminal.

Aunque se tenga un número pequeño de objetos y clases, la generación del posible árbol de decisión puede ser astronómicamente grande, lo que dos restricciones para reducir el espacio de búsqueda. Primero, **limitar el número de objetos** para ser usados en la generación del árbol. Segundo, encontrar un **criterio de partición**, para seleccionar los nodos que formaran el árbol.

La mayoría de las investigaciones realizadas en el diseño de los árboles de decisión se han concentrado en el área de las **reglas de partición de un nodo**; esto incluye naturalmente las **reglas de los nodos terminales.** Para esto los investigadores han intentado combinar el problema de diseño del árbol y el problema de selección de objetos, usando un criterio de partición apropiado en cada nodo interior.

La idea básica es elegir algún criterio de partición para cada nodo, esto hace que los datos o la información presente en los nodos hijos sea lo más pura posible (una

especie de filtro en cada nodo, para obtener una buena clasificación de universo de objetos). Una manera de lograr esta tarea, es definir una **función de partición (clasificación)** *i*(*n*) para cada nodo interior *n*. Suponiendo que para un nodo interior *n* hay un conjunto de objetos a clasificar *S*, el objetivo de la función de partición es, dividir el conjunto de objetos en ese nodo *n*, dando como resultado un nuevo nivel de hijos en el árbol. $\{n_{h1}, n_{h2}, n_{h3},\ldots, n_{hn}\}$, tal que una porción P_{d1} de casos *S*, son clasificado en n_{h1} y una porción P_{d2} en n_{h2} y así sucesivamente. Las investigaciones realizadas para encontrar la **función de partición** o clasificación de un nodo se fundamenta especialmente en las técnicas estadísticas, el análisis estadístico es fuente fundamental en la contracción óptima (clasificar correctamente el conjunto de objetos) del árbol.

Tabla 2. Definiciones básicas estadísticas

Símbolos	Significados
p(X)	Probabilidad de distribución de X
p(Y | X)	La distribución condicional de Y dado X
H(X)	La entropía de la variable randómica X. $H(X) = -\Sigma_x p(x)\log_2 p(x)$
H(X, Y)	Es la entropía join de la variable X y Y. $H(X, Y) = -\Sigma_x p(x,y) \log_2 p(x,y)$
H(Y | X)	La entropía condicional de Y dado X. $H(Y \mid X) = -\Sigma_x p(x)H(Y \mid X=x)$

Existen varias **funciones de partición** tales como, Index Gini, Sum-Minority, ldM Distance, ORT, X^2 (Quinlan 1986), Information Gain, Gain-Ratio.

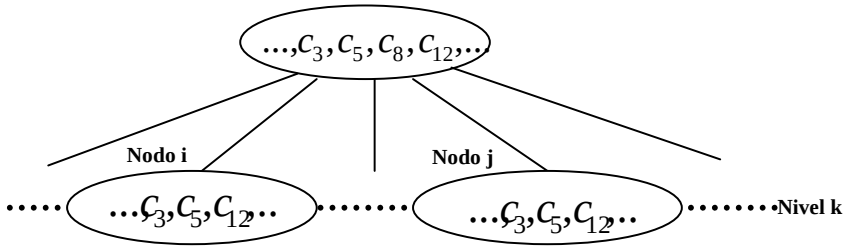
Tabla 3. Criterios utilizados en la selección de un nodo

Función de Partición	Definición
Information – Gain	$gain(A) = I(A,C) = H(C \mid A)$
Gain – Ration	$gain(A) / IV(A) = I(A,C) = H(A)$
LdM – Distance	$\frac{H(A \mid C) + H(C \mid A)}{H(C,A)} = 2 - \frac{H(A) + H(C)}{H(A,A)}$
Gini – Index	$\sum_a p(a) \sum_{ci \neq cj} p(c_i \mid a) p(c_j \mid a)$
X^2	$\sum_a \sum_c (np(a)p(c) - n(a,c))^2 / (np(a)p(c))$
Sum – Minority	$\sum_a n(a) - n(a,c)$
GORT	$\sum_{ai \neq aj} (p(a_i) + p(a_j))(n(a_i) / \mid n(a_i) \mid\mid n(a_i) \mid)$

El componente final en el diseño de un árbol de decisión Top-Down, es determinar cuando la función de partición debe detenerse. Se necesita un punto de parada en el diseño del árbol y una aproximación para seleccionar los nodos terminales en el

árbol. El problema con las funciones de la tabla 3 es que normalmente en las particiones del conjunto de objetos en un nodo *n*, algunas de ellas se detienen muy pronto en algunos nodos y demasiado tarde en otros.

Figura 9. Partición del conjunto de objetos en cada nodo



El diseño de árboles de decisión es bastante sensible a una variedad de reglas de particionamiento y esto es crucial en las reglas de terminación del árbol. Un método de parada propuesto muy atractivo es continuar las particiones hasta que todos los nodos terminales tengan objetos de solo una clase. Pero este resultado conlleva a generar árboles verdaderamente grandes (completos y complejos), los cuales no so óptimos para su análisis y para hacer descubrimientos de conocimiento.

Para contrarrestar el problema de generar árboles de gran magnitud, se puede realizar una selectiva ***poda*** en el árbol, logrando de esta manera, decrecer su magnitud, quedando un subárbol óptimo especialmente en su proceso de clasificación del conjunto de objetos. Se define entonces el proceso de ***poda*** de un árbol:

- Una rama a_t de un árbol T con un nodo raíz $n \in T$ consiste de un árbol n y de todos descendientes de *n* en *T*.
- ***Podar*** una rama a_t de un árbol T consiste en anular de T todos los descendientes de *n*. se denótale árbol podando como $T - T_n$.

- Si consecutivamente se podan ramas de T obteniendo el subárbol T_c, el cual es denotado como $T_c < T$.

En el proceso de diseño de un árbol complejo *T*, se define una medida de complejidad $R_\alpha(T)$ como:

$$R_\alpha(T) = R_\alpha(T) - \alpha |\overline{T}|$$

Donde $R(T)$ es una medida que estima la proporción de la mala clasificación del árbol $T, \alpha \geq 0$ es el costo de la medida de complejidad, y $\overline{T}$ es el conjunto de nodos terminales del árbol. En otras palabras, la complejidad del árbol *T* es medida por el número de nodos

terminales, y α es el costo de la complejidad del nodo. El árbol completo es denotado por $T_{\max}$. El objetivo del proceso de poda es encontrar para cada α un subconjunto $T(\alpha) \leq T_{\max}$ tal que:

$$R_\alpha(T(\alpha)) = \min_{T \leq_{T_{\max}}} R_\alpha(T)$$

Existen diferentes algoritmos de ***poda*** basado esencialmente en:

- ***Aproximación Botton-Up,*** los cual inician en los nodos terminales y la **poda** continua en ele árbol con cada rama viable.

- ***Aproximación Top-Down,*** inicia con los nodos superiores encontrando una medida de estimación la cual permite eliminar el subárbol subyacente de un nodo cualquiera en el árbol.

Aproximación Growing-Pruning: se basa en las siguientes ideas:

- Dividir el conjunto de datos en dos subconjuntos de tamaño aproximadamente iguales, construir el árbol iterativamente con un subconjunto y podarlo con el otro subconjunto.

- Consecutivamente intercambiar los roles de los dos subconjuntos

Aproximación híbrida: usan ambas aproximaciones Bottom-Up y Top-Down secuencialmente. Se Considera primero un conjunto de clases y se usa la aproximación Top-Down para la generación de los nodos del árbol, el árbol es terminado si contiene una sola clase, entonces se etiqueta con el nodo terminal.

3.3.6. Reglas de decisión. Una vez la estructura del árbol es diseñada y los subconjuntos de objetos a ser usados en cada nodo del árbol han sido seleccionados, una **regla de decisión** es extraída de cada rama del árbol, iniciando en la raíz y llegando hasta los nodos hojas (terminal). Cada rama en el árbol determina una regla de decisión, la cual esta especificada por una **condición**, representada por todos los nodos internos del árbol, y una **decisión** representada por las hojas del árbol.

1. d = 0 → 2
2. d = 1 ^ c = 0 ^ a = 2 ^ b = 0 → 2
3. d = 1 ^ c = 0 ^ a = 2 ^ b = 1 → 1
4. d = 1 ^ c = 0 ^ a = 0 → 1

5. d = 1 ^ c = 0 ^ a = 1 → 1

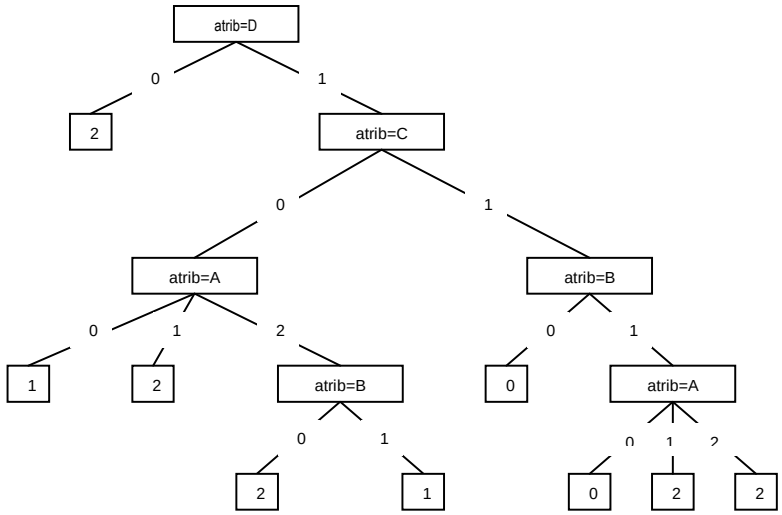
6. d = 1 ^ c = 1 ^ b = 0 → 0

7. d = 1 ^ c = 1 ^ b = 1 ^ a = 0 → 0

8. d = 1 ^ c = 1 ^ b = 1 ^ a = 1 → 2

9. d = 1 ^ c = 1 ^ b = 1 ^ a = 2 → 2

Figura 10. Generación de un árbol de decisión



En general la representación de una regla de decisión es una ***conjunción*** de valores de atributos. La primera rama del árbol de decisión de la Figura 10. es "***Si D*** es igual a 0" ***entonces*** decide la ***clase2*** en otro caso "***D*** es igual a 1" y continua con las ramas descendentes de este nodo.

3.4. APLICACIONES

Existen muchos algoritmos para árboles de decisión y las diferencias entre ellos están en la estrategia de podar los árboles, las reglas para particionar los árboles y el tratamiento de valores perdidos, entre ellos ID3, C4.5, C5.0, GID3*, O-Btree, ASSISTANT, CART, CHAID, SLIQ, SPRINT. A continuación se describen algunos algoritmos de Aprendizaje Automático que han sido utilizados con éxito en la Minería de Datos. Algunos de ellos son generales y pueden ser utilizados en varios dominios de conocimiento, mientras que otros fueron diseñados para dominios particulares [2].

- **Algoritmo ID3:** este sistema ha sido el que más impacto ha tenido en la Minería de Datos. Desarrollado en los años ochenta por Quinlan, ID3 significa *Induction Decision*

Trees, y es un sistema de aprendizaje supervisado que construye árboles de decisión a partir de un conjunto de ejemplos. Estos ejemplos son tuplas compuestas por varios atributos y una única clase. El dominio de cada atributo de estas tuplas está limitado a un conjunto de valores. Las primeras versiones del ID3 generaban descripciones para dos clases: positiva y negativa. En las versiones posteriores, se eliminó esta restricción, pero se mantuvo la restricción de clases disjuntas. ID3 genera descripciones que clasifican cada uno de los ejemplos del conjunto de entrenamiento.

Este sistema tiene una buena performance en un amplio rango de aplicaciones, entre las cuales se nombran, aplicaciones de dominios médicos, artificiales y el análisis de juegos de ajedrez. El nivel de precisión en la clasificación es alto. Sin embargo, el sistema no hace uso del conocimiento del dominio. Además, muchas veces los árboles son demasiado frondosos, lo cual conlleva a una difícil interpretación. En estos casos pueden ser transformados en reglas de decisión para hacerlos más comprensibles.

- **Algoritmo ID3 normalizado:** su objetivo es obtener un árbol de decisión mediante un proceso que sea independiente del número de valores de un atributo. Se consigue construyendo un árbol binario (siempre serán dos los valores que pueda tomar un atributo). Los inconvenientes son que crea árboles de decisión poco eficiente, debido a que pregunta varias veces por el mismo atributo, construye árboles menos legibles que ID3.

  Quinlan propuso normalizar la ganancia por un factor que nos indicase la cantidad de información necesaria para conocer el valor de un cierto atributo.

$$IV(X,A) = - Sv\ p(X,v) \log2\ p(X,v)$$

  Donde, p(X,v)= probabilidad de que x pertenezca a X, A(x)=v y v pertenece a V(A). La ganancia normalizada es:

$$GN(X,A) = \frac{G(X,A)}{IV(X,A)}$$

  El inconveniente es que para aquellos casos con denominador anormalmente bajo GN se hace muy grande, entonces la solución es aplicar GN sólo a aquellos atributos con G por encima de la media.

- **Algoritmo RLM:** desarrollado por R. López de Mántaras. Su objetivo es escoger aquel atributo que provoque una partición de X más próxima a la correcta.

  Partición correcta: todos los elementos de cada subconjunto de la partición son de la misma clase.

- **Algoritmos incrementales:** no parten de un conjunto fijo de instancias conocidas a priori. Permiten revisar el árbol y alterarlo para adaptarlo a nuevos ejemplos. Permite observar cómo evoluciona el árbol a medida que se le proporcionan instancias. Como los ID4, ID4R, ID5, ID5R.

- **Algoritmos ID4 e ID4R:** fueron la primera tentativa de construir un árbol de decisión de forma incremental. Desarrollados por Schimmer y Fisher. Toma como entrada un árbol de decisión y una entrada, y devuelve un nuevo árbol adaptado a ella. Mantiene información sobre el número de instancias en las clases para cada valor de cada atributo que pueda servir como decisión en un nodo. Calcula cuál es el mejor nodo para decidir, y si dicho nodo no es la raíz del subárbol tratado la se cambia por el nodo correspondiente; pero, ¿cómo cambiarlo? A). Expandiendo el nuevo atributo en sus valores; es decir, creando un solo nivel más (ID4). B). Reconstruyendo el árbol hasta el final utilizando ID3 (ID4R).

  Ventaja: es una mejora respecto a ID3, ya que ID3 para cada nueva instancia construye el árbol partiendo de cero.

  Inconveniente: para determinadas instancias el algoritmo da lugar a continuos descartes, no llegando a una estabilización final del árbol.

- **Algoritmos ID5 e ID5R:** desarrollados por Utgoff. Cuando se ha de reemplazar la raíz de un árbol no se descartan los subárboles ya existentes, sino que reestructuran aprovenchado el trabajo realizado cuando se crearon (técnica pull-up).

  Información que debe contener un árbol de decisión para aplicar estos algoritmos: Si es un nodo hoja, un nombre de una clase y el conjunto de instancias que discrimina. Si es un nodo de decisión, un nombre de atributo, con una rama hacia otro árbol de decisión para cada valor del atributo. Todos los posibles atributos del test en ese nodo, y contadores del número de instancias pertenecientes a cada clase para cada valor de dichos atributos.

- **Algoritmo ID3 para múltiples categorías:** versión más avanzada del ID3 que permite generar árboles para más de dos categorías, tratamiento de ruido, atributos lineales y valores perdidos.

- **Algoritmo IDL:** desarrollado por W. Van de Velde. Utiliza los tres operadores básicos de árboles de decisión: expansión, poda, transposición. Se utiliza en dos fases:

  1. Se utiliza la expansión al estilo ID3 para clasificar la nueva instancia. La medida de selección está basada en la distancia entre particiones de RLM.

2. Una vez obtenida la hipótesis discriminatoria, se revisa el camino de discriminación de la instancia pero desde la hoja a la raíz. Para ello utiliza el concepto de relevancia topológica: utiliza la transposición para rehacer el camino podando siempre que sea posible para obtener así el camino más corto.

- **Algoritmo C4.5:** el C4.5 es una extensión del ID3 que permite trabajar con valores continuos para los atributos, separando los posibles resultados en dos ramas: una para aquellos $A_i <= N$ y otra para $A_i > N$. Este algoritmo fue propuesto por Quinlan en 1993. El algoritmo C4.5 genera un árbol de decisión a partir de los datos mediante particiones realizadas recursivamente. El árbol se construye mediante la estrategia de profundidad-primero (*depth-first*). El algoritmo considera todas las pruebas posibles que pueden dividir el conjunto de datos y selecciona la prueba que resulta en la mayor ganancia de información. Para cada atributo discreto, se considera una prueba con n resultados, siendo n el número de valores posibles que puede tomar el atributo. Para cada atributo continuo, se realiza una prueba binaria sobre cada uno de los valores que toma el atributo en los datos.

- **Algoritmo AQ15:** el AQ15 fue desarrollado por Michalski. Es un sistema de aprendizaje inductivo que genera reglas de decisión, donde el antecedente es una fórmula lógica. Una característica particular de este sistema es la inducción constructiva (*constructive induction*), es decir, el uso de conocimientos del dominio para generar nuevos atributos que no están presentes en los datos de entrada.

  Al igual que el ID3, el AQ15 está diseñado para la generación de reglas fuertes, es decir, que para cada clase, se construye una regla que cubre todos los ejemplos positivos y ningún ejemplo negativo. El sistema soluciona el problema de los ejemplos incompletos o inconsistentes mediante un pre o post procesamiento. En el post procesamiento, además, se reduce de forma drástica la cantidad de reglas generadas mediante el truncamiento de reglas, el cual no afecta la precisión de las reglas obtenidas.

  AQ15 ha sido testeado en dominios médicos, como el diagnóstico en la limfografía, diagnóstico de cáncer de mama y la ubicación del tumor primario. En estos casos, se obtuvieron reglas con el mismo nivel de precisión que el de los expertos humanos. En todos los casos, los datos de entrenamiento son conjuntos chicos, de unos cientos de ejemplos.

- **Algoritmo SLIQ:** el algoritmo SLIQ (Supervised Learning In Quest) fue desarrollado por el equipo Quest de IBM. Este algoritmo utiliza los árboles de decisión para clasificar grandes cantidades de datos. El uso de técnicas de pre-ordenamiento en la etapa de crecimiento del árbol, evita los costos de ordenamiento en cada uno de los nodos. SLIQ mantiene una lista ordenada independiente de cada uno de los valores de los atributos continuos y una lista separada de cada una de las clases. Un registro en la

lista ordenada de atributos consiste en el valor del atributo y un índice a la clase correspondiente en la lista de clases. SLIQ construye el árbol de forma ancho-primero (breadth-first). Para cada uno de los atributos busca en la lista correspondiente y calcula los valores de entropía para cada uno de los nodos de la frontera simultáneamente. A partir de la información obtenida se particionan los nodos de la frontera, y se expanden para obtener una nueva frontera.

Aunque SLIQ trabaja con datos que pueden estar en disco mientras se ejecuta el algoritmo, necesita que cierta información resida en memoria permanentemente durante la totalidad de la ejecución del mismo. Dicha información crece proporcionalmente a la cantidad de registros de entrada, lo cual limita en gran medida la cantidad de registros de entrenamiento. Para solucionar este problema el equipo de desarrollo del Quest, ha desarrollado otro algoritmo de clasificación basado en árboles de decisión: el SPRINT (*Scalable PaRallelizable INduction of decision Trees*). El SPRINT elimina todas las restricciones de memoria presentes en el SLIQ.

- **Algormitmo CHAID:** significa "Chi-square automatic interaction detection", fue introducido por Kass (1980) y es un derivado del THAID: A sequential search program for the analysis of nominal scale dependent variables (Morgan and Messenger, 1973). Está disponible como un modulo del paquete estadístico SPSS. El criterio para particionar está basado en c2 y para terminar el proceso se requiere definir de antemano un "threshold". (umbral).

- **Algoritmo CART:** introducido por Breiman, et al (1984), propiamente es un algoritmo de árboles de decisión binario. Existe una versión similar llamada IndCART y que está disponible en el paquete.

- **Algoritmo C4.5 oblique C4.5 es un algoritmo en dos pasos:** el primer paso consiste en aplicar una función de discriminación lineal. El conjunto de datos se expande con la inserción de nuevos atributos, generados por la proyección de cada ejemplo sobre los hiperplanos de la función discriminante. El segundo paso inductivo lo realiza el C4.5 o el C5.0. Si uno de los nuevos atributos es elegido para un nodo, se crea una superficie de decisión oblicua.

  Una función de discriminación lineal es una combinación lineal de atributos, donde la suma de los cuadrados de las diferencias entre las medias de las clases es máxima respecto a la varianza de las clases.

A continuación se presentan los algoritmos que se implementaron en el desarrollo del proyecto. *C4.5, SLIQ y Mate-Tree.*

3.5. ALGORITMO C4.5

El algoritmo C4.5 fue pensado para los casos en los que se tiene gran cantidad de objetos y atributos y se requiere la construcción de un árbol de clasificación sin mucho tiempo computacional. La estructura básica del algoritmo es iterativa. Se divide el conjunto de datos en subconjuntos de entrenamiento y prueba, como resultado se obtiene un árbol capaz de clasificar todos los elementos de conjunto de entrenamiento, entonces el proceso termina, en caso contrario, se añaden los objetos no clasificados nuevamente a los datos de entrenamiento y el proceso continúa. El problema por tanto radica en los criterios para el cálculo del árbol para una cantidad arbitraria de objetos. El punto crucial en el cálculo del árbol está en la elección del mejor atributo. C4.5 utiliza la **entropía de Shannon** para esta decisión y se demuestra que el costo computacional por cada iteración en C4.5 es proporcional al producto del tamaño del conjunto de entrenamiento, el número de atributos y el número de nodos que no son hojas del árbol. C4.5 consta de un sistema de aprendizaje que obtiene reglas de clasificación y que utiliza árboles como modelo de representación [13], [31].

El núcleo del algoritmo C4.5 es el proceso de construcción del árbol de decisión. La construcción de este árbol se puede hacer de manera recursiva. Los pasos básicos del algoritmo son:

- Calcular la ganancia de información de cada atributo.
- Seleccionar el atributo que da la mayor ganancia de información y utilizarlo como el elemento de decisión en el nodo en el que se encuentre el árbol.
- Ramificar el nodo con todos sus posibles valores.
- Reagrupar los datos en las ramas correspondientes.
- Repetir el proceso para cada rama del árbol.

El algoritmo finaliza cuando todos los objetivos en cada nodo pertenecen a la misma clase. Los detalles precisos de la construcción del árbol varían según las distintas implementaciones.

El punto critico de esta estrategia de generación de árbol, es la selección del atributo partición de cada nodo. El proceso de selección de cada atributo se fundamenta en una función heurística que minimiza la medida de entropía aplicada a los ejemplos en un nodo. Esta función heurística permite seleccionar el atributo que provee la mayor ganancia de información (Información Gain), es decir, el atributo que minimiza la información necesaria en los subárboles resultantes para clasificar los elementos. La medida de entropía favorece los atributos que particionan los datos en subconjuntos que tienen baja entropía de

clase. Un subconjunto de datos tiene una baja entropía de clase cuando la mayoría de ejemplos pertenecen a una sola clase, C4.5 elige el atributo que posee el grado máximo local de discriminación entre clases [31].

3.5.1. Entropía. A mediados de los años 40's Shannon publica su articulo "Teoría de la información de sistemas de comunicaciones", consiguió determinar con unidades el concepto de información. Creando así la medida de "Cantidad de Información". Esto conduce a expresar que la medida de la información está relacionada con la Incertidumbre.

La teoría de la información no representa una herramienta para manipular la incertidumbre, pero si para medirla. Desde este punto de vista, la cantidad de conocimiento (información) y la perdida de esta (incertidumbre) se complementan.

La incertidumbre inherente a la solución de un problema proporciona información deficiente, la cual puede ser imprecisa, incompleta o en alguna forma contradictoria. La incertidumbre simboliza la incompletitud e inexactitud del conocimiento sobre las características del ambiente. Si se puede medir la cantidad de incertidumbre inherente a una situación y reducirla hasta obtener información relevante, entonces la cantidad de información que se obtiene mediante este proceso puede cuantificar una medida, la cual disminuye la incertidumbre resultante.

Cuando existen varios espacios de probabilidades, a cada estado se le asocia una probabilidad, es decir, a una variable aleatoria se le asocia su espacio de probabilidades. Se define entonces la C*antidad de Información* de un estado i como:

$$I_i = \log(p_i)$$

Siendo P_i la probabilidad asociada al estado *i*. el signo menos es para hacer que I_i siempre sea positivo, ya que P_i siempre estará entre 0 y 1, y su logaritmo es negativo. Con la suma de todas las cantidades de información de todos los estados posibles queda:

$$H(S) = \sum_{i=1}^{n} P(s_i) \log_2 (1/P(s_i))$$

La formula de Entropía utiliza el **$\log_2$** (logaritmo en base 2) como medio para contrarrestar el carácter exponencial de los estados posibles y porque la entropía es una medida de la codificación en bits.

3.5.2. Ganancia de información. El proceso de selección de los atributos en el algoritmo **C4.5**, se fundamenta en una ***heurística (Information Gain)***[7] que minimiza la medida en ***entropía*** aplicada a los ejemplos en un nodo. Esta ***heurística*** permite seleccionar el atributo que provee la mayor ***ganancia de información,*** es decir, el atributo que minimiza la información necesaria en los subárboles resultantes para clasificar los elementos. La medida de entropía favorece los atributos que particionan los datos en subconjuntos que tienen baja entropía de clase. Un subconjunto de datos tiene baja entropía de clase cuando la mayoría de ejemplos pertenecen a una sola clase. **C4.5** básicamente elige el atributo que posee en grado máximo local de discriminación entre clases.

La ***Ganancia de información*** utiliza el ***criterio estadístico del valor esperado*** donde se desea ***minimizar o maximizar*** el costo esperado. En términos de la teoría de la información, la Ganancia se denota por ***Gain(D, A)***, la cual representa la diferencia entre la cantidad de información de D, la cual se simboliza por *H(D)*, y la cantidad de conocimiento de los valores A, H(D/A). La Gain(D, A) se calcula matemáticamente.

$$Gain(D, A) = H(D) - H(D / A)$$

Donde,

$$H(D) = -\sum_{i=1}^{n} p_i * \log_2 p_i$$

Entropía de *D*, donde p_i es la proporción de *D* perteneciente a la clase *i*. Y

$$H(D / A) = -\sum_{i=1}^{n} \frac{|A_i|}{|D|} * H(A_i)$$ Entropía de *D* es con respecto a *A*

El objetivo central del algoritmo **C4.5** es seleccionar al atributo a probar en cada nodo del árbol. Se debe seleccionar el atributo que tenga mayor clasificación sobre el conjunto de objetos de decisión *S*. **C4.5** usa la medida de **ganancia de información** para seleccionar el atributo candidato en cada etapa mientras va creciendo el árbol. La *Ganancia de Información, Gain(S,A)* de un atributo A, con respecto al conjunto de objetos *S*, es definido como:

$$Gain(S, A) = Entropia(s) - \sum_{v \in Valores(A)} \frac{|S_v|}{|S|} Entropia(S_v) \qquad (1)$$

Donde Valores(A) es el conjunto de todos los posibles valores para el atributo A, y S_v es el subconjunto de S para el atributo A que toma valores $v, S_v = \{s \in S \mid A(s) = v\}$. El primer termino de la ecuación (1) es la entropía del conjunto S, y el segundo término es

[7] La heurística usada en el algoritmo C4.5 es la **"Ganancia de información",** se fundamenta en la heurística **"Hill-Climbing"**, el cual se trata de un ciclo que continuamente se desplaza en la dirección de un valor ascendente**,** es decir, busca máximos locales.

simplemente la suma de la entropía de cada subconjunto S_v, ponderando la fracción de objetos $\frac{|S_v|}{|S|}$ que pertenece a S_v.

3.5.3. Ejemplo del algoritmo C4.5. Sea un conjunto de datos de entrenamiento (Tabla 4.), el cual es representado en la siguiente tabla:

Tabla 4. Datos de entrenamiento para C4.5

Estado	Humedad	Viento	Jugar-Tenis
Soleado	Alta	Débil	No
Soleado	Alta	Fuerte	No
Nublado	Alta	Débil	Si
Lluvia	Alta	Débil	Si
Lluvia	Normal	Débil	Si
Lluvia	Normal	Fuerte	No
Nublado	Normal	Fuerte	Si
Soleado	Alta	Débil	No
Soleado	Normal	Débil	Si
Lluvia	Normal	Débil	Si
Soleado	Normal	Fuerte	Si
Nublado	Alta	Fuerte	Si
Nublado	Normal	Leve	Si
Lluvia	Alta	Fuerte	Si

Con el conjunto de datos de entrenamiento, se busca clasificar correctamente la ***clase decisión (jugar-Tenis),*** el cual toma valores ***si, no***. Esta clasificación se fundamenta en los atributos condición.

Sea S la clase decisión jugar-Tenis, formado por 14 objetos los de los cuales hay 9 objetos *positivos (Si)* y 5 objetos *negativos (No),* se adopta la notación [9+, 5-] para simplificar el proceso de expresión.
La Entropía de S es:

$$Entropia([9+,5-]) = H(S) = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right) = 0.940$$

A continuación se calcula la *Ganancia de Información* de los atributos condición con respecto a la clase decisión.
Inicialmente el atributo viento:

Valores(Viento) = Débil, Fuerte:

$S = [9+,5-]$

$$S_{Debil} = [6+,2-]$$
$$S_{Fuerte} = [3+,3-]$$

$$Gain(S, Viento) = Entropia(S) - \sum_{v \in \{Debil, Fuerte\}} \frac{|S_v|}{|S|} Entropia(S_v)$$

$$= Entropia(S) - \left(\frac{8}{14}\right) Entropia(S_{Debil}) - \left(\frac{6}{14}\right) Entropia(S_{Fuerte})$$

$$= 0.940 - \left(\frac{8}{14}\right) 0.811 - \left(\frac{6}{14}\right) 1.00$$

$$= 0.048$$

Ahora se calcula la ganancia para el atributo Estado:

Valores(Estado) =Soleado, Nublado, Lluvioso:

$$S = [9+,5-]$$
$$S_{Soleado} = [2+,3-]$$
$$S_{Nublado} = [4+,0-]$$
$$S_{Lluvioso} = [3+,2-]$$

$$Gain(S, Estado) = Entropia(S) - \sum_{v \in \{Soleado, Nublado, Lluvioso\}} \frac{|S_v|}{|S|} Entropia(S_v)$$

$$= Entropia(S) - \left(\frac{5}{14}\right) Entropia(S_{Soleado}) - \left(\frac{4}{14}\right) Entropia(S_{Nublado}) - \left(\frac{5}{14}\right) Entropia(S_{Lluvioso})$$

$$= 0.940 - \left(\frac{5}{14}\right) 0.97 - \left(\frac{4}{14}\right) 0.00 - \left(\frac{5}{14}\right) 0.97$$

$$= 0.246$$

El proceso continua igual con el resto de atributos presentes en la Tabla 4. La Ganancia de Información de todos los atributos presentes es:

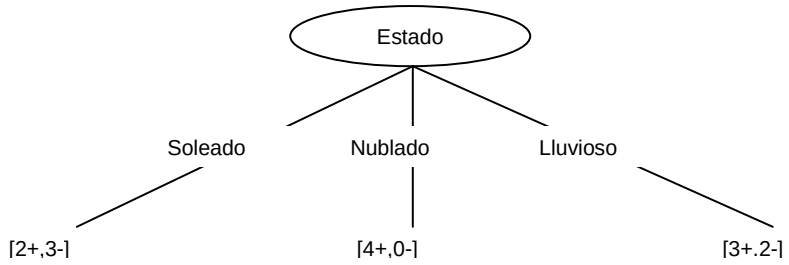
Gain(S, Estado) = 0.246
Gain(S, Humedad) = 0.151
Gain(S,Viento) = 0.048

C4.5 determina la ganancia de información para cada atributo candidato (Estado, Humedad, Viento), entonces selecciona el que posee la ganancia más alta. Acorde a la medida de ganancia de información, el atributo ***Estado*** provee la mejor predicción del atributo objetivo (jugar-Tenis), sobre los datos de entrenamiento. Estado es seleccionado como el

atributo decisión para el nodo raíz del árbol y las ramas son creadas debajo de la raíz para cada posible valor del atributo *(Soleado, Nublado y Lluvioso)*

El resultado de la partición del árbol de decisión es mostrado en la figura 11. con los datos de entrenamiento ordenados para cada nuevo nodo descendente. Para cada objeto donde *Estado = Nublado,* se presenta una clasificación positiva sobre la *clase decisión,* el nodo esta conformado por un nodo hoja en el cual la clasificación es J*ugar-Tenis Si* y el valor de la entropía es *cero.* En contraste con los atributos nodo descendientes, E*stado = Soleado* y *Estado = Lluvioso,* donde el valor de entropía no es *cero,* y un nuevo árbol debe ser generado por debajo de estos nodos.

Figura 11. Construcción del primer nivel del árbol

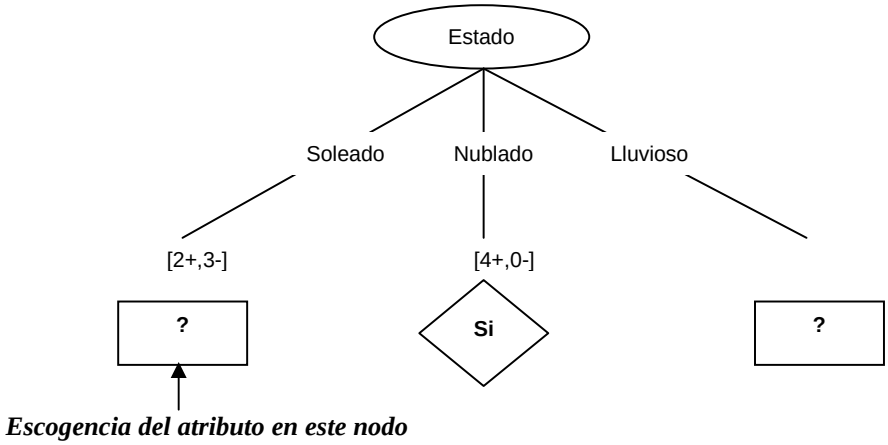


El proceso de selección de un nuevo atributo y el particionamiento de los datos de entrenamiento es nuevamente repetido para cada nodo descendiente no terminal, a ese nivel sólo se usan los datos de entrenamiento asociados con este nodo. Los atributos que son incorporados en al parte de arriba del árbol son excluidos. En el camino desde la raíz del árbol hasta un nodo hoja sólo pueden aparecer atributos que no se repitan. Este proceso continúa para cada nuevo nodo hoja hasta que se cumplan dos condiciones necesarias:

- Cada atributo haya sudo incluido en la construcción de árbol ó
- Los datos de entrenamiento asociados con un nodo hoja, tengan todos los mismos valores del atributo *(entropía Cero).*

En la Figura 12. se ilustra el proceso de escogencia de un nuevo nodo, para esto hay que calcular nuevamente la ganancia de información para la próxima etapa de crecimiento del árbol de decisión.

Figura 12. Selección de un nuevo atributo para C4.5



Calcula el valor de Ganancia a la rama *Estado=Soleado,* para crear el nodo y así poder generar el subárbol.

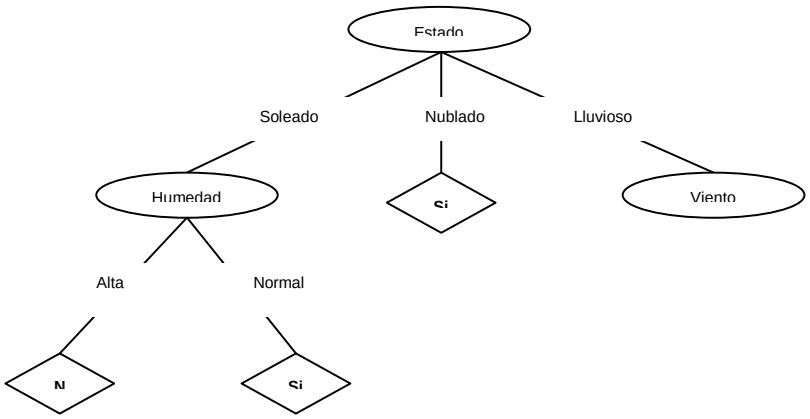
$$Gain(S_{Soledo}, Humedad) = 0.970 - \left(\frac{3}{5}\right)0.0 - \left(\frac{2}{5}\right)0.0 = 0.970$$

$$Gain(S_{Soledo}, Viento) = 0.970 - \left(\frac{2}{5}\right)0.0 - \left(\frac{3}{5}\right)0.918 = 0.019$$

El atributo con mayor Ganancia de Información es *Humedad* por lo cual será anexado al árbol quedando como se ve en la Figura 13.

Para continuar con la clasificación de los datos de entrenamiento, se calcula de la forma anterior la ganancia para *Estado=Lluvioso.*

Figura 13. Subárbol

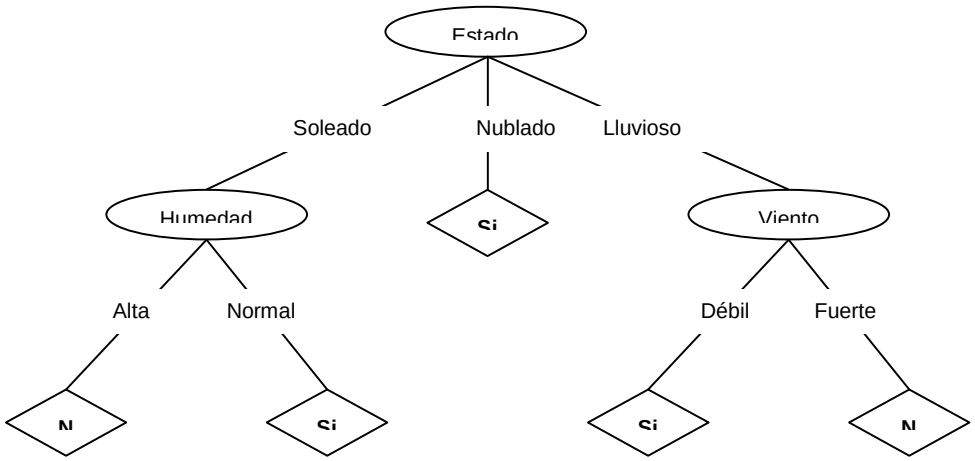


$$Gain(S_{Lluvioso}, Humedad) = 0.970 - \left(\frac{2}{5}\right)1.0 - \left(\frac{3}{5}\right)0.917 = 0.0198$$

$$Gain(S_{Lluvioso}, Viento) = 0.970 - \left(\frac{3}{5}\right)0.0 - \left(\frac{2}{5}\right)0.0 = 0.970$$

El atributo con mayor Ganancia de Información es *Viento* por lo tanto se anexa al árbol quedando como se ve en la Figura 14.

Figura 14. Árbol de decisión del ejemplo



Las reglas deducibles a través del árbol de decisión del ejemplo son:

$$Estado = Soelado \wedge Humedad = Alta \rightarrow No$$
$$Estado = Soelado \wedge Humedad = Norma \rightarrow Si$$
$$Estado = Nublado \rightarrow Si$$
$$Estado = Lluvioso \wedge Viento = Débil \rightarrow No$$
$$Estado = Lluvioso \wedge Viento = Fuerte \rightarrow Si$$

3.5.4. Especificaciones del algoritmo C4.5. C4.5 (Datos de Entrenamiento, Clase – Decisión, Atributos)

Como ya se ha mencionado los datos de Entrenamiento, son los objetos utilizados en la construcción, es el conjunto de valores utilizados para predecir el árbol. Clase-Decisión, es

el conjunto de valores utilizados para predecir el árbol. Atributos, es la lista de los atributos condición que pueden ser probados para el aprendizaje del árbol de decisión. Este retorna un árbol de decisión que correctamente clasifica los objetos [13].

- Crear un nodo raíz para el árbol.
- Si todos los objetos son positivos, retorne un simple nodo raíz, con etiqueta = +.
- Si los atributos son negativos, retorne un simple nodo raíz, con etiqueta = -.
- Si los atributos son vacíos, retorne un simple nodo raíz del árbol, con etiqueta = al valor más común de la Clase-Decisión.
- En otro caso se inicia. Figura 15.

Figura 15. Algoritmo C4.5

- A ← El atributo de los atributos que mejor clasifique los objetos (Ganancia de Información).

El atributo decisión para la raíz ← A.

- Para cada posible valor de v_i, de A,
- Adicione una nueva rama al árbol perteneciente a la raíz, correspondiente a la prueba A = v_i
- Buscar en los objetos$_i$ del subconjunto de objetos que tengan valor v_i para A.
- Si los objetos$_{v_i}$ es vació.
  - Entonces esta nueva rama adicionarla al nodo hoja con etiqueta = al más común valor de la Clase-Decisión
  - Sino pertenece esta nueva rama adiciónela al subárbol.
    C4.5 (Datos_entrenamiento, Clase-Decisión, Atributos-{A})

3.5.5. La poda del árbol. C4.5 efectúa la poda después de haber desarrollado el árbol completo, a diferencia de otros sistemas que realizan la construcción del árbol y la poda a la vez; es decir, estiman la necesidad de seguir desarrollando un nodo aunque no posea el carácter de hoja. Por cualquiera de los métodos se obtienen árboles en los que las hojas pueden cubrir ejemplos de más de una clase y que, por tanto, cometerán errores de clasificación sobre el conjunto de ejemplos de entrenamiento. Precisamente, la estimación del error de clasificación de una hoja, no ya sobre el conjunto de entrenamiento sino sobre ejemplos posteriores a la fase de aprendizaje, decidirá la poda en C4.5. Dicha estimación se efectúa mediante técnicas no muy ortodoxas desde el punto de vista de la Estadística; como el mismo Quinlan comenta [2]: ...constituyen un gran grano de sal. Es destacable en el proceso un nivel de confianza que por defecto el sistema toma tan solo del 25%, por ser el nivel que mejores resultados proporciona por lo general, aunque permita modificarlo a la hora de ejecutar el programa. El valor de este porcentaje está en relación inversa al grado de la poda que determinará. La poda se decide por comparación del error estimado para un conjunto de hojas, hijas todas de un mismo nodo, y el error estimado para el padre en caso

de podar sus hojas. Decidida la poda de las hojas se repite la operación en un nivel superior; así hasta que la poda no proceda.

Sea un cierto nodo hoja que cubre N ejemplos y de ellos E errores. La probabilidad de error de reescritura de la hoja puede aproximarse por el cociente *E/N*, pero este cociente no es adecuado para estimar la probabilidad de error de la hoja en la clasificación de ejemplos que no pertenezcan al conjunto de entrenamiento. Sin embargo, la probabilidad de error sobre esos otros ejemplos seguirá una distribución de probabilidad que Quinlan resume mediante el intervalo de confianza de la distribución binomial, para el nivel de confianza *CF* y que por defecto es, como ya se dijo, del 25%. La probabilidad de error de la hoja sobre ejemplos no utilizados en el entrenamiento se identifica con el límite superior de ese intervalo de confianza, $S_{CF}(E,N)$, con el argumento de que el árbol fue construido para minimizar el error. Para simplificar la última fase de este proceso, se supone que se trabaja con un conjunto de *N* ejemplos nuevos; es decir, el número de ejemplos de entrenamiento que clasificaba la hoja. Así, el número de errores previstos para la hoja será: $N \times S_{CF}(E,N)$. El número de errores previstos para todas las hojas; es decir el subárbol que cuelga de su padre, es la suma de los errores previstos para las hojas. Este último valor se comparará con el número de errores previstos para el padre una vez convertido en hoja al podar provisionalmente todos sus hijos. La comparación de los dos valores decidirá la poda final.

3.5.6. Medidas alternativas para la selección de atributos. Existe un sesgo natural en la medida de ganancia de información, el cual favorece atributos con muchos valores, sobre aquellos que tienen poco valores. Por ejemplo, un atributo fecha, tendráia mayor ganancia de información que cualquiera de los atributos en nuestro ejemplo. Esto se debe a que este atributo predice perfectamente el valor del atributo objetivo. El problema es que este atributo tiene tantos valores distintos que tiende a separar perfectamente los ejemplos de entrenamiento en pequeños subconjuntos, que se ajustan al concepto buscado. Por esto, el atributo fecha tiene una ganancia de información elevada, a pesar de ser un predictor pobre [16].

Una solución a este problema es usar una métrica alternativa a la ganancia de información. Quinlan, propone una medida alternativa que ha sido usada con éxito, gain ratio. Esta métrica penaliza atributos como fecha incorporando un término conocido como split information, que es sensible a qué tan amplia y uniforme es la partición que un atributo induce en los datos:

$$splitInformation(S, A) = -\sum_{i=1}^{c} \frac{|S_i|}{S} \log_2 \frac{|S_i|}{S}$$

Observen que este término es la entropía de S con respecto al atributo A. La medida gain radio está definida como:

$$gainRation(S, A) = \frac{gain(S, A)}{splitInformation(S, A)}$$

Un problema práctico con este enfoque es que el denominador de esta medida puede ser 0 o muy pequeño, si |Si| ≈ |S|, lo cual hace que la medida sea indefinida para atributos que tienen casi el mismo valor para todos los ejemplos.

3.5.7. La obtención de reglas a partir del árbol. El conocimiento obtenido es expresado en forma de **reglas de clasificación.** Estas **reglas** se expresan en forma de un par de condición – acción $\{C \Rightarrow A\}$, donde *C* es un conjunto de atributos condición y *A* es el atributo decisión o clase. El árbol representa una disjunción de conjunciones, donde cada camino desde la raíz hasta una hoja corresponde a una conjunción de atributos *C* y las hojas del árbol *A* a las clases. $C_1 = v_1 \wedge C_2 = v_2 \wedge C_3 = v_3 ... \Rightarrow A$, donde $C_1, C_2, C_3 \cdots$ son los atributos que se encuentran presente en cada nodo del árbol y $v_1, v_2, \cdots$ son los posibles valores que toma cada atributo, representados por los arcos en el árbol.

Es este un proceso largo y complejo. C4.5 extrae reglas del árbol que obtiene previamente a aplicar la poda. Es este un proceder que no es mucho más complejo que los sistemas que extraen las reglas a la vez que elaboran el árbol podado y que produce buenos resultados, argumenta Quinlan.

La lectura de cada camino de la raíz a una hoja en el árbol previo a la poda produce una regla en forma normal conjuntiva, normalmente demasiado específica. Con la familia de reglas resultante de la lectura de todos los caminos se procede en tres fases bien diferenciadas:

- *La cualificación* de las reglas, que consiste en eliminar algunos valores de atributo en el antecedente de cada regla, con lo que se convierten en más generales. Se eliminarán aquellos atributos cuya ausencia del antecedente no signifique mayor merma en la fiabilidad futura de la regla.

La eliminación de atributos se realiza de forma secuencial. Valorado el rendimiento de la regla de partida R y el de la obtenida R^- tras la eliminación de un cierto atributo A en ella, se decide su eliminación definitiva. Si hay pocos atributos se prueba con todos y se elimina el que mejor comportamiento conceda a la regla resultante. Si hay muchos atributos en el antecedente se les ordena por medio de un heurístico y se respeta el orden en el estudio de su posible eliminación; el proceso algorítmico correspondiente es voraz; sin vuelta atrás. Para la valoración de las reglas R y R^- se comparan las probabilidades de error estimadas para cada una; equiparando esas probabilidades con los límites superiores de los intervalos de confianza calculados de forma similar a como se hizo en la poda de árboles.

- *La selección* de una familia de reglas. El conjunto de reglas obtenido en la fase anterior normalmente es redundante como recubrimiento del conjunto de ejemplos. En esta fase se trata de eliminar aquellas reglas peores pero dejando como remanente una familia de reglas suficiente para cubrir a la mayoría de los ejemplos.

La selección se efectúa siguiendo el principio de mínima descripción (Rissanen 1978). En líneas generales consiste en considerar la familia inicial de reglas como una teoría y sus errores como excepciones a la teoría y tratar de buscar una subfamilia que optimice la suma de las codificaciones de la teoría correspondiente y de las excepciones. En las codificaciones de las reglas y de las excepciones hay que realizar algún ajuste debido a la conmutatividad de los atributos del antecedente de cada regla y de los errores en la semántica del sistema teoría + excepciones.

Donde w representa un factor menor que uno y que depende de la probabilidad de que dos teorías representen la misma colección de ejemplos. Lo que está íntimamente relacionado con la redundancia de los atributos; es decir, con la relación del número de atributos que se manejan y el número de ellos realmente necesarios para determinar completamente el atributo clase (concepto de atributos clave en bases de datos). Por defecto el sistema trabaja con w= 0.5, sin embargo Quinlan aclara que: ... por suerte el sistema no es muy sensible a este parámetro.

- *La regla por defecto*. La regla por defecto, regla que carece de antecedente, se introduce para clasificar a los ejemplos que no quedan cubiertos por la familia de reglas de la fase anterior. El consecuente es la clase más numerosa entre los ejemplos que restan por cubrir.

Todavía después de haber introducido la regla por defecto se realiza una nueva depuración eliminando reglas con bajo rendimiento aunque, para mejorar la comprensión y coherencia de la familia de reglas resultante en muchas ocasiones, se mantiene reglas que son del todo superfluas para la correcta clasificación de los ejemplos de entrenamiento.

Todas las fases están pensadas para obtener una buena fiabilidad de la familia de reglas final sobre conjuntos de ejemplos diferentes al de entrenamiento.

3.6. ALGORITMO SLIQ

La información que se presenta a continuación fue extraída de el documento SLIQ: *a Fast Scalable Classifier for Data Mining. Traducción libre* [26].

SLIQ considerado un algoritmo escalable para Data Mining, es un algoritmo que mejora el tiempo de aprendizaje para el clasificador sin perder la exactitud. Al mismo tiempo, permite la clasificación de grandes conjuntos de datos de entrenamiento residentes en disco, SLIQ exhibe algunas características de exactitud, su ejecución es rápida y produce árboles pequeños.

SLIQ no restringe la cantidad de datos de entrenamiento ni el número de atributos de los ejemplos. Además, puede potencialmente obtener gran exactitud clasificando grandes conjuntos de datos de entrenamiento que no pueden manejar otros clasificadores.

SLIQ es un clasificador que usa árboles de decisión y que pude manejar tanto atributos numéricos como categoríales. También usa una técnica de pre-clasificación en la etapa de construcción del árbol para reducir el coste de la evaluación de particiones por atributos numéricos. Este proceso de clasificación se integra con una estrategia de crecimiento del árbol primero en anchura para permitir a SLIQ clasificar grandes bases de datos residentes en disco. Además SLIQ usa un algoritmo eficiente de obtención de subconjuntos para determinar las particiones con atributos categoríales. Otra característica interesante es el uso de un nuevo algoritmo de poda basado en el principio de Descripción de Longitud Mínima (*MDL, Minimun Description Length*). Este algoritmo es poco costoso y produce árboles compactos y eficaces. La combinación de todo lo anterior hace que SLIQ posea una gran escalabilidad y que sea capaz de clasificar grandes conjuntos de datos con un gran número de clases, atributos y ejemplos.

3.6.1. Construcción del árbol. Para atributos numéricos, el tiempo de ordenamiento es el factor dominante cuando se encuentra la mejor división de un nodo del árbol de decisión. Sin embargo, la primera técnica usada en SLIQ implementa un esquema que elimina la necesidad de ordenar los datos para cada nodo del árbol de decisión.

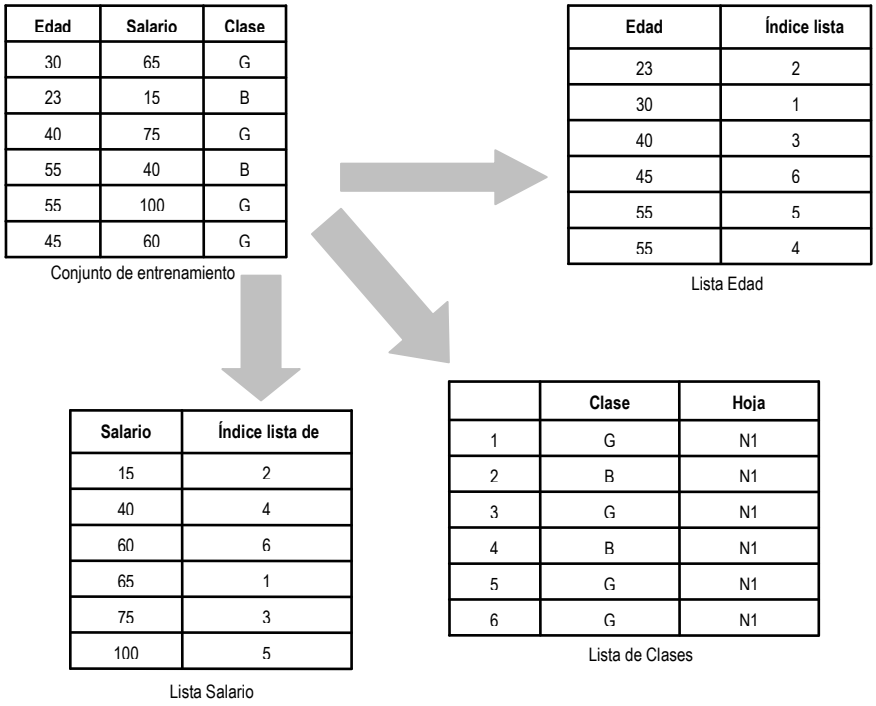
En cambio, los datos de entrenamiento sólo una vez para cada atributo numérico al principio de la fase de construcción del árbol.

Para lograr este pre-ordenamiento, se usa la siguiente estructura de datos. (ver figura 16)

Se crea una lista separada para cada atributo del conjunto de entrenamiento.
Adicionalmente, se utiliza otra lista, llamada lista de clase, para las marcas de clase ligadas a los ejemplos. La lista de atributos tiene dos campos: uno contiene el valor del atributo, el otro un índice de la lista de clase.

Una entrada de la lista de clase contiene dos campos: uno contiene la marca de la clase, el otro una referencia a un nodo hoja del árbol de decisión. La i-esima posición de la lista de clase corresponde al i-ésimo ejemplo de los datos de entrenamiento. Cada nodo hoja del árbol de decisión representa una partición de los datos de entrenamiento y está definida por la conjunción de predicados de un camino desde la raíz hasta el nodo. De esa manera, la lista de clase puede en cualquier momento identificar la partición a la cual pertenece un ejemplo. Se asume que existe suficiente memoria para mantener la lista de clase residente en ella. La lista de atributos se guarda en el disco, si es necesario.

Figura 16. Preclasificación de SLIQ



Inicialmente, los campos de referencia de las hojas de todas las posiciones de la lista de clase son puestas apuntando a la raíz del árbol de decisión. Luego se realiza una pasada sobre los datos de entrenamiento, distribuyendo a través de toda la lista los valores de los atributos para cada ejemplo. Cada valor de atributo es también etiquetado con el correspondiente índice de la lista de clase. Las listas de atributos para los atributos numéricos se ordenan independientemente. La Figura 16 ilustra el estado de la estructura de datos antes y después del pre-ordenamiento.

- **Métrica GINI:** Esta métrica se usa en los algoritmos de clasificación por el método de árboles de decisión para medir el grado de impureza de un atributo con respecto al atributo de decisión. Dadas la probabilidades para cada clase (P_i), la función GINI se define como:

$GINI(t) = 1 - \sum[p(j|t)]^2$, donde $p(j|t)$ es la frecuencia relativa de la clase j en el nodo p.

3.6.2 División del nodo procesado. En lugar de utilizar la estrategia primero en profundidad usada en los primeros clasificadores de árboles de decisión, se construye el árbol utilizando la estrategia primero en anchura. Consecuentemente, la división de todas las hojas del árbol actual son simultáneamente evaluados en una pasada sobre los datos. La Figura 17. proporciona un esquema del proceso de evaluación.

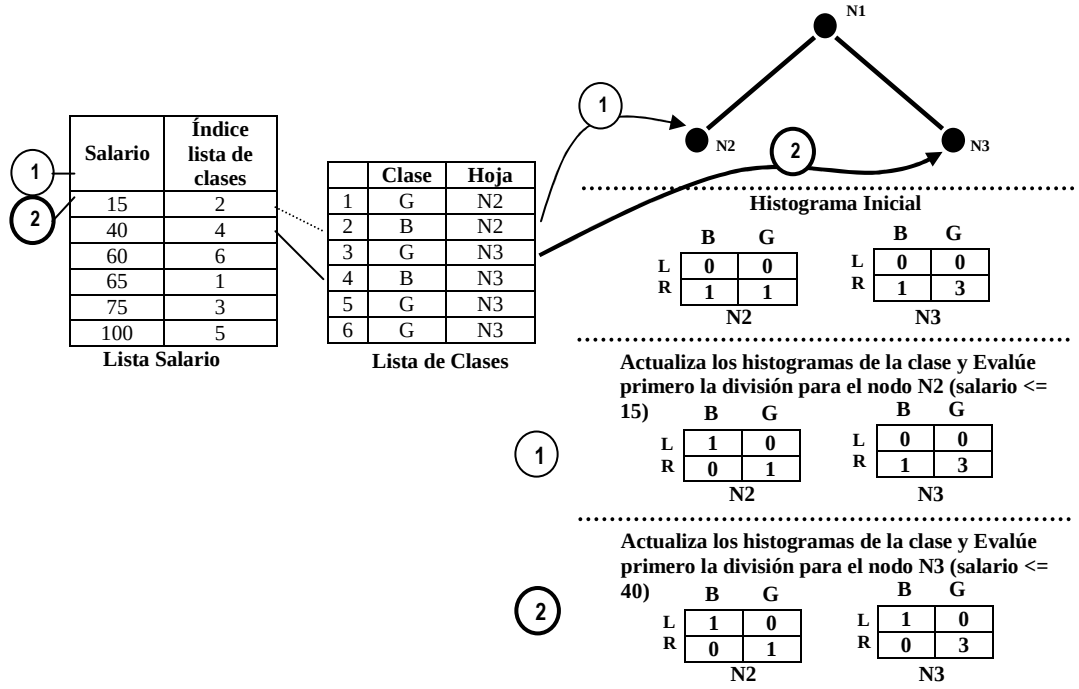
Para calcular el índice de partición *gini* para un atributo en un nodo son necesarios los valores de distribución de frecuencias de las clases en la partición correspondiente al nodo. La distribución se almacena en un histograma que se adjunta a cada nodo.

Las listas de atributos son procesadas una por una, para cada valor v en la lista del atributo en análisis se busca la entrada correspondiente en la lista de clases. De esta forma es posible saber a que nodo hoja esta asociado. A continuación se hace la actualización del histograma asociado al nodo hoja para reflejar que se ha encontrado un ejemplo más que cumple los tests correspondientes a esa hoja (Figura 18.).

Figura 17. Evaluación de la división

```
EvaluateSplits()
Para cada atributo A hacer
 Recorrer la lista de Atributos A
  Para cada valor v de la lista de atributos hacer
    Encontrar la entrada correspondiente de la lista de clase
          Y por consiguiente la clase correspondiente del nodo hoja
                Actualizar el histograma de clase de la hoja 1
                      Si A es un atributo numérico entonces
                        Calcular el índice de división para la prueba
                              (A < V) para la hoja 1
                      Si A es un atributo categórico entonces
                         Para cada hoja del árbol hacer
                          Encontrar el subconjunto de A con la mejor división
```

Figura 18. Procesamiento de particiones de nodos



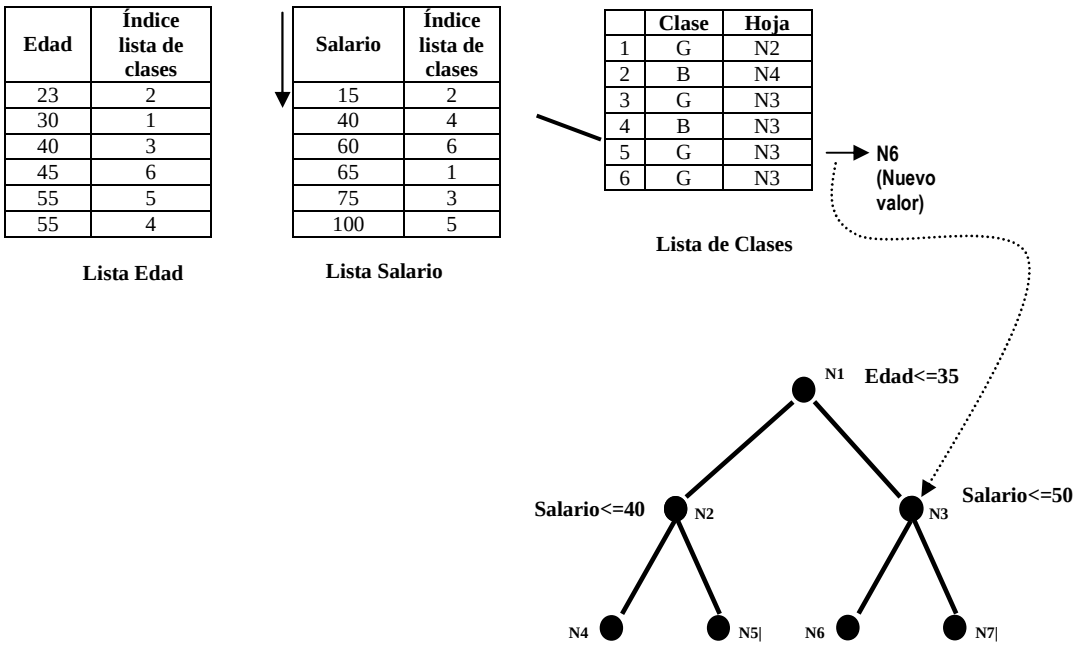
3.6.3. Actualización de la lista de clase. El siguiente paso una vez conocida la mejor partición aplicable es crear los nodos hijos y actualizar la lista de clases. En la lista de clases, para las entradas correspondientes a los ejemplos que se han visto afectados por la partición, se ha de reflejar que el nodo hoja asociado ha cambiado (será un hijo del anterior). La Figura 19. describe el proceso de actualización.

Figura 19. Algoritmo para actualización de la lista de clases

```
UpdateLabels()
Para cada atributo A usado en la división hacer
  Recorrer la lista de atributos de A
  Para cada valor v de la lista de atributos hacer
        Encontrar la entrada correspondiente de la lista de clase
        Encontrar la nueva clase c para la cual pertenece y aplicando
              la prueba de división para el nodo referenciado desde e
        Actualizar la marca de clase para e hasta c
        Actualizar el nodo referenciado en e para los hijos
              Correspondientes a la clase c
```

La Figura 20. Muestra este procedimiento con un ejemplo puntual:

Figura 20. Actualización de la lista de clases



3.6.4. Partición de atributos numéricos. Una partición binaria es de la forma $A \leq v$, donde v es un número real. El primer paso en la evaluación de una partición para atributos

numéricos es clasificar los ejemplos basándose en el valor del atributo considerado para partir. Sean los valores ordenados $v_1, v_2,\ldots, v_n$, como cualquier valor entre v_i y v_{i+1} divide el conjunto en dos partes solo se ha de evaluar *n-1* posibles particiones binarias.

El coste de la evaluación de una partición por un atributo numérico esta dominado por el coste de clasificar los ejemplos.

3.6.5. Particiones de atributos categoríales. Las particiones para atributos categoríales son de la forma $A \in S'$, donde $S' \subset S$ y *S* es el conjunto de todos los posibles valores del atributo A. La evaluación de todos los subconjuntos puede tener un coste prohibitivo si la cardinalidad del conjunto es alta, si el número de posibles valores del atributo A es n hay 2^n posibles subconjuntos.

SLIQ toma una política híbrida, de forma que si la cardinalidad es inferior a un valor umbral (típicamente 10) se evalúan todos los posibles subconjuntos. En caso de superar este umbral se opta por la aplicación de un algoritmo greedy para la construcción del subconjunto deseado. El algoritmo comienza con un conjunto vacío de atributos S' y añade aquel elemento de S que proporciona la mejor partición. Este proceso se repite hasta que no se consigue mejora en la partición obtenida. Con este enfoque se consiguen buenos resultados para conjuntos con cardinalidad alta.

3.6.6. Ejemplo del algoritmo Sliq. Sea un conjunto de datos de entrenamiento, el cual es representado en la siguiente tabla:

Tabla 5. Datos de entrenamiento para Sliq

Id	Color	Tamaño	Tacto	Material	Actitud
1	Verde	Grande	Duro	Madera	Negativa
2	Verde	Grande	Duro	Plástico	Negativa
3	Azul	Grande	Duro	Plástico	Positiva
4	Verde	Pequeño	Duro	Madera	Positiva
5	Verde	Pequeño	Duro	Cartón	Indiferente
6	Rojo	Pequeño	Duro	Cartón	Indiferente
7	Rojo	Mediano	Blando	Cartón	Indiferente

Figura 21. Pre-ordenamiento de los atributos

Index Lista de Clases	Color		Index Lista de Clases	Material			Clase	Hoja
1	Verde		1	Madera		1	Negativa	**N1**
2	Verde		2	Plástico		2	Negativa	**N1**
3	Azul		3	Plástico		3	Positiva	**N1**
4	Verde		4	Madera		4	Positiva	**N1**
5	Verde		5	Cartón		5	Indiferente	**N1**
6	Rojo		6	Cartón		6	Indiferente	**N1**

7	Rojo		7	Cartón		7	Indiferente	**N1**
Lista Color			Lista Material			Lista de Clases		

Se crea una lista por cada atributo y una lista de clase como se muestra en la figura 21.

Como los atributos son categóricos entonces se calculan todos los posibles subconjuntos de estos atributos:

Para un atributo con tres valores diferentes como Color, se generan los siguientes subconjuntos de posibles divisiones.

[[0],[1,2]]	[[0,1],[2]]	[[0,2],[1]]	[[0],[1],[2]]

Si un atributo categórico tiene cuatro valores diferentes, las posibles divisiones son las siguientes:

[[0],[1,2,3]]	[[0,1],[2,3]]	[[0,1,2],[3]]	[[0,1,3],[2]]
[[0,1],[2],[3]]	[[0,2],[1,3]]	[[0,2,3],[1]]	[[0,2],[1],[3]]
[[0,3],[1,2]]	[[0,3],[1],[2]]	[[0],[1],[2,3]]	[[0],[1,2],[3]]
[[0],[1,3],[2]]	[[0],[1],[2],[3]]		

Ahora se calcula el index Gini para cada posible división del ejemplo tratado. Se muestra el histograma final para cada posible división del atributo Material

Para la división de [[0], [1,2]] se tiene:

	[Madera]	[Cartón, Plástico]	Total
Indiferente	0	3	3
Positiva	1	1	2
Negativa	1	1	2
Total	2	5	7

El index gini para este histograma es:

$$\frac{2}{7}*\left[1-\left[\left(\frac{0}{2}\right)^2+\left(\frac{1}{2}\right)^2+\left(\frac{0}{2}\right)^2\right]\right]+\left[1-\left[\left(\frac{3}{5}\right)^2+\left(\frac{1}{5}\right)^2+\left(\frac{1}{5}\right)^2\right]\right]=0.542857$$

Para la división de [[0,1], [2]] se tiene:

	[Madera, Cartón]	[Plástico]	Total
Indiferente	3	0	3
Positiva	1	1	2
Negativa	1	1	2
Total	5	2	7

El index gini para este histograma es:

$$\frac{5}{7}*\left[1-\left[\left(\frac{3}{5}\right)^2+\left(\frac{1}{5}\right)^2+\left(\frac{1}{5}\right)^2\right]\right]+\left[1-\left[\left(\frac{0}{2}\right)^2+\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right]\right]=0.542857$$

Para la división de [[0,2],[1]] se tiene:

	[Madera, Plástico]	[Cartón]	Total
Indiferente	0	3	3
Positiva	2	0	2
Negativa	2	0	2
Total	4	3	7

El index gini para este histograma es:

$$\frac{4}{7}*\left[1-\left[\left(\frac{0}{4}\right)^2+\left(\frac{2}{4}\right)^2+\left(\frac{2}{4}\right)^2\right]\right]+\left[1-\left[\left(\frac{3}{3}\right)^2+\left(\frac{0}{3}\right)^2+\left(\frac{0}{3}\right)^2\right]\right]=0.285714$$

Para la división de [[0,[1],[2]] se tiene:

	[Madera]	[Cartón]	[Plástico]	Total
Indiferente	0	3	0	3
Positiva	1	0	1	2
Negativa	1	0	1	2
Total	2	3	2	7

El index gini para este histograma es:

$$\frac{3}{7}*\left[1-\left[\left(\frac{0}{3}\right)^2+\left(\frac{1}{3}\right)^2+\left(\frac{1}{3}\right)^2\right]\right]+\frac{3}{7}\left[1-\left[\left(\frac{3}{3}\right)^2+\left(\frac{0}{3}\right)^2+\left(\frac{0}{3}\right)^2\right]\right]+\frac{2}{7}\left[1-\left[\left(\frac{0}{2}\right)^2+\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right]\right]=0.28571$$

La mejor división para el atributo materia es: [[Madera, Plástico], [Cartón]], es ejemplo muestra que existen dos divisiones con el mismo valor, en este caso se escoge la primera mejor división encontrada.

A continuación se muestra los index gini para cada posible división de los otros atributos:

Index Gini del atributo Color

0.5238095238095238
0.5476190476190476
0.4571428571428571
0.3571428571428571

El index gini de la mejor división para el atributo Color es: 0.3571428571428571

Index Gini del atributo Tamaño

0.5476190476190476
0.5714285714285715
0.40476190476190477
0.38092538092598093
El index gini de la mejor división para el atributo Tamaño es: 0.38092538092598093

Index Gini del atributo Tacto

0.5714285714285715

El index gini de la mejor división para el atributo Tacto es 0.5714285714285715
El menor Index gini de las mejores divisiones de los atributos de constituye en el mejor atributo para dividir el conjunto de entrenamiento. Por tanto, material con la división [[Madera, Plástico], [Cartón]] es el primer atributo de la división. La Figura 22 muestra la actualización de la lista de clase.

Figura 22. Actualización de la lista de clases para la primera división

Index Lista de Clase	Material			Clase	Hoja
1	Madera		1	Negativa	N2
2	Plástico		2	Negativa	N2
3	Plástico		3	Positiva	N2
4	Madera		4	Positiva	N2
5	Cartón		5	Indiferente	N3

6	Cartón		6	Indiferente	N3
7	Cartón		7	Indiferente	N3

Lista Material ***Lista Clase***

Figura 23. Actualización de la lista de clases para la segunda división

Index Lista de Clase	Material			Clase	Hoja
1	Madera		1	Negativa	N4
2	Plástico		2	Negativa	N5
3	Plástico		3	Positiva	N5
4	Madera		4	Positiva	N5
5	Cartón		5	Indiferente	N3
6	Cartón		6	Indiferente	N3
7	Cartón		7	Indiferente	N3

Lista Material ***Lista Clase***

El proceso de división continúa para el nodo N2, donde el mejor atributo es Color con división [[Azul], [Rojo, Verde]]. La figura 22 muestra la actualización de la lista de clase.

Procesa ahora el nodo N5, donde luego de calcular el index gini el mejor atributo Tamaño con división [[Pequeño], [Grande, Mediano]]. La figura 23 muestra la actualización de la lista de clase.

En este momento el procedimiento termina, porque todas las hojas contienen atributos con el mismo valor.
Las reglas generadas son:

- **Material** = Cartón ➔ **Indiferente**
- (**Material** = Plástico ó **Material** = Madera) y **Color** = Azul ➔ **Positiva**
- (**Material** = Plástico ó **Material** = Madera) y (**Color** = Rojo ó **Color** = Verde) y **Tamaño** = Pequeño ➔ **Positiva**
- (**Material** = Plástico ó **Material** = Madera) y (**Color** = Rojo ó **Color** = Verde) y (**Tamaño** = Grande ó **Tamaño** = Mediano) ➔ **Negativa**

En la figura 25. Se muestra en forma de árbol las reglas generadas.

Figura 24. Actualización de la lista de clases para la tercera división.

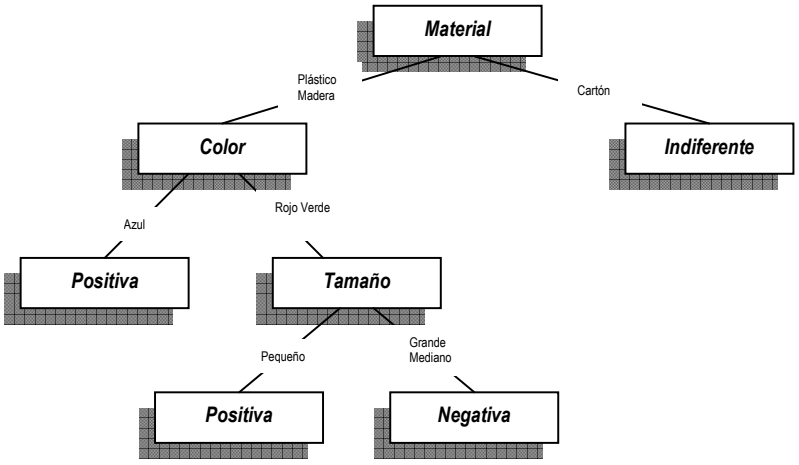
Index Lista de Clase	Material
1	Madera
2	Plástico
3	Plástico
4	Madera
5	Cartón
6	Cartón
7	Cartón

Lista Material

	Clase	Hoja
1	Negativa	N4
2	Negativa	N7
3	Positiva	N7
4	Positiva	N6
5	Indiferente	N3
6	Indiferente	N3
7	Indiferente	N3

Lista Clase

Figura 25. Árbol de decisión generado por Sliq



Se describe ahora el funcionamiento con atributos numéricos; para ello se cuantifican los atributos Temperatura y Humedad. Temperatura con valores entre [9,37] y humedad con valores entre [100.6, 186.3], este proceso es mostrado en la tabla 6.

Tabla 6. Conjunto de datos de entrenamiento don atributos continuos

Día	*Pronostico*	*Temperatura*	*Humedad*	*Viento*	*Jugar Tenis*
D1	*Soleado*	*36*	*180.5*	*Débil*	*No*
D2	*Soleado*	*35*	*181.2*	*Fuerte*	*No*
D3	*Nublado*	*36*	*182.6*	*Débil*	*Si*
D4	*Lluvia*	*20*	*180.5*	*Débil*	*Si*

D5	*Lluvia*	*12*	*102.6*	*Débil*	*Si*
D6	*Lluvia*	*22*	*180.9*	*Fuerte*	*No*
D7	*Nublado*	*22*	*180.3*	*Fuerte*	*Si*
D8	*Soleado*	*11*	*101.6*	*Débil*	*No*
D9	*Soleado*	*9*	*104.1*	*Débil*	*Si*
D10	*Lluvia*	*21*	*102.4*	*Débil*	*Si*
D11	*Soleado*	*21*	*100.3*	*Fuerte*	*Si*
D12	*Nublado*	*20*	*186.3*	*Fuerte*	*Si*
D13	*Nublado*	*37*	*100.6*	*Débil*	*Si*
D14	*Lluvia*	*22*	*180.6*	*Fuerte*	*no*

Se muestra los index de los atributos categóricos:

Index Gini para el atributo Categórico Pronóstico:

0.3936507936507937
0.31746031746031744
0.4428571428571429
0.3142857142857143
Index Gini para el atributo Categórico Viento:

0.42857142857142855

El proceso para el cálculo del index gini para los atributos Numéricos es el siguiente: Los valores de los atributos numéricos son ordenados, la figura 27 ilustra el ordenamiento del atributo Temperatura:

Figura 27. Ordenamiento del atributo numérico Temperatura

Temperatura	Índice de Lista		Temperatura	Índice de Lista
9	9		No	N1
11	8		No	N1
12	5		Si	N1
20	4		Si	N1
20	12		Si	N1
21	10		No	N1
21	11		Si	N1
22	6		No	N1
22	7		Si	N1
22	14		Si	N1
35	2		Si	N1
36	1		Si	N1
36	3		Si	N1
37	13		No	N1

Lista para el atributo Temperatura ***Lista de Clase Inicial***

Luego se construye el histograma de clase particionando en cada valor del atributo para calcular el index gini. Los valores del index gini para cada valor del atributo temperatura y humedad son presentados a continuación.

Tabla 8 Index Gini

9	0.43956043956043955		100.3	0.47619047619047616
11	0.45238095238095233		100.6	0.49107142857142855
12	0.4588744588744589		101.6	0.45928375350140037
20	0.45		102.4	0.46428571428571425
20	0.43174603174603166		102.6	0.4721804511278196
21	0.4047619047619047		104.1	0.4809523809523811
21	0.3673469387755103		180.3	0.4897959183673469
22	0.42857142857142855		180.5	0.46753245753246747
22	0.3936507936507937		180.5	0.47342995169082136
22	0.4428571428571429		180.6	0.4619047619047618
35	0.4588744588744589		180.9	0.4592207792207792
36	0.4166666666666666		181.2	0.462454212245421253
36	0.43956043956043955		182.6	045991045991045987
37	0.5		186.3	0.4591836734693877

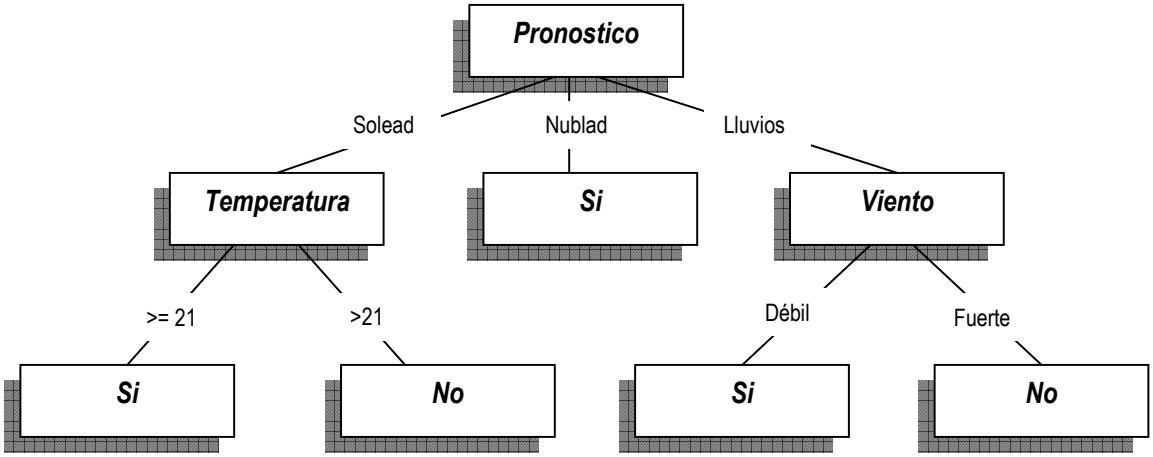
La mejor división es el atributo Pronóstico con el subconjunto: [[Soleado], [Nublado], [Lluvia]] cuyo index gini es 0.3142857142857143, luego entonces, este atributo es el primer atributo de división del conjunto de entrenamiento.

El procedimiento continua recursivamente coma ya se ha descrito, hasta que cumpla la condición de parada. Las siguientes son las reglas generadas.

Pronostico *= Soleado ó* ***Temperatura*** *<= 21 ➔ Yes*
Pronostico *= Soleado ó* ***Temperatura*** *> 21 ➔ No*
Pronostico *= Soleado ó* ***Viento*** *= Fuerte ➔ No*
Pronostico *= Soleado ó* ***Viento*** *= Débil ➔ Yes*
Pronostico *= Nublado ➔ Yes*

En la figura 28. Se muestra en forma de árbol las reglas generadas.

Figura 28. Reglas generadas por Sliq



3.7. ALGORITMO MATE-TREE

Mate-tree es un algoritmo propuesto por Timarán, se basa en los nuevos operadores del álgebra relacional Mate, Describe Classifier, y las funciones agregadas *Entro()* y *Gain()* para la generación de reglas de clasificación propuestos por Timarán [36], [37]. El algoritmo es el siguiente.

Figura 29. Algoritmo Mate-Tree

```
//Calcula entropía atributo clase Ac
Entro(Ac;Ac, R);
Forall t Î R do begin
// Se aplica el operador Mate
        R1=mLC,Ac(R)
// cuenta las ocurrencias de las diferentes combinaciones
// de los atributos condición LC y atributo clase Ac
        Count(R1)
// calcula entropia
        Entro(LC,R1)
// calcula ganancia
        R2=Gain(LC,R1)
End
//construye el arbol de decisión con operador Describe Classifier
        Arbol=bm(R2)
```

El primer paso del algoritmo calcula la entropía del atributo clase de la relación R. En el subsiguiente paso, se aplica el operador Mate con el fin de generar todas las posibles combinaciones de los atributos condición con el atributo clase. Se cuenta el número de ocurrencias de cada combinación y a la relación resultante R1, se aplica la función agregada Entro() que calcula para cada combinación de atributos condición y clase la entropía de la relación R1 con respecto a estos atributos. Posteriormente se calcula con la función agregada *Gain()*, la ganancia de Información obtenida por el particionamiento de la

relación R1 por las diferentes combinaciones de atributos condición y clase. Finalmente con la relación resultante R2, el operador describe *classifier* construye el árbol de decisión. En la Figura 29., se muestra el algoritmo Mate-tree.

3.7.1 Operador Mate (μ). El operador Mate genera, por cada una de las tuplas de una relación, todas los posibles combinaciones formadas por los valores no nulos de los atributos pertenecientes a una lista de atributos denominados Atributos Condición, y el valor no nulo del atributo denominado ***atributo clase***, tiene la siguiente sintaxis:

$$\mu_{\text{lista\_atributos\_condición; atributo\_clase}}(R)$$

Donde <lista_atributos_condición> es el conjunto de atributos de la relación R a combinar con el atributo clase y <atributo_clase> es el atributo de R definido como clase.

Mate toma como entrada cada tupla de R y produce una nueva relación cuyo esquema está formado por los atributos condición, <lista_atributos_condición> y el atributo clase, <atributo_clase> con tuplas formadas por todas las posibles combinaciones de cada uno de los atributos condición con el atributo clase, los demás valores de los atributos se hacen nulos.

Mate empareja en cada partición todos los atributos condición con el atributo clase, lo que facilita el conteo y el posterior cálculo de las medidas de entropía y ganancia de información. El operador Mate genera estas combinaciones, en una sola pasada sobre la tabla de entrenamiento (lo que redunda en la eficiencia del proceso de construcción del árbol de decisión).

Ejemplo. Sea la relación *R(A,B,C,D*) de la tabla 2 obtener las diferentes combinaciones de los atributos *A,B* con el atributo *D*, es decir, R1= $\mu_{A,B;D}$(R). El resultado de la operación R1= $\mu_{A,B;D}$(R), se muestra en la tabla 10.

Tabla 10. Relación R

A	B	C	D
A1	B1	c1	d1
A1	B2	c1	d2

Tabla 11. Resultado operación R1=μA,B;D(R)

A	B	D
A1	null	d1
Null	B1	d1
A1	B1	d1

A1	null	d2
Null	B2	d2
A1	B2	d2

3.7.2. Operador Describe Classifier (βμ). Describe Classifier (βμ) es un operador unario que toma como entrada la relación resultante de los operadores *Mate By*, *Entro() y Gain()* y produce una nueva relación donde se almacenan los valores de los atributos que formarán los diferentes nodos del árbol de decisión. La sintaxis del operador Describe Classifier es la siguiente:

$$\beta\mu\ (R)$$

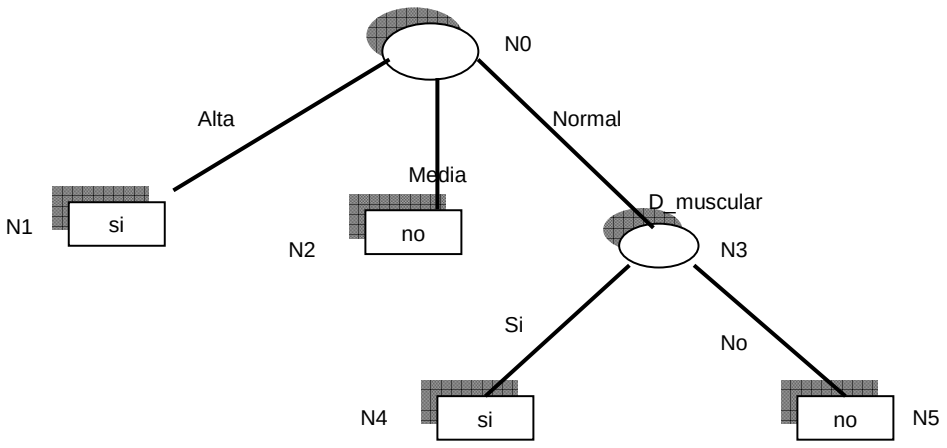
Describe Classifier facilita la construcción del árbol de decisión y por consiguiente la generación de reglas de clasificación.

Ejemplo: sea la relación ÁRBOL (NODO, PADRE, ATRIBUTO, VALOR, CLASE) de la tabla 12 resultado del operador Describe Classifier. El resultado se muestra en la Figura 30.

Tabla 12. Relación árbol

NODO	PADRE	ATRIBUTO	VALOR	CLASE
N0	Null	Temperatura	Null	Null
N1	N0	Temperatura	Alta	Si
N2	N0	Temperatura	Media	No
N3	N0	Temperatura	Normal	Null
N4	N3	D_Muscular	Si	Si
N5	N3	D_Muscular	No	No

Figura 30. Relación árbol



3.7.3. Función algebraica agregada Entro(). La función *Entro()* permite calcular la entropía de una relación R con respecto a un atributo denominado atributo condición y un atributo clase. Tiene la siguiente sintaxis:
Entro(atributo; atributo_clase;R)

donde <atributo> es el atributo condición de la relación R y <atributo_clase> es el atributo con el que se combina el atributo <atributo>.

3.7.4 Función algebraica agregada Gain(). La función *Gain()* permite calcular la reducción de la entropía causada por el conocimiento del valor de un atributo de una relación. Su sintaxis es:

Gain(atributo;atrib_clase;R)

Donde <atributo> es el atributo condición de la relación R y <atributo_clase> es el atributo con el que se combina el atributo <atributo>.

La función *Gain()* permite calcular la ganancia de información obtenida por el particionamiento de la relación R de acuerdo con el atributo <atributo> y se define: *Gain(Ak;Ac; R)={y | y = Entro(Ac;Ac; R) - Entro(Ak;Ac; R)}* donde *Entro (Ac; Ac; R)* es la entropía de la relación *R* con respecto al atributo clase *Ac* y *Entro(Ak;Ac; R)* es la entropía de la relación *R* con respecto al atributo *Ak* .

3.7.5. Ejemplo del algoritmo Mate-Tree. Sea un conjunto de datos de entrenamiento, el cual es representado en la siguiente tabla:

Tabla 13. Datos de entrenamiento para Mate-tree

SID	D_MUSCULAR	TEMPERATURA	GRIPA
1	Si	Alta	Si
2	No	Alta	Si
3	Si	Media	No
4	No	Media	Si
5	Si	Normal	Si
6	No	Normal	No

Si Se aplica a esta tabla la primitiva MATE BY, se forman todas las posibles combinaciones con los atributos D_MUSCULAR, TEMPERATURA y el atributo clase GRIPA (Tabla 13). Se cuenta la frecuencia de las parejas y finalmente se almacena en una nueva tabla.

Tabla 14. Resultado primitiva Mate

D_MUSCULAR	TEMPERATURA	GRIPA
Si	Null	Si
Null	Alta	Si
Si	Alta	Si
No	Null	Si
Null	Alta	Si
No	Alta	Si
Si	Null	No
Null	Media	No
Si	Media	No
No	Null	Si
Null	Media	Si
No	Media	Si
Si	Null	Si
Null	Normal	Si
Si	Normal	Si
No	Null	No
Null	Normal	No
No	Normal	No

Tabla 15. Tabla Clase

D_MUSCULAR	TEMPERATURA	GRIPA	COUNT
Si	Null	Si	2
Si	Null	No	1
Si	Normal	Si	1
Si	Media	No	1
Si	Alta	Si	1
No	Null	Si	2
No	Null	No	1
No	Normal	No	1
No	Media	Si	1
No	Alta	Si	1
Null	Normal	Si	1
Null	Normal	No	1
Null	Media	Si	1
Null	Media	No	1
Null	Alta	Si	2

A esta tabla se le aplica la función agregada Entro() que calcula la entropía de una tabla con respecto a cada una de las combinaciones de los atributos condición con el atributo clase. Se obtiene la tabla de entropía.

Tabla 16. Resultado parcial función Entro()

D_MUSCULAR	TEMPERATURA	GRIPA	COUNT	ENTRO()
Si	Null	Si	2	$2/3\log_2(2/3)$
Si	Null	No	1	$1/3\log_2(1/3)$
Si	Normal	Si	1	$1/1\log_2(1/1)$
Si	Media	No	1	$1/1\log_2(1/1)$
Si	Alta	Si	1	$1/1\log_2(1/1)$
No	Null	Si	2	$2/3\log_2(2/3)$
No	Null	No	1	$1/3\log_2(1/3)$
No	Normal	No	1	$1/1\log_2(1/1)$
No	Media	Si	1	$1/1\log_2(1/1)$
No	Alta	Si	1	$1/1\log_2(1/1)$
Null	Normal	Si	1	$1/2\text{og}_2(1/2)$
Null	Normal	No	1	$1/2\text{og}_2(1/2)$
Null	Media	Si	1	$1/2\text{og}_2(1/2)$
Null	Media	No	1	$1/2\text{og}_2(1/2)$
Null	Alta	Si	2	$2/2\text{og}_2(2/2)$

Tabla 17. Resultado Entropía

D_MUSCULAR	TEMPERATURA	GRIPA	COUNT	ENTRO()
Si	Null	SiNo	3	$2/3\log_2(2/3) + 1/3\log_2(1/3)$
Si	Normal	Si	1	$1/1\log_2(1/1)$
Si	Media	No	1	$1/1\log_2(1/1)$
Si	Alta	Si	1	$1/1\log_2(1/1)$
No	Null	SiNo	3	$2/3\log_2(2/3) + 1/3\log_2(1/3)$
No	Normal	No	1	$1/1\log_2(1/1)$
No	Media	Si	1	$1/1\log_2(1/1)$
No	Alta	Si	1	$1/1\log_2(1/1)$
Null	Normal	SiNo	2	$1/2\text{og}_2(1/2) + 1/2\text{og}_2(1/2)$
Null	Media	SiNo	2	$1/2\text{og}_2(1/2) + 1/2\text{og}_2(1/2)$
Null	Alta	Si	2	$2/2\text{og}_2(2/2)$

Una vez calculada la entropía se aplica la función agregada gain(), la cual, inicialmente, concatena los diferentes valores de los atributos condición y clase en un solo valor, así mismo suma las ocurrencias y los valores de las entropías. Posteriormente, se calcula la entropía del conjunto clase y la de los posibles nodos y finalmente, con estos datos, calcula la ganancia de información y la almacena en la tabla de ganancia Tabla 18.

Tabla 18. Tabla de Ganancia

D_MUS-CULAR	TEMPE-RATURA	GRIPA	COUNT	ENTRO()	GAIN()
Si No	Null	SiNo SiNo	6	(2/3log$_2$(2/3) + 1/3log$_2$(1/3)) + (2/3log$_2$(2/3) + 1/3log$_2$(1/3))	(4/6log$_2$(4/6) + 2/6 log$_2$(2/6)) – 3/6 (2/3 log$_2$(2/3) + 1/3 log$_2$(1/3)) - 3/6 (2/3 log$_2$(2/3) + 1/3 log$_2$(1/3))
Si No	Normal	Si No	2	(1/1log$_2$(1/1)) + (1/1log$_2$(1/1))	(1/2 log$_2$(1/2) + ½ log$_2$(1/2)) – 1/2 (1/1 log$_2$(1/1)) – 1/2 (1/1 log$_2$(1/1))
SiNo	Media	SiNo	2	(1/1log$_2$(1/1)) + (1/1log$_2$(1/1))	2/2 log$_2$(2/2) – 1/2 (1/1 log$_2$(1/1)) – 1/2 (1/1 log$_2$(1/1))
SiNo	Alta	SiSi	1	(1/1log$_2$(1/1)) + (1/1log$_2$(1/1))	2/2 log$_2$(2/2) – 1/2 (1/1 log$_2$(1/1)) – 1/2 (1/1 log$_2$(1/1))
Null	Normal Media Alta	SiNo SiNo Si	6	(1/2og$_2$(1/2) + 1/2og$_2$(1/2)) + (1/2og$_2$(1/2) + 1/2og$_2$(1/2)) + (2/2og$_2$(2/2))	(4/6 log$_2$(4/6) + 2/6 log$_2$(2/6)) - 2/6 (1/2 log$_2$(1/2) + 1/2 log$_2$(1/2)) - 2/6 (1/2 log$_2$(1/2) + 1/2 log$_2$(1/2)) - 2/6 (2/2 log$_2$(2/2))

Y genera la tabla contenedora de nodos Tabla 18. A la cual se le aplica el operador describe classifier y genera el árbol de decisión de la Figura 27.

Tabla 19. Relación árbol

NODO	PADRE	ATRIBUTO	VALOR	CLASE
N0	Null	Temperatura	Null	Null
N1	N0	Temperatura	Alta	Si
N2	N0	Temperatura	Media	No
N3	N0	Temperatura	Normal	Null
N4	N3	D_Muscular	Si	Si
N5	N3	D_Muscular	No	No

3.8. TRABAJOS RELACIONADOS

Actualmente se encuentra una cantidad considerable de herramientas licenciadas y de libre distribución algunas ya con trayectoria y varias versiones y otras experimentales. Algunas como Alice, C5.0_RuleQuest, Qyield, CoverStory, See5, MLC++, ODBCmine y c4.5r8, entre otros, que soportan solamente la etapa de minería de datos en el proceso de DCBD y que se pueden acoplar con otras aplicaciones que ofrecen pre y post procesamiento de datos.

También se encuentran herramientas como Clementine, DBMiner, DBLearn, Data Mine, IMACS, Intelligent Miner, Quest y Weka que ofrecen soporte en más de una etapa del DCBD y además variedad en modelos de descubrimiento como clasificación, asociación,

visualización, clustering, entre otros. Otra característica con la que cuentan algunas herramientas es que poseen un modulo que permite hacer comparaciones entre los algoritmos de descubrimiento que soporta.

La mayoría de herramientas que existentes presentan una arquitectura débilmente acoplada aun SGBD, es decir con conexiones ODBC a SGBD o trabajan con archivos planos que contienen los datos a minar con determinada estructura, y algunos integran estos dos mecanismos para la adquisición de datos, lo que les permite ser portables y versátiles sin embargo, algunas herramientas de DM sólo permiten la conexión con determinados SGBD como Oracle, Postgres.

Hay herramientas exclusivas para el método de clasificación como See5, MLC++ y c4.5 [22].

See5 es la versión creada para Windows de C5.0 diseñado para Linux, es una herramienta genérica de tareas múltiples ya que soporta la etapa de Minería de Datos y aplicación de las reglas, tiene una interfaz gráfica muy básica, los datos para aplicar la técnica de clasificación los adquiere de archivos planos los cuales deben tener una estructura especifica para que los pueda interpretar, los nombres de los atributos y su características como los posibles valores deben estar en un archivo de extensión *.name* y los datos en otro archivo de extensión *.data,* los algoritmo de clasificación que utiliza son clasificadores basados en reglas y árboles de decisión, permite configurar los parámetros del clasificador y realiza pruebas al modelo creado. Los resultados se muestran en modo texto, es decir no posee un módulo gráfico para representar el árbol de decisión que se genera con el modelado, sin embargo, muestra cada una de las reglas en detalle; Otra herramienta que es específicamente para clasificación es la ODBCmine la cual aplica los algoritmo C4.5, ID3, esta herramienta trabaja con el SGBD Microsoft Access y se conecta utilizando los OBDC del sistema; El C4.5r8 para Linux trabaja con los algoritmos C4.5 y C4.5rules, maneja con archivos planos con la misma estructura de See5; También esta MLC++, que es una herramienta para Windows con interfaz gráfica y que utiliza archivos planos, entre los algoritmos de clasificación que maneja están el ID3, Vecino-cercano, Bayes, C4.5, C4.5rules,OC1 y T2.

Por otro lado existen herramientas genéricas de tareas múltiples, es decir, soportan mas de dos etapas del proceso KDD, una que se destaca dentro de este grupo es WEKA desarrollada bajo los lineamientos del software libre, WEKA es un entorno para experimentación de análisis de datos que permite aplicar, analizar y evaluar las técnicas más relevantes de análisis de datos, principalmente las provenientes del aprendizaje automático, sobre cualquier conjunto de datos del usuario. Para ello únicamente se requiere que los datos a analizar se almacenen con un cierto formato, conocido como ARFF (Attribute-Relation File Format). Está constituido por una serie de paquetes de código abierto con diferentes técnicas de preprocesado, clasificación, agrupamiento, asociación, y visualización, así como facilidades para su aplicación y análisis de prestaciones cuando son aplicadas a los datos de entrada seleccionados. Estos paquetes pueden ser integrados en

cualquier proyecto de análisis de datos, e incluso pueden extenderse con contribuciones de los usuarios que desarrollen nuevos algoritmos. Con objeto de facilitar su uso por un mayor número de usuarios, WEKA además incluye una interfaz gráfica de usuario para acceder y configurar las diferentes herramientas integradas. Hay diferentes tipos de operaciones, en etapas independientes, que se pueden realizar sobre los datos Preprocesamiento (selección de la fuente de datos y preparación (filtrado)); Clasificación en donde cuenta con algoritmos de bayes, functions, lazy, meta, mise, trees (ADtree, Id3, J48, NBtree), rules; Cluster Algoritmos de agrupamiento como EM, SimpleMeans y otros; Associate Algoritmos de búsqueda de reglas de asociación como A priori y Testius; Select Attributes búsqueda supervisada de subconjuntos de atributos representativos con algoritmos como CfsSubsetEval, OneRAttributeEval y Visualize Herramienta interactiva de presentación gráfica en 2D. La herramienta permite cargar los datos de tres maneras: ficheros de datos, acceso a una base de datos y acceso a través de Internet sobre una dirección URL de un servidor Web, los atributos pueden ser principalmente de dos tipos: numéricos de tipo real o entero (indicado con las palabra real o integer tras el nombre del atributo), y simbólicos, en cuyo caso se especifican los valores posibles que puede tomar entre llaves.

Matekdd es una herramienta que ofrece soporte en todas las fases del proceso de Descubrimiento de Conocimiento en Bases de Datos (Selección, preprocesamiento, transformación, minería de datos y evaluación), además que opera directamente dentro del SGBD que para efectos es PostgreSQL al aplicar los algoritmos implementados en Funciones definidas por el usuario (FDU's), es decir, que su arquitectura es medianamente acoplada con el Gestor, esto es favorable en el sentido en que trabaja directamente con una base de datos y aplica el proceso de minería desde el gestor ganando en velocidad de procesamiento. Para manejar la herramienta cuenta con una interfaz gráfica que permite al usuario (analista) interactuar con el Gestor de una forma fácil, permite configurar el clasificador seleccionando el algoritmo a aplicar, definir el porcentaje de partición de los datos para entrenamiento y prueba para luego aplicar el modelo y obtener unos resultados de forma gráfica al poder visualizar el árbol que se genera, y las reglas de una forma detallada y clara.

4. ÁNALISIS DEL SISTEMA

En este capítulo se presenta el análisis orientado a objetos del sistema Mate-kdd en UML.

4.1. ANÁLISIS ORIENTADO A OBJETOS DEL SISTEMA MATE-KDD

4.1.1 Descripción del sistema actual. En la actualidad no existe en la región con el acceso a herramientas mineras de datos, esto ya que el concepto es nuevo para nosotros, aun cuando la noción de KDD tiene tiempo de estar difundiéndose, no se esta a la vanguardia de los procesos que se han implementado para aprovechar los grandes volúmenes que se están produciendo en todos los sectores.

Por otro lado en cuanto a herramientas, se han implementado algunas de libre distribución, pero una desventaja de ellas es que no cubren todos los procesos de KDD, pero se puede integrar éstas a otros proyectos de manera experimental para aprovecharlas a manera de estudio, más sin embargo, en el mercado del software licenciado existen algunas que ofrecen muchos servicios y apoyo en todas de las etapas del proceso KDD y que están ya apoyando a grandes empresas en el mundo: son herramientas costosas, de difícil acceso para le pequeña empresa, por eso en la región no se ha optado por buscar en este tipo de posibilidades el éxito de las mismas.

Hay en el entorno varios sistemas para minería de datos como Alice, C4.0RuleQuest, herramientas que contemplan únicamente la fase de minería de datos, en lo referente a preparación y preprocesamiento de datos hay que acudir a un pre y un post procesamiento, no se encuentran herramientas que ofrezcan soporte a todas las etapas de DCBD y que a la vez sean medianamente acopladas a un SGBAD, por otra parte, Weka es una herramienta se software libre que ofrece soporte en todas las fases de descubrimiento de conocimiento y en el cual se va guiar el presente proyecto.

Existe una herramienta completa en el mercado, Clementine, pero funciona bajo los lineamientos de Software Licenciado. Se suma a lo anterior que las herramientas existentes son débilmente acopladas con SGBD lo que implica que haya una conexión, se extraigan los datos y se les aplique de forma independiente al gestor las técnicas de minería o simplemente utilizan ficheros de datos (archivos planos) como es el caso de Weka, los cuales se cargan a la aplicación sin tener que interactuar con un gestor de bases de datos.

4.1.2 Sistema propuesto. MATE-KDD será una herramienta DCBD: selección, preprocesamiento, transformación, minería de datos y visualización, pero enfocada a la tarea de minería de datos Clasificación. En esta tarea, se utilizaran los algoritmos Mate-Tree, C4.4 y Sliq para clasificación. En la etapa de Visualización se contempla el desarrollo de una interfaz gráfica que le permita al usuario observar e interpretar los resultados de manera fácil.

El sistema MATE-KDD será medianamente acoplado con es SGBD PostgreSQL, permite conectarse con una Base de Datos determinada que se encuentre en Postgres, ver los atributos de una tabla seleccionada y sus características. El sistema puede aplicar funciones de preprocesamiento de datos como adicionar un atributo, resultado de una función matemática simple, Discretización de valores continuos, normalización, remover atributos, obtener un subconjunto de datos; el usuario (analista) selecciona el atributo clase así como la técnica de Minería de Datos a aplicar y define el porcentaje de división de los datos de entrenamiento y prueba, se puede escoger entre aplicar el algoritmos C4.4, el SLIQ o el MATE-TREE; para esto, antes el sistema aplica funciones de discretización si hay valores continuos y procesos de transformación de datos y luego crea el modelo de clasificación con el algoritmo escogido por el usuario, después de esto el sistema hace una transformación a los resultados para que el usuario los comprenda, hace una prueba del modelo creado e imprime las reglas y los resultados de la prueba, el sistema brinda una opción de visualización de forma gráfica del árbol, y el usuario puede guardar y consultar un modelo.

Metas del sistema. El objetivo principal del sistema es generar modelos de clasificación (reglas de decisión) aplicando el proceso de DCBD, utilizando técnicas de clasificación en la etapa de minería de datos y mostrar el modelo de forma gráfica. Y para esto debe:

- Permitir seleccionar, preprocesamiento y transformación de datos, además permita aplicar la minería de datos y visualización de resultados gráficamente.

- Aplicar los algoritmos de clasificación SLIQ, C4.4 y MATE-TREE.

- Almacenar y consultar modelos de clasificación.

Requerimientos no funcionales. La interfaz de usuario del sistema MATE-KDD es completamente gráfica, sencilla e intuitiva aún así, el usuario de este sistema debe tener conocimiento en Descubrimiento de Conocimiento en Bases de Datos y en el dominio de la aplicación para la correcta interpretación de los resultados. La herramienta debe estar en capacidad de manejar volúmenes de datos altos, y frente a éstos tener un buen rendimiento en cuanto a tiempo de procesamiento. El sistema debe presentar un modelo óptimo. Hay que tener en cuenta que el rendimiento de la herramienta depende básicamente de las características del hardware utilizado

El sistema debe comunicar en caso que no se pueda realizar la conexión con una base de datos específica, y por ende no puede permitir continuar con las fases de minería hasta tanto la conexión sea exitosa. Si se llegan a presentar casos en los cuales los datos no cumplan los requerimientos mínimos para aplicar un proceso específico el sistema debe informar y cancelar el proceso.

Por último, el software se escribirá usando el lenguaje de programación Java ya que la herramienta se desarrollara bajo los lineamientos de software libre, y para las funciones definidas por el usuario (FDU's) que se integraran en el SGBD se utilizara Pg/Sql que es un lenguaje procedural.

4.1.3 Funciones y atributos

- **Funciones para inicializar el sistema**

Ref#	Función	Categoría	Atributo	Detalle y Restricciones	Categoría
R1.1	Iniciar	Evidente	Tiempo de respuesta	5 segundos como máximo	Obligatoria.
			Metáfora de la interfaz	pantallas basadas en formas	Obligatoria.
				Colorido	Opcional.

- **Funciones para Selección de Datos**

R2.1	Conectar BD	Evidente	Tiempo de respuesta	5 segundos como máximo	Obligatoria.
			Metáfora de la interfaz	pantallas basadas en formas	Obligatoria.
				Colorido	Opcional.
			Tolerancia a fallos	La conexión se maneja en forma de transacciones atómicas.	Obligatoria.
R2.2	Mostrar Tablas	Evidente	Tiempo de respuesta	5 segundos como máximo	Obligatoria.
			Metáfora de la interfaz	pantallas basadas en formas	Obligatoria.
				Colorido	Opcional.
R2.3	Mostrar Información Tabla	Evidente	Tiempo de respuesta	5 segundos como máximo	Obligatoria.
			Metáfora de la interfaz	pantallas basadas en formas	Obligatoria.
				Colorido	Opcional.
R2.4	Guardar Tabla	Evidente	Tiempo de respuesta	5 segundos como máximo	Obligatoria.
			Metáfora de la interfaz	pantallas basadas en formas	Obligatoria.
				Colorido	Opcional.
R2.5	Adicionar Atributo	Evidente	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.

R2.6	Elimina Atributo	Evidente	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.
R2.7	Normalizar	Evidente	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.
R2.8	Discretizar	Evidente	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.
R2.9	Filtrar Tabla	Evidente	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.
R2.10	Seleccionar Registros	Evidente	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.
R2.11	Seleccionar Atributos	Evidente	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.

- **Funciones para preprocesamiento y transformación de datos**

R3.1	Depurar Tabla	Oculta	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.
R3.2	Transformar Tabla	Oculta	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.
R3.3	Dividir Tabla	Oculta	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.

- **Funciones para Crear Modelos de Clasificación**

R4.1	C4.5	Oculta	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.
R4.2	Matetree	Oculta	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.
R4.3	Sliq	Oculta	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.

- **Funciones para Validar e Interpretar el Modelo**

R5.1	Probar Modelo	Evidente	Metáfora de la interfaz	pantallas basadas en formas	Obligatoria.
				Colorido	Opcional.
			Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.
R5.2	Transformar Reglas	Oculta	Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.
R5.3	Interpretar Reglas	Evidente	Metáfora de la interfaz	pantallas basadas en formas	Obligatoria.
				Colorido	Opcional.
			Tolerancia a fallos	Se maneja en forma de transacciones atómicas.	Obligatoria.

- **Funciones para gráficar árbol**

R6.1	Ver árbol	Evidente	Metáfora de la Interfaz	Pantallas basadas en formas	Obligatorio
				Gráfico	Obligatorio
				Colorido	Opcional
			Tiempo de Respuesta	10 Segundos como máximo	Obligatorio

- **Funciones para manejar el modelo**

R7.1	Guardar Modelo	Evidente	Tiempo de respuesta	5 segundos como máximo	Obligatoria.
			Metáfora de la interfaz	pantallas basadas en formas	Obligatoria.
				Colorido	Opcional.
R7.2	Abrir Modelo	Evidente	Tiempo de respuesta	5 segundos como máximo	Obligatoria.
			Metáfora de la interfaz	pantallas basadas en formas	Obligatoria.

4.1.4 Casos de uso extendidos

- **Iniciar aplicación**

Caso de uso	**Iniciar Aplicación**
Objetivo	Iniciar el sistema y verificar que el SGBD este funcionando
Actor	Analista (iniciador)
Resumen	El analista solicita iniciar la aplicación. El sistema se verifica que el SGBD Postgres este inicializado.
Tipo	Primario y esencial
Referencias Cruzadas	Funciones: R1.1
Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista solicita iniciar el sistema.	2. Verifica que el SGBD Postgres este inicializado.
Cursos Alternos:	
Línea 2. Si el SGBD no esta inicializado. Indicar error	

- **Conectar Base de Datos**

Caso de uso	**Conectar Base de datos**
Objetivo	Conectarse a una base de datos en el SGBD Postgres
Actor	Analista (iniciador)
Resumen	El analista ingresa los datos de la base de datos y solicita conexión, al terminar la operación la aplicación esta conectada a la base de datos.
Tipo	Primario y esencial
Referencias Cruzadas	Funciones: R2.1 y R2.2
Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista ingresa al sistema los datos de la BD.	2. Tramita la conexión con la base de datos. Crea las funciones FDU en el SGBD 3. Postgres. Muestra la lista de tablas de la base de 4. datos
Casos AlternAtivos:	
Línea 2 Si no se pudo conectar Indicar error	

- **Abrir tabla**

Caso de uso	**Abrir Tabla**
Objetivo	Ver información de una tabla especifica de la base de datos
Actor	Analista (iniciador)
Resumen	El analista selecciona una tabla y el sistema muestra la información de la tabla.
Tipo	Primario y esencial
Referencias Cruzadas	Funciones: R2.3
Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el caso de uso Conectar Base de Datos lista las tablas de la base de datos. 2. El analista selecciona una tabla de la lista.	3. Muestra la información de la tabla seleccionada.
Cursos Alternos:	

- **Guardar tabla**

Caso de uso	Guardar Tabla
Objetivo	Guardar registros en una nueva tabla
Actor	Analista (iniciador)
Resumen	El analista ingresa el nombre para la tabla y el sistema la guarda en la BD.
Tipo	Primario y esencial
Referencias Cruzadas	Funciones: R2.4
Curso Normal de Eventos:	
Acción de los actores	Respuesta del sistema
1. Este caso de uso comienza cuando el analista ingresa a la opción guardar tabla. 2. Ingresa el nombre para la tabla a guardar.	3. Verifica que no exista una tabla con ese nombre. 4. Guarda la tabla en la BD actual. 5. Envía confirmación al analista.
Cursos Alternos:	
Línea 3 Si ya existe, Indicar error.	

- **Adicionar atributo**

Caso de uso	**Adicionar Atributo**
Objetivo	Adicionar un atributo en un conjunto de datos.
Actor	Analista (iniciador)
Resumen	El analista ingresa datos al sistema. Al finalizar el proceso se obtiene un nuevo conjunto de datos (tabla).
Tipo	Secundario y esencial
Referencias Cruzadas	Funciones: R2.3, R2.5 y R 2.6
Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista solicita adicionar atributo. 2. El analista ingresa datos al sistema. 4. Inicia aplicar	3. Valida los datos y arma la instrucción SQL. 5. Adiciona el nuevo atributo al conjunto de datos. 6. Actualiza en pantalla la información de la tabla.
Cursos Alternos:	
Línea 3 Si los datos no son validos. Indicar error	

- **Eliminar atributo**

Caso de uso	**Eliminar Atributo**
Objetivo	Eliminar un atributo de un conjunto de datos.
Actor	Analista (iniciador)
Resumen	El analista ingresa datos al sistema. Al finalizar el proceso se obtiene un nuevo conjunto de datos (tabla).
Tipo	Secundario y esencial
Referencias Cruzadas	Funciones: R2.3, R2.5 y R2.7
Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista solicita eliminar atributo. El analista ingresa datos al sistema. 2. Inicia aplicar 4.	3. Valida los datos y arma la instrucción SQL. Elimina el atributo del conjunto de datos. 5. Actualiza en pantalla la información de la tabla. 6.
Cursos Alternos:	
Línea 3 Si los datos no son validos. Indicar error	

- **Normalizar tabla**

Caso de uso	**Normalizar Tabla**
Objetivo	Normalizar los valores de un atributo o de todos los atributos de una tabla seleccionada.
Actor	Analista (iniciador)
Resumen	El analista ingresa datos al sistema. Al finalizar el proceso se obtiene un nuevo conjunto de datos (tabla).
Tipo	Secundario y esencial
Referencias Cruzadas	Funciones: R2.3, R2.5 y R2.8
Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista solicita normalizar. El analista ingresa datos al sistema. 2. Inicia aplicar 4.	3. Valida los datos y arma la instrucción SQL. Normaliza el o los atributos del conjunto de 5. datos. Actualiza en pantalla la información de la 6. tabla.
Cursos Alternos:	
Línea 3 Si los datos no son validos. Indicar error	

- **Discretizar tabla**

Caso de uso	**Discretizar Tabla**
Objetivo	Discretizar los valores de un atributo o de todos los atributos de una tabla seleccionada. Al finalizar el proceso se obtiene un nuevo conjunto de datos (tabla).
Actor	Analista (iniciador)
Resumen	El analista ingresa datos al sistema. Al finalizar el proceso se obtiene un nuevo conjunto de datos (tabla).
Tipo	Secundario y esencial
Referencias Cruzadas	Funciones: R2.3, R2.5 y R2.9
Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista solicita discretizar. 2. El analista ingresa datos al sistema. 4. Inicia aplicar	3. Valida los datos y arma la instrucción SQL. 5. Discretiza el o los atributos del conjunto de datos. 6. Actualiza en pantalla la información de la tabla.
Cursos Alternos:	
Línea 3 Si los datos no son validos. Indicar error	

- **Filtrar registros**

Caso de uso	**Filtrar Registros**
Objetivo	Seleccionar un conjunto de datos de la tabla que cumplan una condición en especial.
Actor	Analista (iniciador)
Resumen	El analista ingresa datos al sistema. Al finalizar el proceso se obtiene un nuevo conjunto de datos (tabla).
Tipo	Secundario y esencial
Referencias Cruzadas	Funciones: R2.3, R2.5 y R2.10
Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista solicita filtrar. 2. El analista ingresa datos al sistema. Inicia aplicar 4.	3. Valida los datos y arma la instrucción SQL. Filtra los registros de la tabla. 5. Actualiza en pantalla la información de la tabla. 6.
Cursos Alternos:	
Línea 3 Si los datos no son validos. Indicar error	

- **Seleccionar registros**

Caso de uso	**Seleccionar Registros**
Objetivo	Seleccionar un determinado número de registros (tuplas) de forma aleatoria.
Actor	Analista (iniciador)
Resumen	Selecciona aleatoriamente un grupo de tuplas. Al finalizar el proceso se obtiene un nuevo conjunto de datos (tabla).
Tipo	Secundario y esencial
Referencias Cruzadas	Funciones: R2.3, R2.5 y R2.11
Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista solicita seleccionar. El analista ingresa datos al sistema. 2. Inicia aplicar 4.	3. Valida los datos y arma la instrucción SQL. Selecciona un número de registros de la 5. tabla. Actualiza en pantalla la información de la 6. tabla.
Cursos Alternos:	
Línea 3 Si los datos no son validos. Indicar error	

- **Preprocesamiento de datos**

Caso de uso	**Preprocesamiento de Datos**
Objetivo	Obtener un conjunto de datos procesados para minar
Actor	Analista (iniciador)
Resumen	El sistema aplica a la tabla seleccionada procesos de depuración, discretización, transformación y división. Al terminar el proceso se obtiene un tabla lista para minar
Tipo	Primario y esencial
Referencias Cruzadas	Funciones: R3.1, R3.2, R3.3 y R3.4

Curso Normal de Eventos:

Acción de los actores	**Respuesta del sistema**
	1. Este caso de uso es inicializado por uno de los casos de uso que crean modelo de clasificación. Depura la tabla. 2. Discretiza datos 3. Transforma datos. 4. Divide los datos de la tabla y genera una tabla de 5. entrenamiento y una de prueba

Cursos Alternos:

Línea 2	Si no finalizo depuración correctamente, Indicar error
Línea 3	Si no finalizo discretización correctamente, Indicar error
Línea 4	Si no finalizo transformación proceso correctamente, Indicar error
Línea 5	Si no finalizo división correctamente, Indicar error

- **Validar modelo**

Caso de uso	**Validar Modelo**
Objetivo	Probar el modelo creado además, interpretar y mostrar resultados
Actor	Analista (iniciador)
Resumen	Se prueba el modelo de clasificación creado con los datos de prueba se transforman e interpretan las reglas de decisión y se muestran en pantalla.
Tipo	Secundario y esencial
Referencias Cruzadas	Funciones: R5.1, R5.2 y R5.3

Curso Normal de Eventos:

Acción de los actores	**Respuesta del sistema**
	1. Este caso de uso es inicializado por otro caso de uso. Prueba el modelo con la tabla de prueba y 2. muestra los resultados. Traduce las reglas del modelo 3. Interpreta y muestra las reglas del modelo 4.

Cursos Alternos:

- **Crear modelo C4.5**

Caso de uso	**Crear Modelo C4.5**
Objetivo	Obtener un modelo de clasificación con el algoritmo C4.5
Actor	Analista (iniciador)
Resumen	El analista selecciona el porcentaje de división de datos. El sistema crea el modelo y muestras las reglas obtenidas.
Tipo	Primario y esencial
Referencias Cruzadas	Funciones: R4.1 Casos de Uso: Preprocesamiento de datos y Validar Modelo

Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista solicita clasificar con C4.5 Ingresa datos al sistema 2.	3. Realiza el preprocesamiento de la tabla seleccionada. 4. Crea el modelo de clasificación C4.5 Valida el modelo obtenido 5. Muestra las reglas del modelo y el resultado de 6. la prueba.

Cursos Alternos:	
Línea 4	Si no se pudo crear el modelo. Indicar error

- **Crear modelo MateTree**

Caso de uso	**Crear Modelo MateTree**
Objetivo	Obtener el modelo de clasificación con el algoritmo Mate-tree
Actor	Analista (iniciador)
Resumen	El analista selecciona el porcentaje de división de datos. El sistema crea el modelo y muestras las reglas obtenidas.
Tipo	Primario y esencial
Referencias Cruzadas	Funciones: R4.2 Casos de Uso: Preprocesamiento de Datos y Validar Modelo

Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista solicita clasificar con Mate-Tree. Ingresa datos al sistema 2.	3. Realiza el preprocesamiento de la tabla seleccionada. 4. Crea el modelo de clasificación Mate-Tree. Valida el modelo obtenido 5. Muestra las reglas del modelo y el resultado de 6. la prueba.

Cursos Alternos:	
Línea 4	Si no se pudo crear el modelo. Indicar error

- **Crear modelo Sliq**

Caso de uso	**Crear Modelo Sliq**
Objetivo	Obtener un modelo de clasificación con el algoritmo Slip
Actor	Analista (iniciador)
Resumen	El analista selecciona el porcentaje de división de datos. El sistema crea el modelo y muestras las reglas obtenidas.
Tipo	Primario y esencial
Referencias Cruzadas	Funciones: R3.1, R3.4 y R4.3 Casos de Uso: Validar Modelo

Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista solicita clasificar con Sliq. Ingresa datos al sistema 2.	3. Depura los registros de la tabla seleccionada. 4. Divide los registros de la tabla (entrenamiento y prueba) 5. Crea el modelo de clasificación Sliq 6. Valida el modelo obtenido. 7. Muestra las reglas del modelo y los resultados de la prueba.
Cursos Alternos:	
Línea 5 Si no se pudo crear el modelo. Indicar error	

- **Ver árbol de decisión**

Caso de uso	**Ver Árbol de Decisión**
Objetivo	Graficar en pantalla el árbol de clasificación del modelo tratado
Actor	Analista (iniciador)
Resumen	El sistema Gráfica el árbol de decisión en pantalla.
Tipo	Secundario y esencial
Referencias Cruzadas	Funciones: R6.1

Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista solicita ver árbol de decisión	2. Gráfica el árbol de decisión en pantalla.
Cursos Alternos:	

- **Guardar modelo de clasificación**

Caso de uso	**Guardar Modelo de Clasificación**
Objetivo	Guardar el modelo de clasificación creado
Actor	Analista (iniciador)
Resumen	El analista ingresa el nombre para el modelo y el sistemas lo guarda.
Tipo	Secundario y esencial
Referencias Cruzadas	Funciones: R7.1

Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista ingresa a la opción guardar modelo	
2. Ingresa el nombre para el modelo.	3. Verifica que no exista un archivo con ese nombre. 4. Guarda el modelo 5. Envía confirmación al analista.
Cursos Alternos:	
Línea 3 No se logro guardar. Indicar error	

- **Abrir modelo de clasificación**

	Abrir Modelo de Clasificación
Objetivo	Ver información de un modelo de clasificación guardado
Actor	Analista (iniciador)
Resumen	El analista escribe el nombre del modelo. El sistema muestra en pantalla las reglas del modelo
Tipo	Secundario y esencial
Referencias Cruzadas	Funciones: R5.3 y R7.2

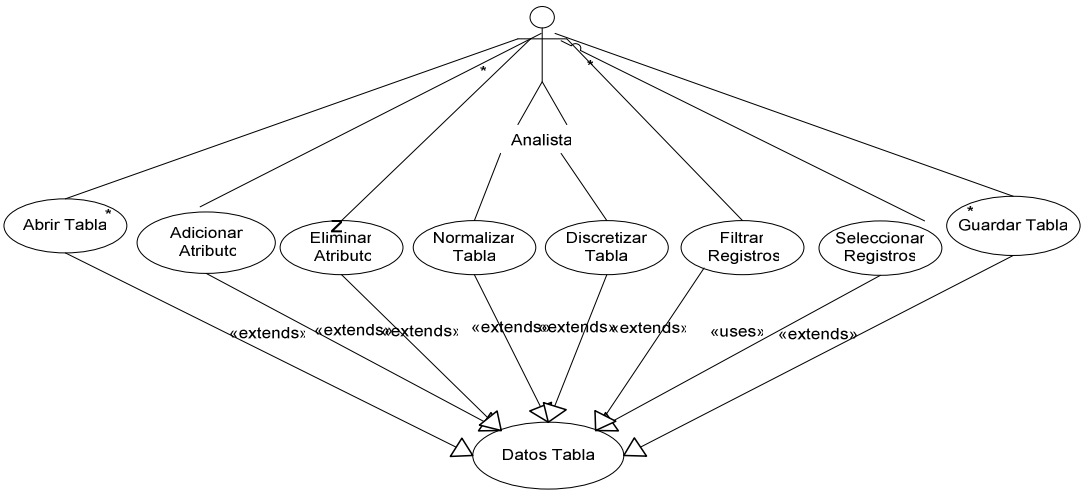
Curso Normal de Eventos:	
Acción de los actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el analista ingresa a la opción abrir modelo.	
2. Ingresa el nombre del modelo.	3. Busca el modelo 4. Interpreta las reglas del modelo 5. Muestra las reglas de clasificación.
Cursos Alternos:	
Línea 3. El modelo no existe. Indicar error	

4.1.5 Diagramas de casos de uso

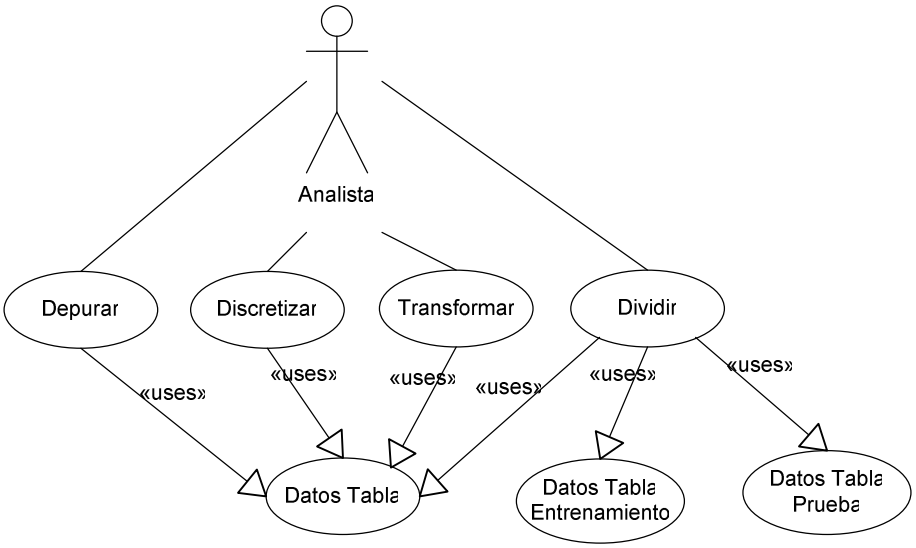
- **Diagrama caso de uso iniciar aplicación**

- **Diagrama caso de uso conectar base de datos**

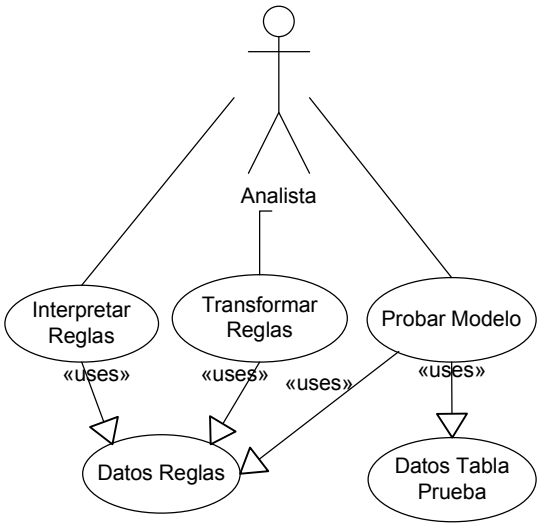
- **Diagrama caso de uso manejar tabla**



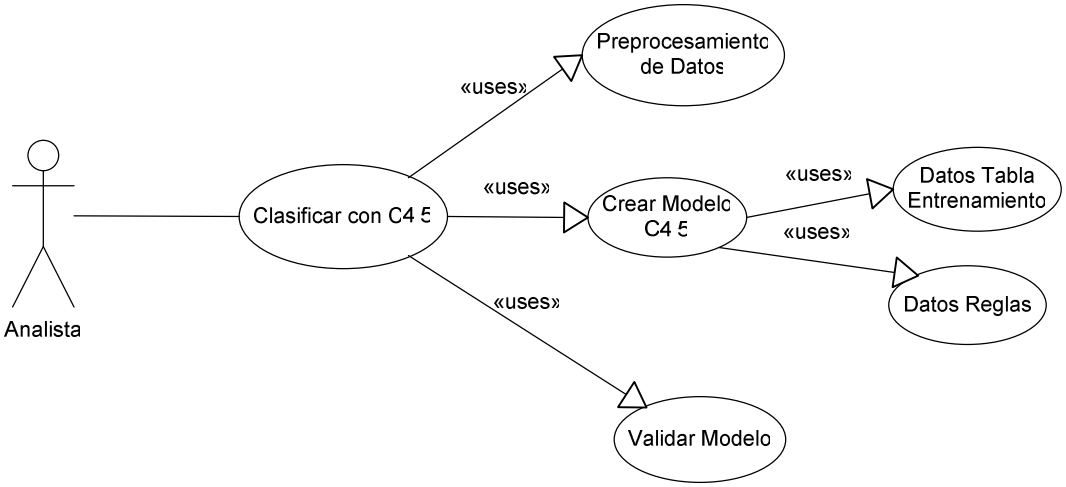
- **Diagrama caso de uso preprocesamiento de datos**



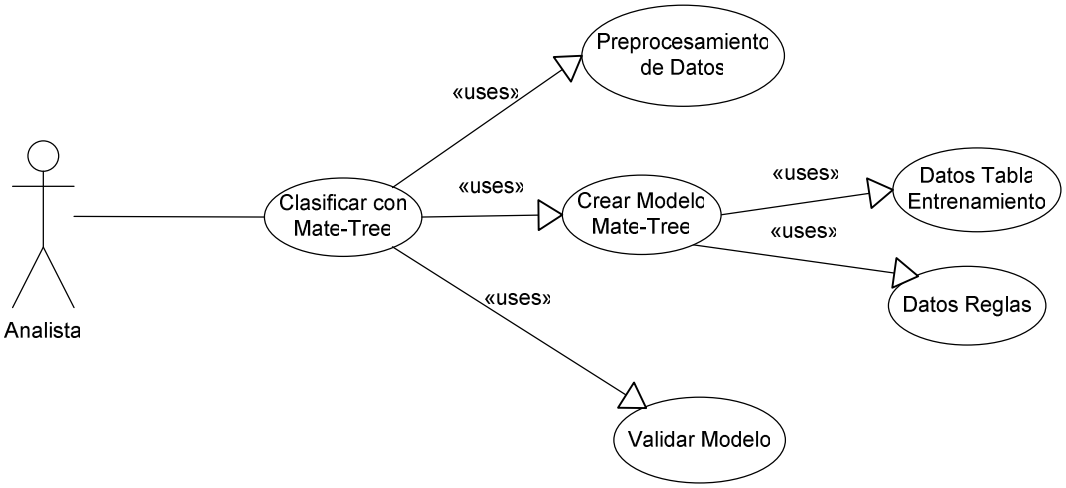
- **Diagrama caso de uso validar modelo**



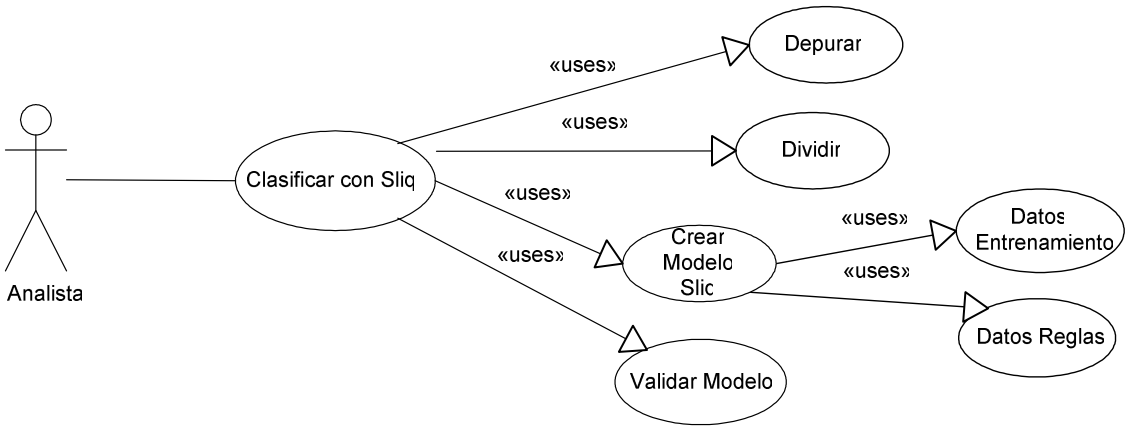
- **Diagrama caso de uso clasificar con C4.5**



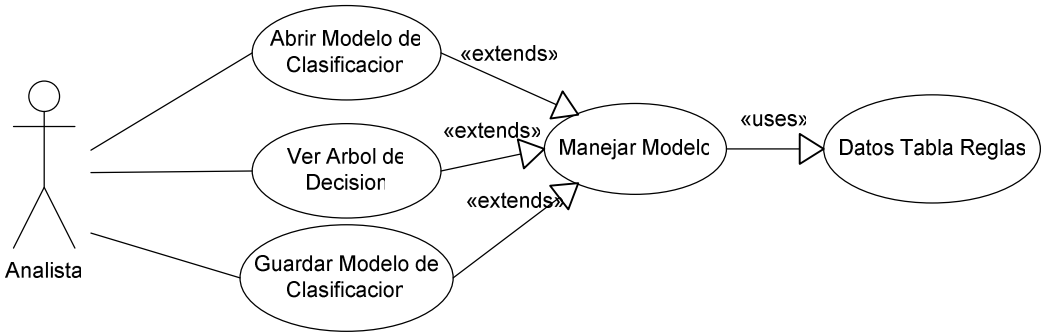
- **Diagrama caso de uso clasificar con Mate-Tree**



- **Diagrama caso de uso clasificar con Sliq**

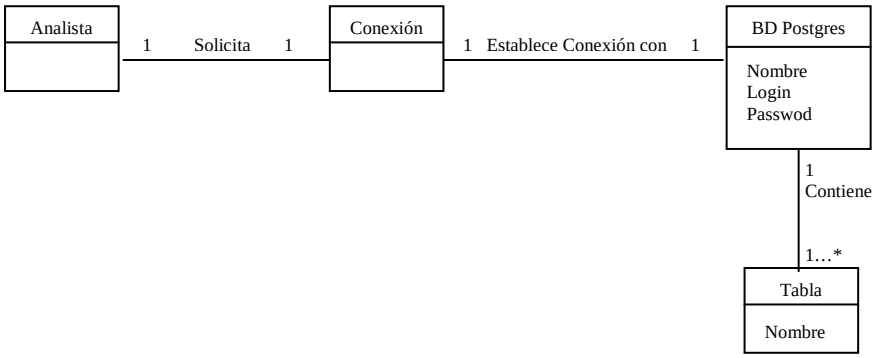


- **Diagrama caso de uso manejar modelo**

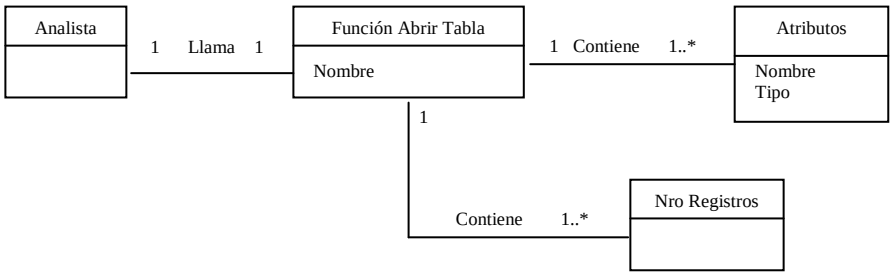


4.1.6 Modelo conceptual

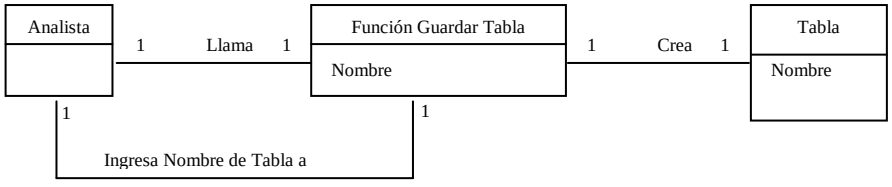
- **Conectar BD**



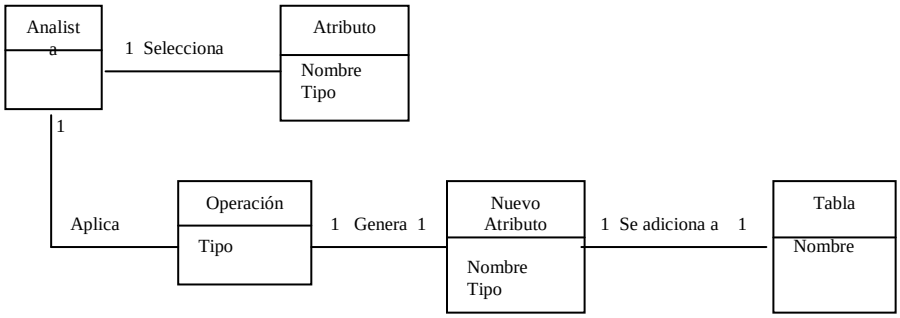
- **Abrir tabla**



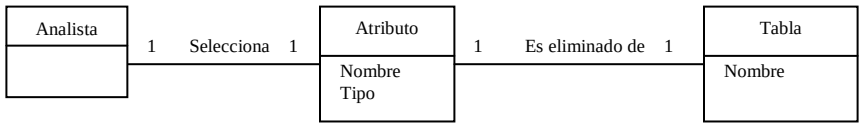
- **Guardar tabla**



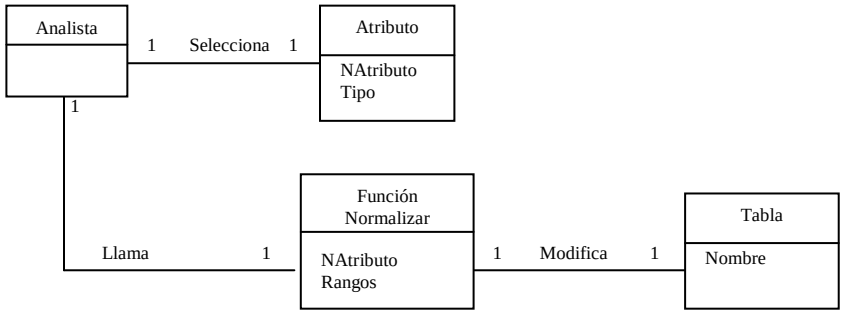
- **Adicionar atributo**



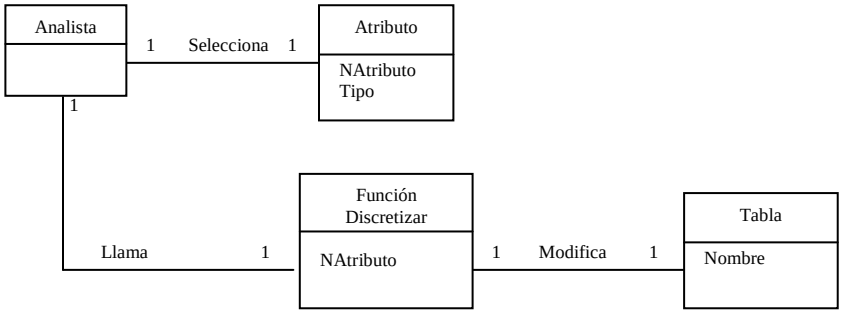
- **Eliminar atributo**



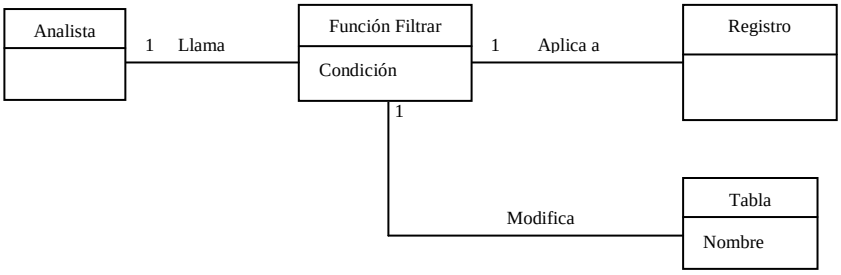
- **Normalizar atributo**



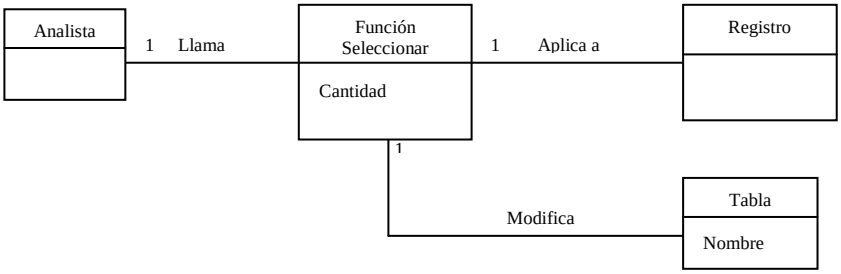
- **Discretizar**



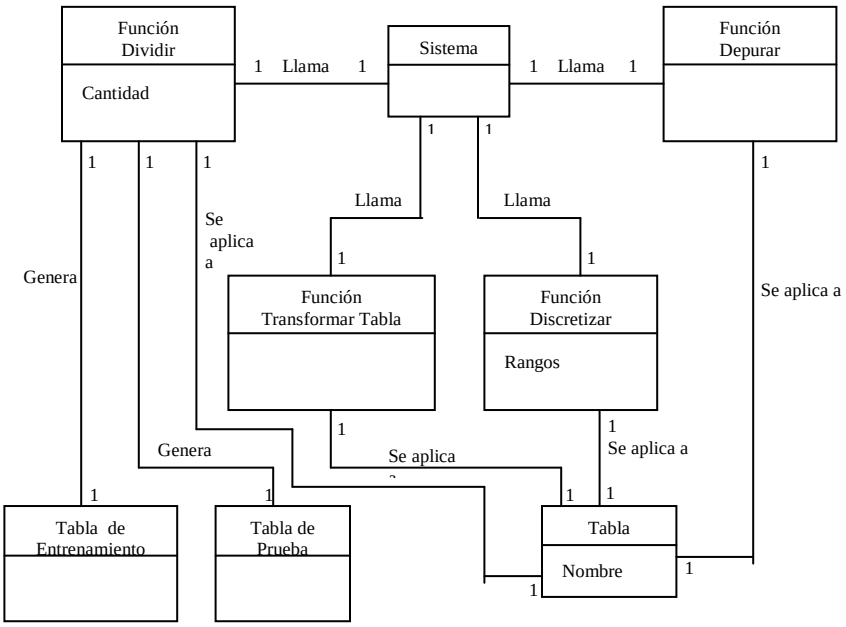
- **Filtrar registros**



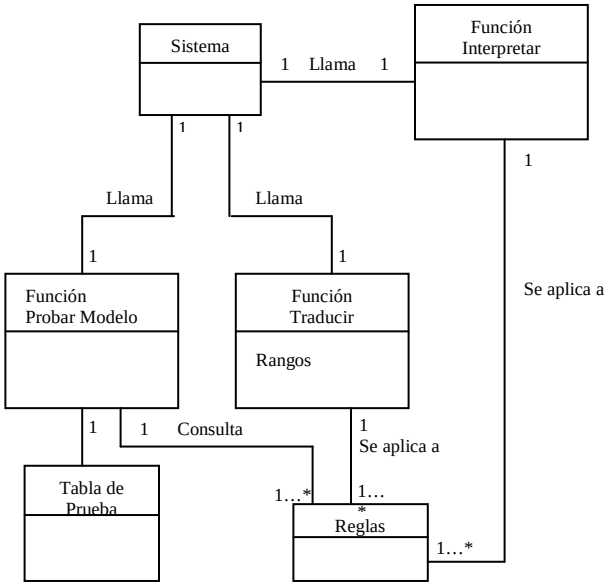
- **Seleccionar registros**



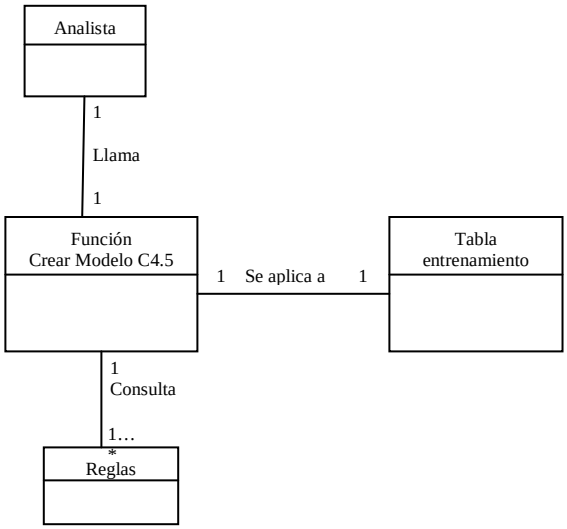
- **Procesamiento de Datos**



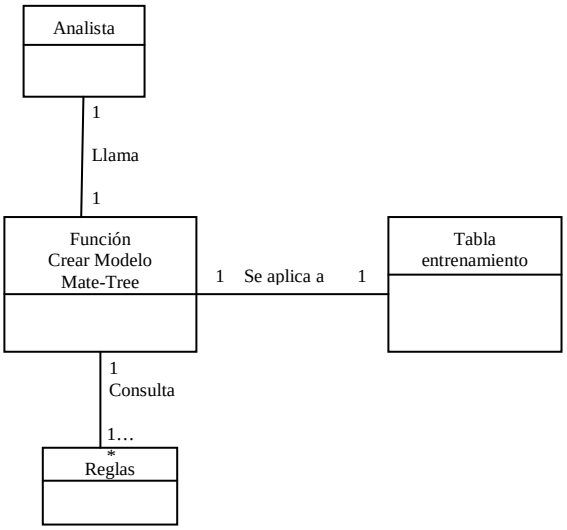
- **Validar modelo**



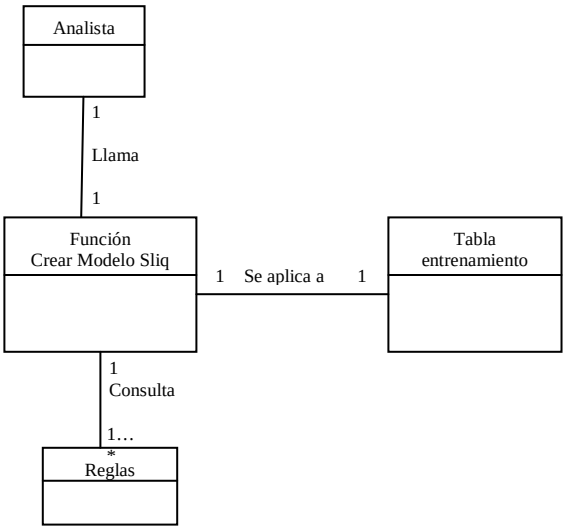
- **Crear modelo C4.5**



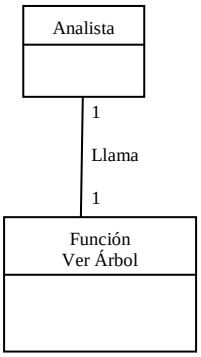
- **Crear modelo Mate-Tree**



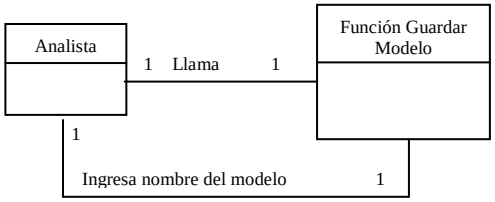
- **Crear modelo Sliq**



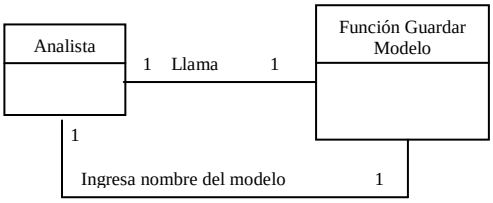
- **Ver árbol de decisión**



- **Guardar modelo de clasificación**

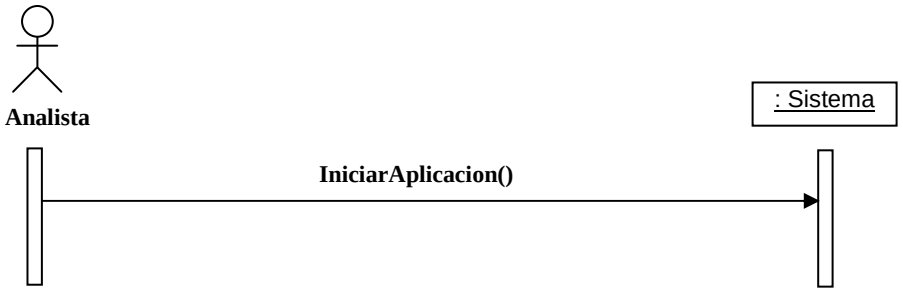


- **Abrir modelo de clasificación**

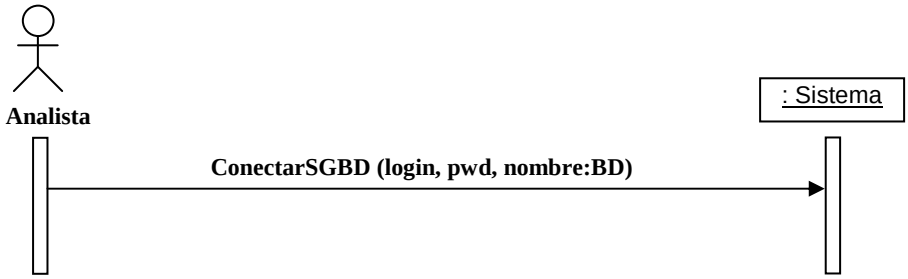


4.1.7 Diagramas de secuencia

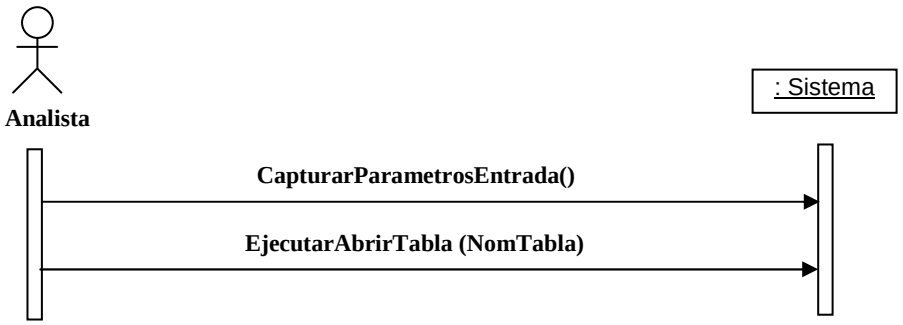
- **Iniciar aplicación**



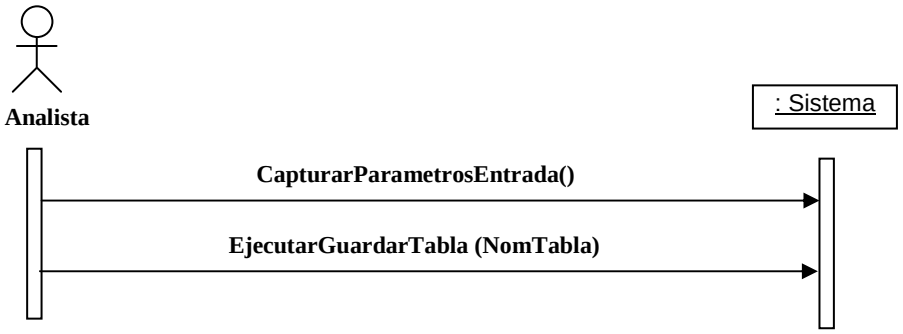
- **Conectar base de datos**



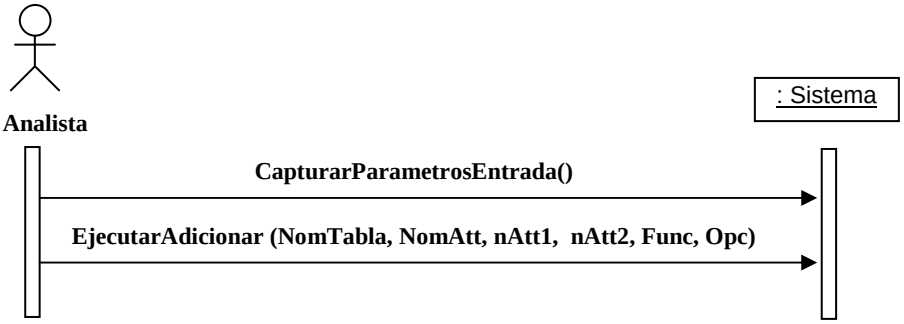
- **Abrir tabla**



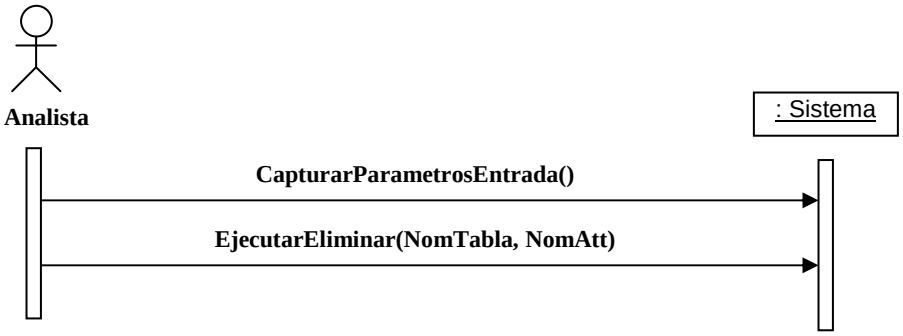
- **Guardar tabla**



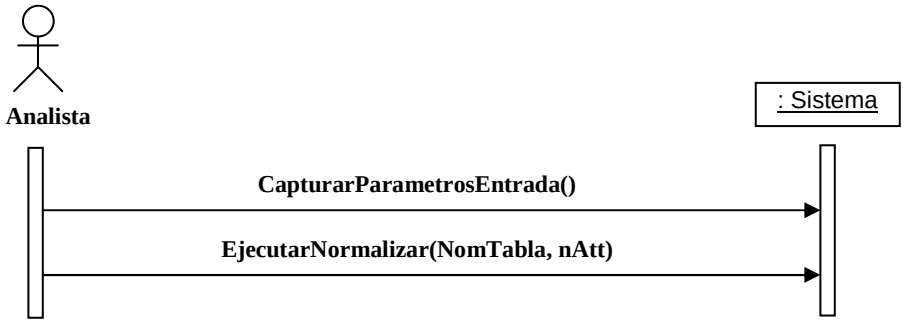
- **Adicionar atributo**



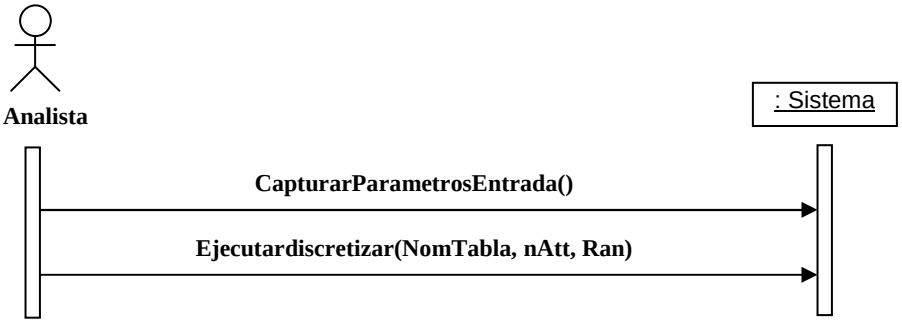
- **Eliminar atributo**



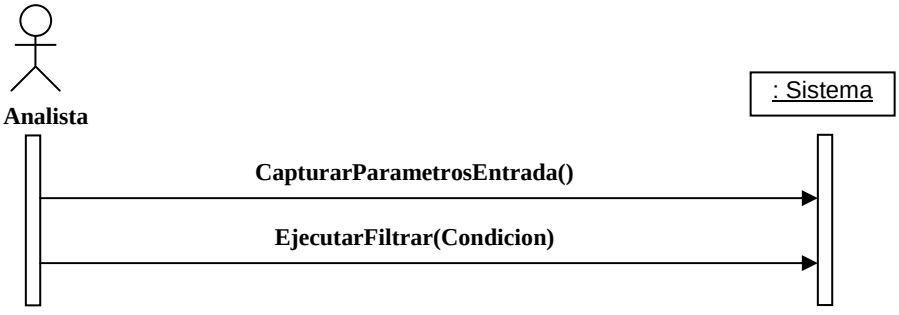
- **Normalizar tabla**



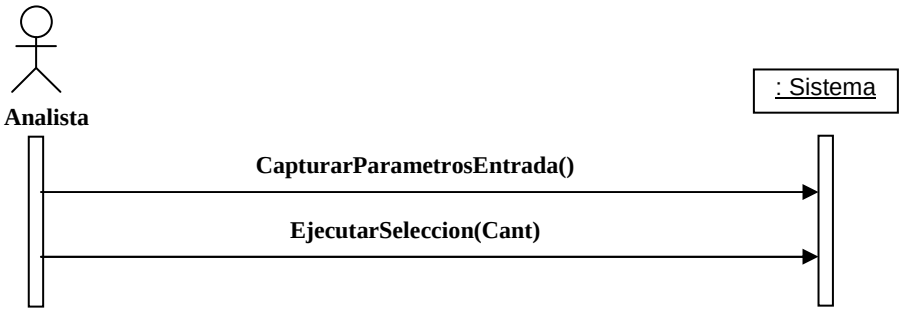
- **Discretizar**



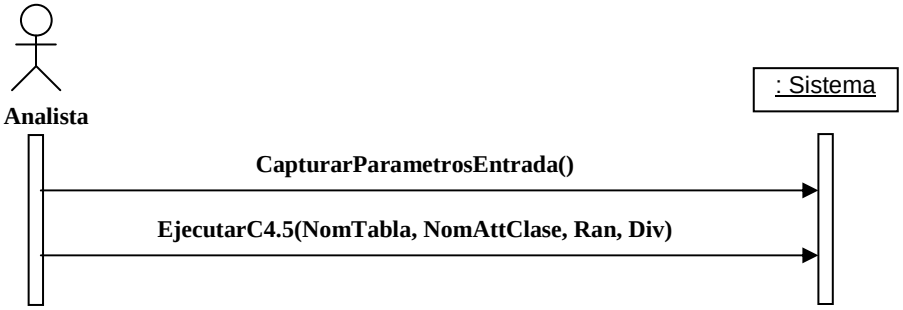
- **Filtrar registros**



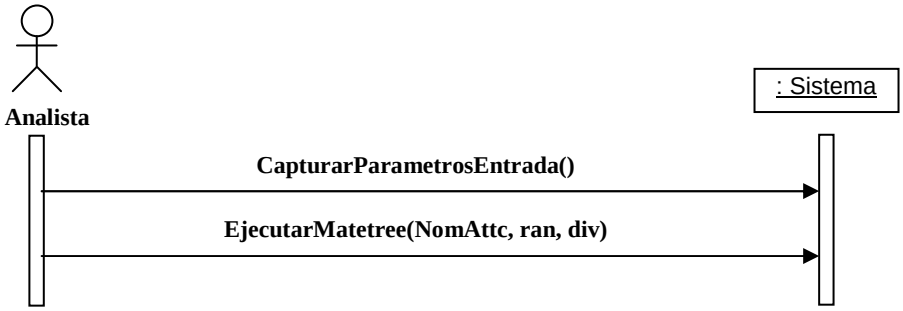
- **Seleccionar registros**



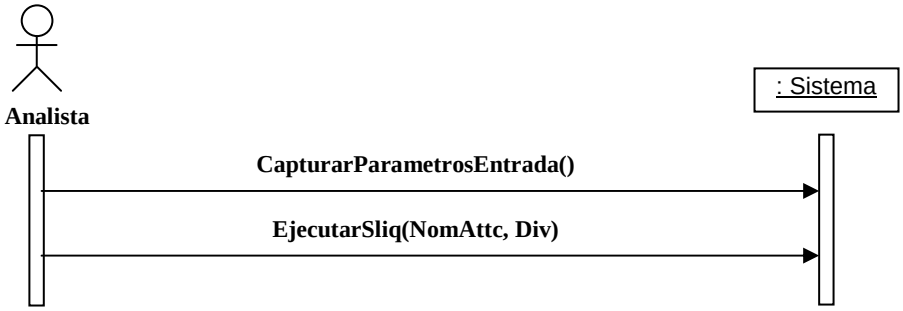
- **Crear modelo C4.5**



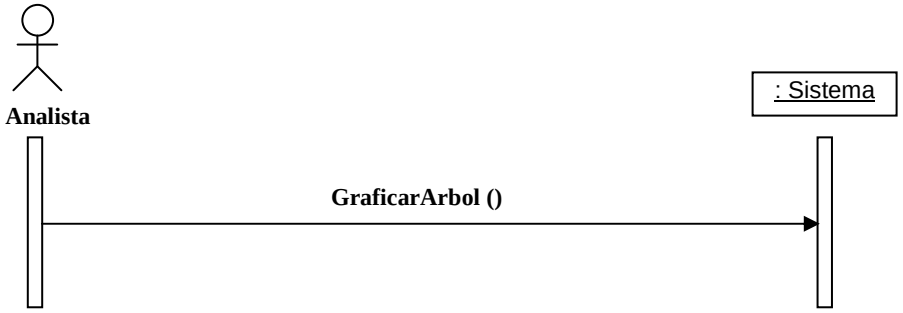
- **Crear modelo Mate-Tree**



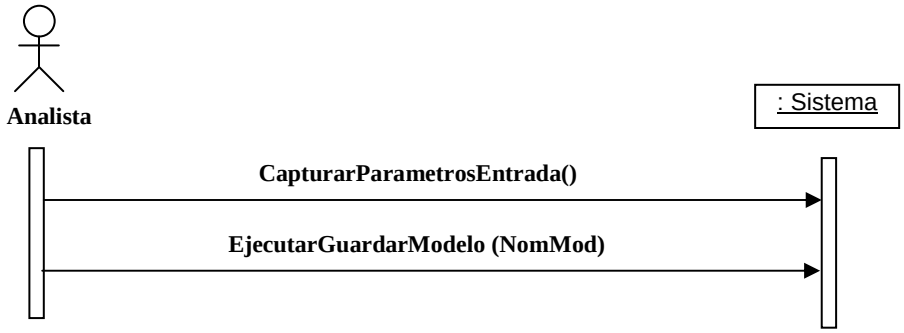
- **Crear modelo Sliq**



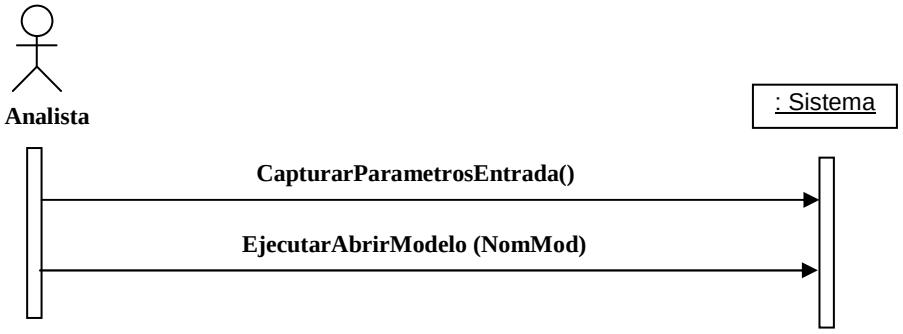
- **Ver Árbol de Decisión**



- **Guardar modelo de clasificación**



- **Abrir modelo de clasificación**



4.1.8 Contratos

- **Iniciar aplicación**

Nombre	**iniciarAplicacion()**
Responsabilidades	Iniciar el sistema y verificar que el SGBD Postgres este inicializado.
Tipo	Sistema
Referencias cruzadas	Funciones: R1.1 Casos de uso: Iniciar Aplicación
Notas	
Excepciones	Si el SGBD Postgres no esta inicializado, indique error y termine el sistema.
Salida	
Precondiciones	El SGBD Postgres debe estar inicializado
Poscondiciones	• Fue creada una instancia del sistema.

- **Conectar SGBD**

Nombre	**ConectarSGBD (login: carácter, pwd: carácter, NombreBD: carácter)**
Responsabilidades	Se conecta con una base de datos en Postgres. Crea las FDU en la base de datos y despliega una lista con los nombre de las tablas que integran la base de datos.
Tipo	Sistema
Referencias cruzadas	Funciones: R2.1 y R2.2 Casos de uso: Conectar Base de Datos
Notas	
Excepciones	Si no se puede conectar a la base de datos, indique que se cometió un error.
Salida	
Precondiciones	Debe estar inicializado el SGBD Postgres.
Poscondiciones	• Fue creada una instancia de conexión con la base de datos indicada.

- **Capturar parámetros entrada**

Nombre	**CapturarParametrosEntrada()**
Responsabilidades	Capturar y validar la información que se necesite para los procesos posteriores.
Tipo	Sistema
Referencias cruzadas	Funciones: R 2.1 – R 2.9, R 4.1 – R 4.3 y R 7.1 – R 7.
Notas	Este contrato es el mismo para todas las funciones mencionadas.
Excepciones	Informar sobre alguna inconsistencia en la validación de los parámetros.
Salida	
Precondiciones	
Poscondiciones	

- **Ejecutar abrir tabla**

Nombre	**EjecutarAbrirTabla(NomTabla: carácter)**
Responsabilidades	Muestra el número de registros que contiene la tabla seleccionada así como el nombre y el tipo de los atributos que la integran.
Tipo	Sistema
Referencias cruzadas	Funciones: R2.3 Casos de uso: Abrir Tabla
Notas	
Excepciones	
Salida	Información de la tabla escogida
Precondiciones	El sistema debe tener información de las tablas de la base de datos
Poscondiciones	

- **Ejecutar guardar tabla**

Nombre	**EjecutarGuardarTabla(NomTabla: carácter)**
Responsabilidades	Guardar una tabla en la BD.
Tipo	Sistema
Referencias cruzadas	Funciones: R2.4 Casos de uso: Guardar Tabla
Notas	
Excepciones	
Salida	Mensaje de confirmación
Precondiciones	El sistema debe tener información de una tabla actual.
Poscondiciones	• Fue creada una tabla (Creación de instancia)

- **Ejecutar adicionar**

Nombre	**EjecutarAdicionar(NomTabla: carácter, NomAtt: carácter, nAtt1: entero, nAtt2: entero, func: carácter, opc: entero)**
Responsabilidades	Adiciona un nuevo atributo a la tabla en la que se esta trabajando con el resultado de la operación seleccionada y actualiza la información de la tabla en pantalla.
Tipo	Sistema
Referencias cruzadas	Funciones: R2.3, R2.5 y R2.6 Casos de uso: Adicionar Atributo
Notas	
Excepciones	Si la operación no es realizable, indicar que se cometió un error
Salida	Modifica información de la tabla en pantalla.
Precondiciones	El sistema debe conocer los atributos que componen la tabla, su tipo y las posibles operaciones a realizarse.
Poscondiciones	• Fue creada una tabla (Creación de instancia)

- **Ejecutar eliminar**

Nombre	**EjecutarEliminar(NomTabla: carácter, nAtt: entero)**
Responsabilidades	Verifica que el atributo escogido no sea el atributo clase y pide confirmación
Tipo	Sistema
Referencias cruzadas	Funciones: R2.3, R,5 y R2.7 Casos de uso: Eliminar Atributo
Notas	
Excepciones	Si el atributo es el Atributo clase. Indicar error.
Salida	Pide confirmación para eliminar el atributo
Precondiciones	El sistema debe conocer los atributos que componen la tabla actual
Poscondiciones	• Fue creada una tabla (Creación de instancia)

- **Ejecutar normalizar**

Nombre	**EjecutarNormalizar(NomTabla: carácter, nAtt: entero)**
Responsabilidades	Normalizar el atributo seleccionado o toda la tabla y actualiza la información de la tabla en pantalla.
Tipo	Sistema
Referencias cruzadas	Funciones: R2.3, R2.5 y R2.8 Casos de uso: Normalizar Tabla
Notas	
Excepciones	Si el atributo seleccionado no es numérico (base 10), indique se cometió un error
Salida	Actualiza información de la tabla en pantalla
Precondiciones	El sistema debe conocer el tipo y los atributos que componen la tabla actual
Poscondiciones	• Fue creada una tabla (Creación de instancia)

- **Ejecutar discretizar**

Nombre	**EjecutarDiscretizar(NomTabla: carácter, nAtt: entero, ran: entero)**
Responsabilidades	Discretiza el atributo seleccionado o toda la tabla y actualiza la información de la tabla en pantalla.
Tipo	Sistema
Referencias cruzadas	Funciones: R2.3, R2.5 y R2.9 Casos de uso: Discretizar
Notas	
Excepciones	Si el atributo seleccionado no es numérico (base 10), indique se cometió un error
Salida	Actualiza información de la tabla en pantalla
Precondiciones	El sistema debe conocer el tipo y los atributos que componen la tabla actual
Poscondiciones	• Fue creada una tabla (Creación de instancia)

- **Ejecutar filtrar**

Nombre	**EjecutarFiltrar(Condición: carácter)**
Responsabilidades	Filtra la tabla de acuerdo a la condición ingresada y actualiza la información que hay en la pantalla de la tabla.
Tipo	Sistema
Referencias cruzadas	Funciones: R2.3, R2.5 y R2.10 Casos de uso: Filtrar Registros
Notas	
Excepciones	Si la condición no es valida, indicar que se cometió un error.
Salida	Muestra información de la tabla en pantalla
Precondiciones	El sistema debe conocer el tipo y los atributos que componen la tabla actual
Poscondiciones	• Fue creada una tabla (Creación de instancia)

- **Ejecutar seleccionar**

Nombre	**EjecutarSeleccionar(Div: entero)**
Responsabilidades	Selecciona un grupo de registros para la tabla y actualiza la información de la misma en pantalla.
Tipo	Sistema
Referencias cruzadas	Funciones: R2.3, R2.5 y R2.11
Notas	
Excepciones	Si la cantidad de registros indicada es mayor que la cantidad de registros existente, indicar que se cometió un error.
Salida	Muestra información de la tabla en pantalla
Precondiciones	El sistema debe conocer el número de registros que componen la tabla actual
Poscondiciones	• Fue creada una tabla (Creación de instancia)

- **Ejecutar C45**

Nombre	**EjecutarC4.5 (NomAttClase: carácter, Ran: entero, Div: entero)**
Responsabilidades	Hace el preprocesamiento de datos: Depuración y discretiza los registros; hace la transformación de los registros; Divide los registros en dos grupos (de entrenamiento y de prueba); Crea el modelo de decisión con el algoritmo C4.5. Prueba el modelo obtenido con los registros de prueba; Traduce las reglas del modelo; Interpreta y muestra las reglas y el resultado de la prueba.
Tipo	Sistema
Referencias cruzadas	Funciones: R3.1, R3.2, R3.3, R3.4, R4.1, R5.1, R5.2 y R5.3 Casos de uso: Preprocesamiento de Datos, Validar Modelo y Crear Modelo C45
Notas	
Excepciones	Si no se puede crear el modelo, indicar que se cometió un error.
Salida	Muestra en las reglas obtenidas y el resultado de la prueba
Precondiciones	El haber seleccionada una tabla y un atributo clase.
Poscondiciones	• Fue creada una tabla de reglas del modelo (creación de instancia) • Fue creada una tabla de nodos del modelo (creación de instancia)

- **Ejecutar Mate**

Nombre	**EjecutarMate (NomAttClase: carácter, Ran: entero, Div: entero)**
Responsabilidades	Hace el preprocesamiento de datos: Depuración y discretiza los registros; hace la transformación de los registros; Divide los registros en dos grupos (de entrenamiento y de prueba); Crea el modelo de decisión con el algoritmo Mate-Tree. Prueba el modelo obtenido con los registros de prueba; Traduce las reglas del modelo; Interpreta y muestra las reglas y el resultado de la prueba.
Tipo	Sistema
Referencias cruzadas	Funciones: R3.1, R3.2, R3.3, R3.4, R4.1, R5.1, R5.2 y R5.3 Casos de uso: Preprocesamiento de Datos, Validar Modelo y Crear Modelo Mate-Tree
Notas	
Excepciones	Si no se puede crear el modelo, indicar que se cometió un error.
Salida	Muestra en las reglas obtenidas y el resultado de la prueba
Precondiciones	El haber seleccionada una tabla y un atributo clase.
Poscondiciones	• Fue creada una tabla de reglas del modelo (creación de instancia) • Fue creada una tabla de nodos del modelo (creación de instancia)

- **Ejecutar Sliq**

Nombre	**EjecutarSliq (NomAttClase: Carácter, Div: entero)**
Responsabilidades	Hace la depuración y divide de registros. Crea el modelo de decisión con el algoritmo Sliq Prueba el modelo obtenido con los registros de prueba; Traduce las reglas del modelo; Interpreta y muestra las reglas y el resultado de la prueba.
Tipo	Sistema
Referencias cruzadas	Funciones: R3.1, R3.4, R4.1.3, R5.1, R5.2 y R5.3 Casos de uso: Validar Modelo y Crear Modelo Mate-Tree
Notas	
Excepciones	Si no se puede crear el modelo, indicar que se cometió un error.
Salida	Muestra en las reglas obtenidas y el resultado de la prueba
Precondiciones	El haber seleccionada una tabla y un atributo clase.
Poscondiciones	• Fue creada una tabla de reglas del modelo (creación de instancia) • Fue creada una tabla de nodos del modelo (creación de instancia)

- **Ver árbol**

Nombre	**VerArbol()**
Responsabilidades	Gráfica en pantalla el árbol de clasificación del modelo de decisión actual.
Tipo	Sistema
Referencias cruzadas	Funciones: R6.1 Casos de uso: Ver Árbol de Decisión
Notas	
Excepciones	Si no hay modelo actual, indicar que se cometió un error
Salida	Árbol grafico en pantalla
Precondiciones	Tener seleccionado un modelo.
Poscondiciones	

- **Ejecutar guardar modelo**

Nombre	**EjecutarGuardarModelo(NomMod: carácter)**
Responsabilidades	Guardar las reglas del modelo de clasificación.
Tipo	Sistema
Referencias cruzadas	Funciones: R7.1 Casos de uso: Guardar Modelo de Clasificación
Notas	
Excepciones	Si no hay modelo actual, indicar que se cometió un error
Salida	Envía mensaje de confirmación al analista
Precondiciones	Tener seleccionado un modelo.
Poscondiciones	• Fue creada una tabla de reglas del modelo (creación de instancia) • Fue creada una tabla de nodos del modelo (creación de instancia)

- **Ejecutar abrir modelo**

Nombre	**EjecutarAbrirModelo(NomMod: carácter)**
Responsabilidades	Mostrar las reglas del modelo de clasificación seleccionado.
Tipo	Sistema
Referencias cruzadas	Funciones: R5.3 y R7.2 Casos de uso: Abrir Modelo de Clasificación
Notas	
Excepciones	Si el nombre del modelo no existe, indicar que se cometió un error
Salida	Muestra las reglas del modelo
Precondiciones	El sistema debe conocer los modelos almacenados
Poscondiciones	

4.2 Glosario

Termino	Categoría	Comentarios
Abrir Modelo de Clasificación	Caso de Uso	Descripción del proceso donde el analista abre un modelo de clasificación.
Adicionar Atributo	Caso de Uso	Descripción del proceso donde el analista agrega un atributo en una tabla resultado de una operación.
Analista	Concepto	Un usuario del sistema
Árbol Grafico	Concepto	Un dibujo en forma de árbol.
Atributo	Concepto	Una columna de una tabla
Atributo Numérico	Concepto	Una columna de tipo numérico en una tabla
BD Postgres	Concepto	Una base de datos perteneciente al SGBD Postgres.
Conectar a BD	Caso de Uso	Descripción del proceso donde el analista se conecta con una base de datos en Postgres.
Conexión	Concepto	Conexión a una base de datos.
Crear Modelo C4.1.4	Caso de Uso	Descripción del proceso donde el analista solicita se cree el modelo de clasificación usando el algoritmo C4.1.4.1.

Crear Modelo Mate-Tree	Caso de Uso	Descripción del proceso donde el analista solicita se cree el modelo de clasificación usando el algoritmo Mate-Tree
Crear Modelo Sliq	Caso de Uso	Descripción del proceso donde el analista solicita se cree el modelo de clasificación usando el algoritmo Sliq.
Eliminar Atributo	Caso de Uso	Descripción del proceso donde el analista elimina un atributo de una tabla.
escogerTabla.Nombretabla: carácter	Atributo	Nombre de la tabla de cual se solicita ver información.
Filtrar Registros	Caso de Uso	Descripción del proceso donde se escoge un los registros de una tabla que cumplen una condición especifica.
Función Abrir Modelo	Concepto	Un Proceso en donde se abre un modelo de clasificación que este guardado.
Función Crear Modelo C45	Concepto	Un Proceso que crea un modelo de clasificación usando el algoritmo C4.5
Función Crear Modelo Mate-Tree	Concepto	Un Proceso que crea un modelo de clasificación usando el algoritmoMate-Tree.
Función Crear Modelo Sqli	Concepto	Un Proceso que crea un modelo de clasificación usando el algoritmo Sliq.
Función Depurar	Concepto	Un proceso que depura los datos de una tabla.
Función Discretizar	Concepto	Un proceso que discretiza los datos de las columnas de una tabla.
Función Dividir	Concepto	Un proceso que divide los registros de una tabla creando dos tablas (de entrenamiento y de prueba)
Función Filtrar	Concepto	Un proceso que selecciona los registros de una tabla que cumplen una condición especifica.
Función Guardar Modelo	Concepto	Un proceso en donde se guarda en BD un modelo creado.
Función Guardar Tabla	Concepto	Un proceso en donde se guarda en BD una tabla.
Función Interpretar	Concepto	Un proceso que interpreta las reglas de un modelo.
Función Normalizar	Concepto	Un proceso que normaliza los valores de una columna de una tabla.
Función Probar	Concepto	Un proceso que prueba las reglas de un modelo de clasificación.
Función Seleccionar	Concepto	Un proceso que escoge un número determinado de registros de una tabla.
Función Traducir	Concepto	Un proceso que transforma los valores de las reglas generadas.
Función Ver Árbol	Concepto	Un proceso que hace el dibujo de un árbol de decisión en pantalla.
Guardar Modelo de Clasificación	Caso de Uso	Descripción del proceso donde el analista guarda un modelo de clasificación creado.
Guardar Tabla	Caso de Uso	Descripción del proceso donde el analista guarda una tabla en una base de datos.
ingresarCantidadRegistros.Cantidad: entero	Atributo	Cantidad de registros que se deben seleccionar para el nuevo conjunto de datos.
ingresarDatosFiltrar.Condición: carácter	Atributo	Condición para filtrar los registros de una tabla.
ingresarDatosModeloC45 .NatributoC: carácter	Atributo	Nombre del atributo clase dispuesto para una tabla.

ingresarDatosModeloC45.NumeroD: entero	Atributo	Número de registros que debe contener la tabla de entrenamiento.
ingresarDatosModeloC45.Rangos: entero	Atributo	Número de rangos dispuestos para la discretización de los atributos numéricos de una tabla.
ingresarDatosModeloMate .NatributoC: carácter	Atributo	Nombre del atributo clase dispuesto para una tabla.
ingresarDatosModeloMate.NumeroD: entero	Atributo	Número de registros que debe contener la tabla de entrenamiento.
ingresarDatosModeloMate.Rangos: entero	Atributo	Número de rangos dispuestos para la discretización de los atributos numéricos de una tabla.
ingresarDatosModeloSliq .NatributoC: carácter	Atributo	Nombre del atributo clase dispuesto para una tabla.
ingresarDatosModeloSliq.NumeroD: entero	Atributo	Número de registros que debe contener la tabla de entrenamiento.
ingresarDatosNormalizar.Natributo: carácter	Atributo	Nombre de la columna a la que se aplicara proceso de normalización.
ingresarDatosNormalizar.Rangos: entero	Atributo	Número de rangos dispuestos para la normalización de una columna.
ingresarNombreBD.NombreBD: carácter	Atributo	Nombre de la base de datos con la cual se va a tramitar la conexión.
ingresarNombreModeloAbrir.Nombremd: carácter	Atributo	Nombre del modelo de clasificación que se va a guardar.
ingresarNombreModeloGuardar.Nombremd: carácter	Atributo	Nombre del modelo de clasificación que se va a abrir
ingresarNombreTabla.Nombretabla: carácter	Atributo	Nombre que se le asigna a la tabla a guardar en la BD.
IngresarOperación.Natributo1: carácter	Atributo	Nombre del primer atributo para operación adicionar.
IngresarOperación.Natributo2: carácter	Atributo	Nombre del segundo atributo para operación adicionar.
IngresarOperación.Posición: entero	Atributo	Posición dentro de la tabla en donde se va a adicionar el nuevo atributo.
IngresarOperación.Tipo: entero	Atributo	Código de la operación matemática a realizar con los atributos seleccionados.
Iniciar Sistema	Caso de Uso	Descripción del proceso donde el analista inicia la herramienta MATE-KDD.
Normalizar Atributo	Caso de Uso	Descripción del proceso donde el analista modifica una tabla con la operación de normalización.
Nro Regitros	Concepto	Un valor que indica cuantos registros tiene una tabla.
Nuevo Atributo	Concepto	Una columna que se va a agregar a una tabla.
Operación	Concepto	Una fusión matemática que se puede realizar entre dos atributos de una tabla.
Preprocesamiento de Datos	Caso de Uso	Descripción del proceso donde el se procesa y alista una tabla para su posterior modelaje.
Registro	Concepto	Una tupla que integra una tabla.
Regla	Concepto	Un conjunto de condiciones que tienen un resultado.
Seleccionar Registros	Caso de Uso	Descripción del proceso donde se selecciona un número determinado de registros.
Seleccionar Tabla	Caso de Uso	Descripción del proceso donde el analista una tabla para ver su información (Columnas, tipo, número de registros)
SeleccionarAtriEliminar.Natributo: carácter	Atributo	Nombre del atributo que se va a eliminar de la tabla.

Tabla	Concepto	Un conjunto de datos
Tabla Entrenamiento	Concepto	Un conjunto de datos utilizado para la construcción de un modelo de clasificación.
Tabla Prueba	Concepto	Un conjunto de datos utilizado para probar el modelo de clasificación construido.
Validar Modelo	Caso de Uso	Descripción del proceso en el cual se valida las reglas generadas de un modelo se las traduce e interpreta.
Ver Árbol de Decisión	Caso de Uso	Descripción del proceso donde el analista puede ver el árbol grafico de un modelo.

5. IMPLEMENTACIÓN

En este capítulo se describe detalladamente la implementación y funcionamiento de los algoritmos de clasificación C4.5, Mate-Tree y Sliq así como los procesos para la fase de selección, preprocesamiento, transformación y validación, estos fueron codificados en su totalidad como funciones definidas por el usuario en el SGBD PostgreSQL utilizando el lenguaje PLPGSQL.

5.1. ARQUITECTURA DE POSTGRES

Los conceptos presentados a continuación fueron tomados de [23] y [40].

PostgreSQL es un Sistema Gestor de Bases de Datos Objeto-Relacional. Esto quiere decir que PostgreSQL es un sistema de manejo de Bases de Datos relacional (SMBD) que soporta un modelo de datos consistente en una colección de relaciones con nombre, que contienen atributos de un tipo específico. A su vez, tiene algunas características que son propias del mundo de las bases de datos orientadas a objetos como son el soporte a conceptos tales como:

- Clases
- Herencia
- Tipos
- Funciones

PostgreSQL es una mejora de Postgres. El código original de Postgres fue desarrollado en la Universidad de California, Berkeley; en 1986, por un grupo integrado por estudiantes de pregrado y postgrado de la universidad, y desarrolladores dirigidos por el profesor Michael Stonebraker. Originalmente, Postgres implementó su propio lenguaje de consultas POSTQUEL. La primera versión de Postgres se lanzó en Junio de 1989. En 1994 se añadió un intérprete de lenguaje SQL a Postgres.

En 1995 se liberó el código fuente de Postgres, y se le dio el nombre Postgres95. Postgres95 fue adaptado a ANSI C y se adoptó el lenguaje de consultas SQL, como lenguaje, además se utilizó GNU make para la compilación. En 1996 se cambió el nombre a PostgreSQL y se añadieron características adicionales que cumplen el estándar SQL92.

5.1.1. Características de Postgres. Postgres (POSTinGRES) es el sucesor del sistema de base de datos relacionales INGRES, que provee un mejor soporte a aplicaciones de inteligencia artificial y desarrollo de ingeniería que su antecesor. Esta meta es cumplida por Postgres, extendiendo el lenguaje de consulta relacional, incluyendo:

- Soporte a tipos de datos abstractos (Abstract Data Type ADT) y métodos de acceso definidos por el usuario.

- Nuevo lenguaje, QUEL, para soportar operaciones de clausura transitiva requeridas en aplicaciones de Inteligencia Artificial.

- Soporte del lenguaje QUEL como un tipo de dato, para incrementar el poder de modelamiento del sistema relacional.

- Soporte de Reglas y Triggers en sistemas relacionales para mejorar la inferencia y encadenamiento hacia delante, requerido por aplicaciones de sistemas expertos.

El lenguaje de consulta que soporta esta característica es llamado POSTQUEL.

Postgres soporta tipos de datos abstractos, permitiéndole al usuario definir sus propios tipos de datos, para simplificar la representación de información compleja. Además, permite definir operadores para ser utilizados, junto con los tipos de datos definidos por el usuario. Permite a los usuarios extender los métodos de acceso existentes, para ser utilizados con los nuevos operadores, por ejemplo, el usuario puede especificar métodos de acceso para el recorrido eficiente de una relación cuando un operador no estándar aparece en una cláusula de restricciones.

El modelo relacional no es el adecuado para representar relaciones jerárquicas.

Postgres propone Consulta Embebidas dentro de los campos de datos y utilizar estas consultas para expresar la relación jerárquica entre la tupla correspondiente y la información en la base de datos.

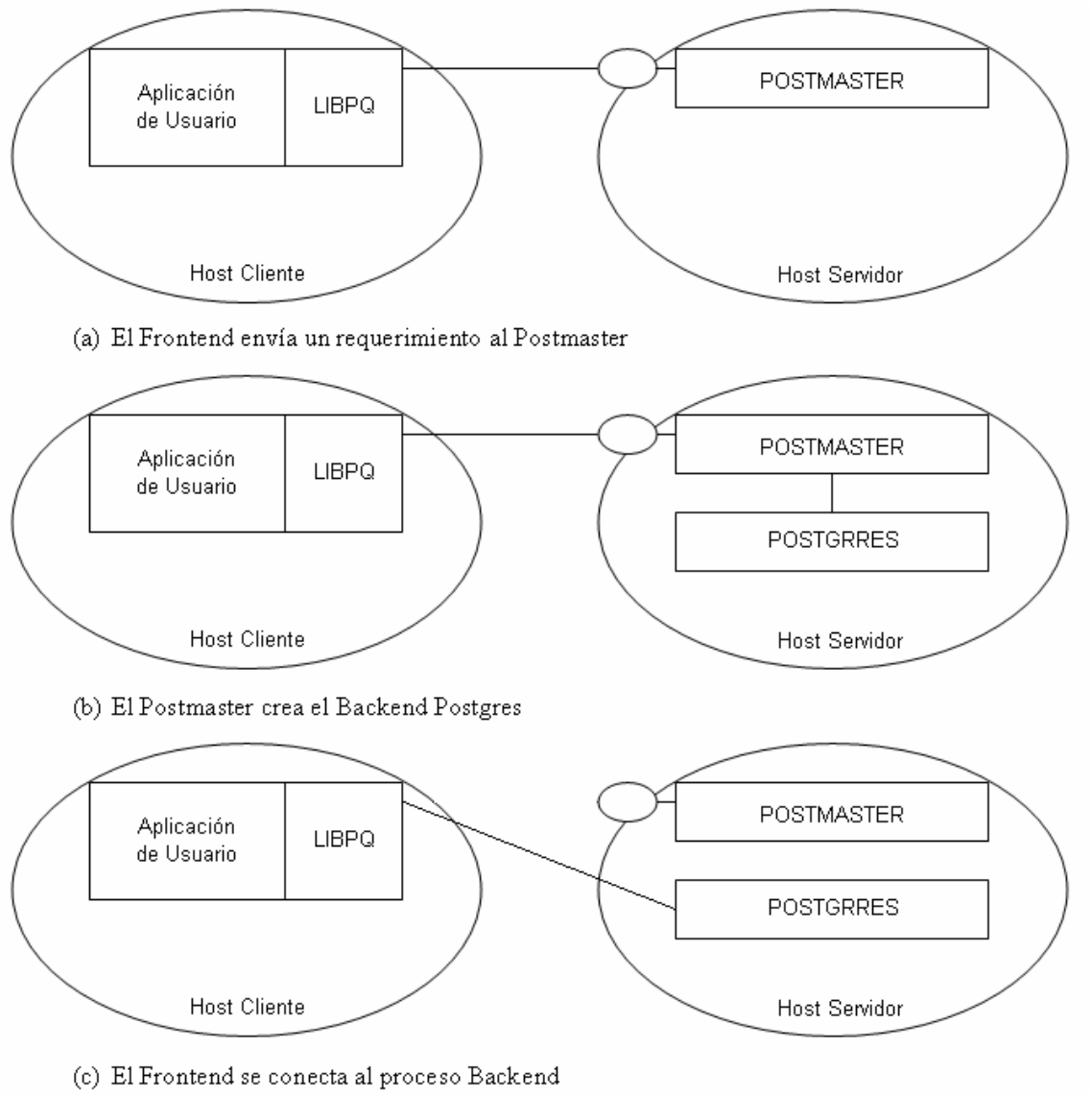
5.1.2. Conceptos de arquitectura de PostgreSQL. PostgreSQL utiliza un modelo de arquitectura de proceso por usuario cliente/servidor; por lo cual el Sistema Manejador de Base de D de PostgreSQL se ejecuta en un equipo diferente al de las aplicaciones que acceden a las Bases de Datos para proveer protección a los datos. En el establecimiento de la conexión se requiere de la colaboración de los siguientes procesos:

- Postmaster: Proceso del demonio supervisor.

- Frontend: La aplicación del usuario (programa pgsql)

- Backend: Servidores de bases de datos (mismo Postgres)

Figura 31. Proceso de conexión



El proceso postmaster es el encargado de atender un puerto TCP/IP especifico, a través del cual las aplicaciones frontend, que quieren acceder a una base de datos específica, dentro de una instalación, hacen los llamados respectivos. Cada vez que el proceso postmaster detecta una solicitud de conexión, inicia un proceso de servidor backend y conecta el proceso frontend al nuevo servidor backend. De esta forma, el proceso postmaster queda libre para esperar una nueva solicitud mientras el proceso frontend se comunica con Postgres, a través del servidor backend. El servidor backend, comunica entre si, los procesos Postgres, usando semáforos y memoria compartida para asegurar la integridad de los datos, a través de los accesos concurrentes a la base de datos.

El proceso cliente puede ser un psql frontend (para consultas iterativas SQL) o alguna aplicación implementada utilizando la librería libpq.

Una vez establecida la conexión, el proceso cliente puede enviar una consulta al backend. La consulta es transmitida en texto plano, no se realiza ningún proceso parser en el lado del cliente. El servidor realiza el proceso de parser a la consulta, crea el plan de ejecución, ejecuta el plan y retorna las tuplas al cliente transmitiéndola a través de la conexión establecida.

5.1.3. Procesamiento de una consulta en PostgreSQL. Las etapas que se deben realizar, a gran escala, para procesar una consulta en Postgres son las siguientes (ver figura 32):

- Establecer una conexión desde un programa aplicación al servidor Postgres. A través de esta conexión, el programa aplicación transmite las consultas y recibe los resultados del Servidor Postgres. Las consultas son transmitidas como una cadena de texto plano.

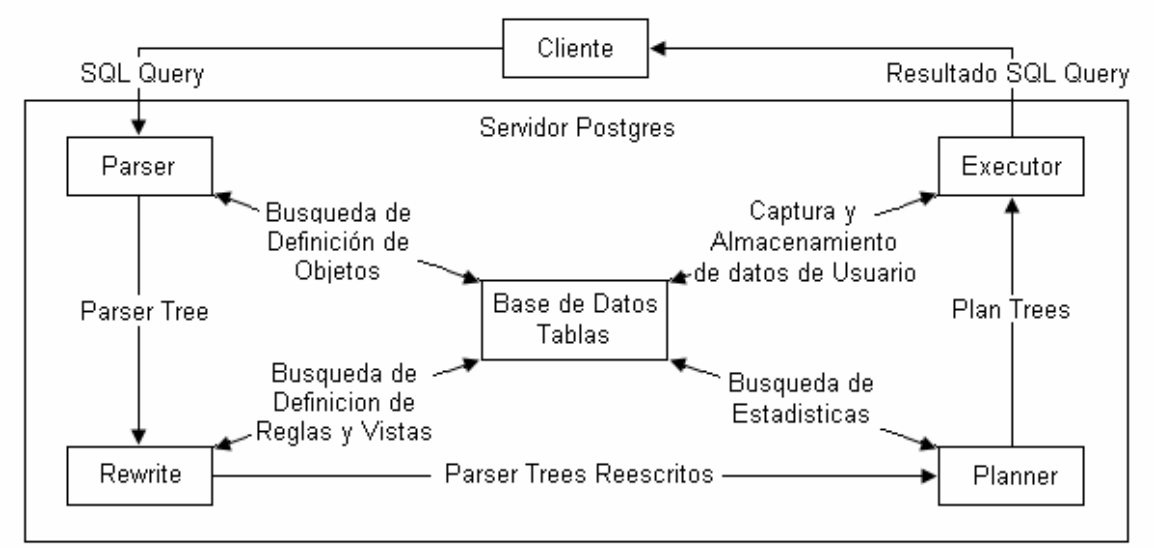
- Realizar un análisis léxico, sintáctico y semántico a la consulta transmitida. Entre las comprobaciones que se realizan, se verifica la existencia de las relaciones y atributos indicados en la consulta. Si la consulta no presenta errores se crea un árbol de consulta llama query tree. Esta etapa es conocida como etapa Parser.

- En la siguiente etapa se toma la estructura query tree, y se modifican las Reglas o Vistas (VIEW) que se puedan aplicar dentro de la estructura query tree. Esta transformación la realiza el sistema rewrite, sistema de reescritura. Siempre que se presenta una Vista en la consulta el sistema rewrite, transforma la consulta del usuario en una consulta que acceda a las tablas base, dadas en la definición de la vista inicial.

- En la etapa de optimización, se toma la estructura query tree, resultado de la etapa rewrite, y se crean todas las posibles rutas de acceso que conduzcan al mismo resultado. Se selecciona aquella ruta de acceso que tenga el menor costo de ejecución y se crea el plan de la consulta, dentro de la estructura query plan.

- El encargado de ejecutar la estructura query plan y de recuperar las tuplas indicadas en la consulta es el executor. En esta etapa, se toma la estructura query plan y se ejecuta recursivamente. Se buscan relaciones, se realizan ordenamientos y Joins, se evalúan cualificaciones, se hace uso del sistema de almacenamiento y se recuperan las tuplas en la forma representada en el plan.

Figura 32. Procesamiento de una consulta



- **Etapa Parser.** La etapa parser consiste de dos procesos: el proceso parser y el proceso de transformación.

Parser. El parser verifica la validez sintáctica de la cadena de consulta. Si la sintaxis es correcta se construye y se retorna una estructura parser tree, de lo contrario, se retorna un mensaje de error. Para implementar esta etapa se usa las herramientas Lex y Yacc (Revisar Anexo C).

El análisis léxico se define en el archivo scan.l y su objetivo es reconocer los identificadores, palabras reservadas SQL, etc. Para cada identificador o palabra clave identificada, un token se genera y se envía al parser (…/src/backend/parser/scan.l).

El análisis semántico se define en el archivo gram.y y consiste en un conjunto de reglas gramaticales y acciones que son ejecutadas cuando una regla se dispara. El código de las acciones está definido en sentencias de código C y se utiliza para construir una estructura parse tree (…/src/backend/parser/gram.y).

El archivo scan.l se transforma a un archivo en código C llamado scan.c, usando el programa lex y el archivo gram.y se transforma a un archivo en código gram.c utilizando la herramienta yacc. Después de realizadas estas transformaciones, un compilador normal de C se puede usar para crear el parser. Note que las transformaciones y compilaciones mencionadas se realizan a través de archivos Makefile distribuidos en el código de PostgreSQL.

Para entender mejor la estructura de datos usada en PostgreSQL para el procesamiento de una consulta, se ilustrará con un ejemplo los cambios hechos a la estructura de datos en cada etapa.

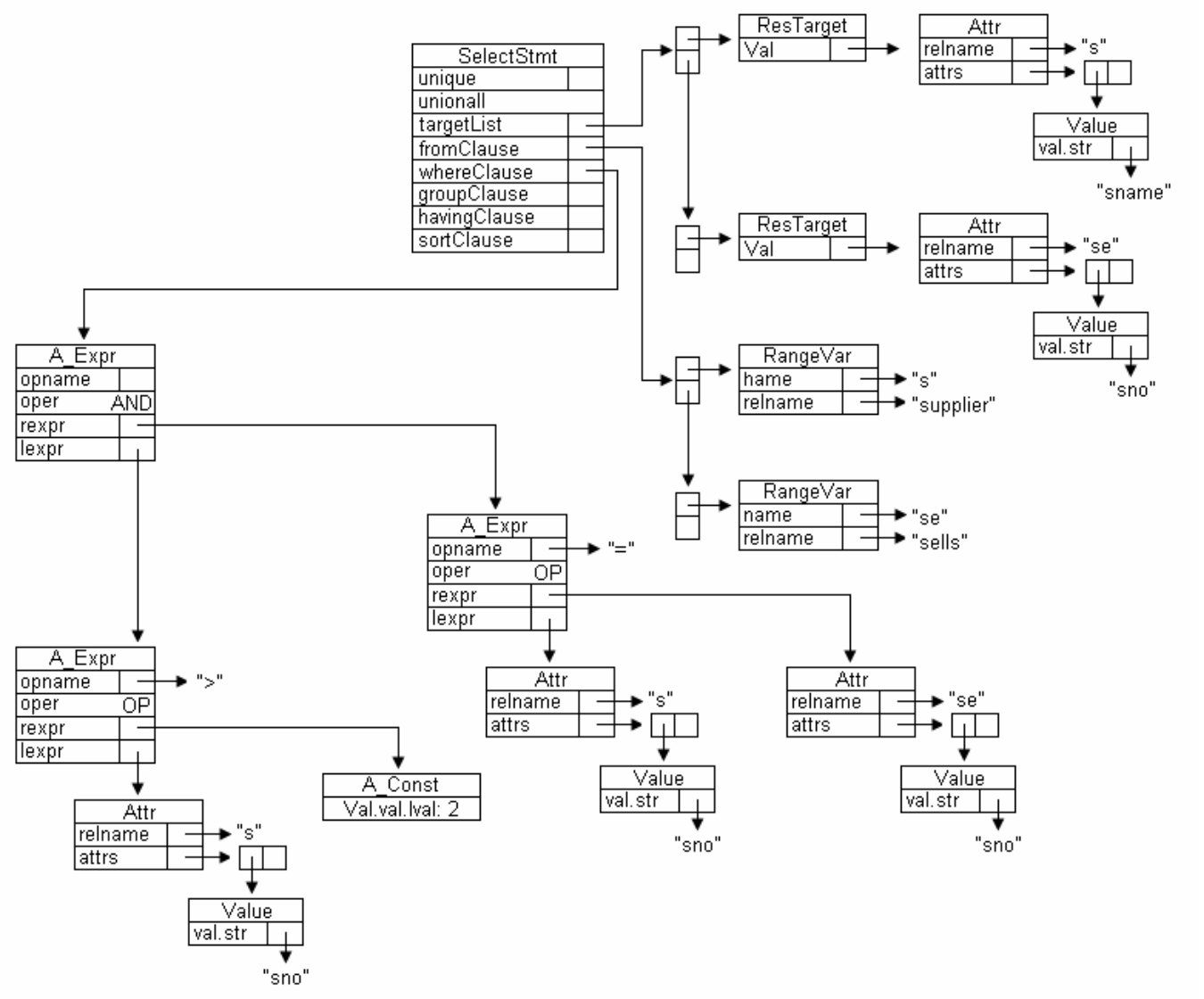
Ejemplo 3.1 Este ejemplo contiene la siguiente consulta simple que será usada en varias descripciones y figuras a través del ejemplo de procesamiento de una consulta en PostgreSQL. La consulta asume que las tablas ya están definidas. (ver figura 33).

```
select s.sname, se.pno
from supplier s, sells se
where s.sno > 2 and
s.sno = se.son;
```

La Figura 33 muestra el parser tree construido por las reglas gramaticales dadas en gram.y para la consulta dada en el ejemplo 3.1.

El nodo raíz del árbol es un nodo SelectStmt. Para cada entrada que aparezca en la cláusula FROM de la consulta SQL, un nodo RangeVar se crea conteniendo el nombre del alias y el nombre de la relación. Todos los nodos RangeVar se coleccionan en una lista que es añadida al campo fromClause del nodo SelectStmt.

Figura 33. Parser tree de la consulta del ejemplo 3.1



Para cada entrada que aparece en la lista SELECT de la consulta SQL, se crea un nodo ResTarget, que contiene un apuntador a un nodo Attr. El nodo Attr contiene el nombre de la relación de entrada y un puntero a un nodo Value que contiene el nombre del atributo. Todos los nodos ResTarget son coleccionados en una lista que está conectada al campo targetList del nodo SelectStmt.

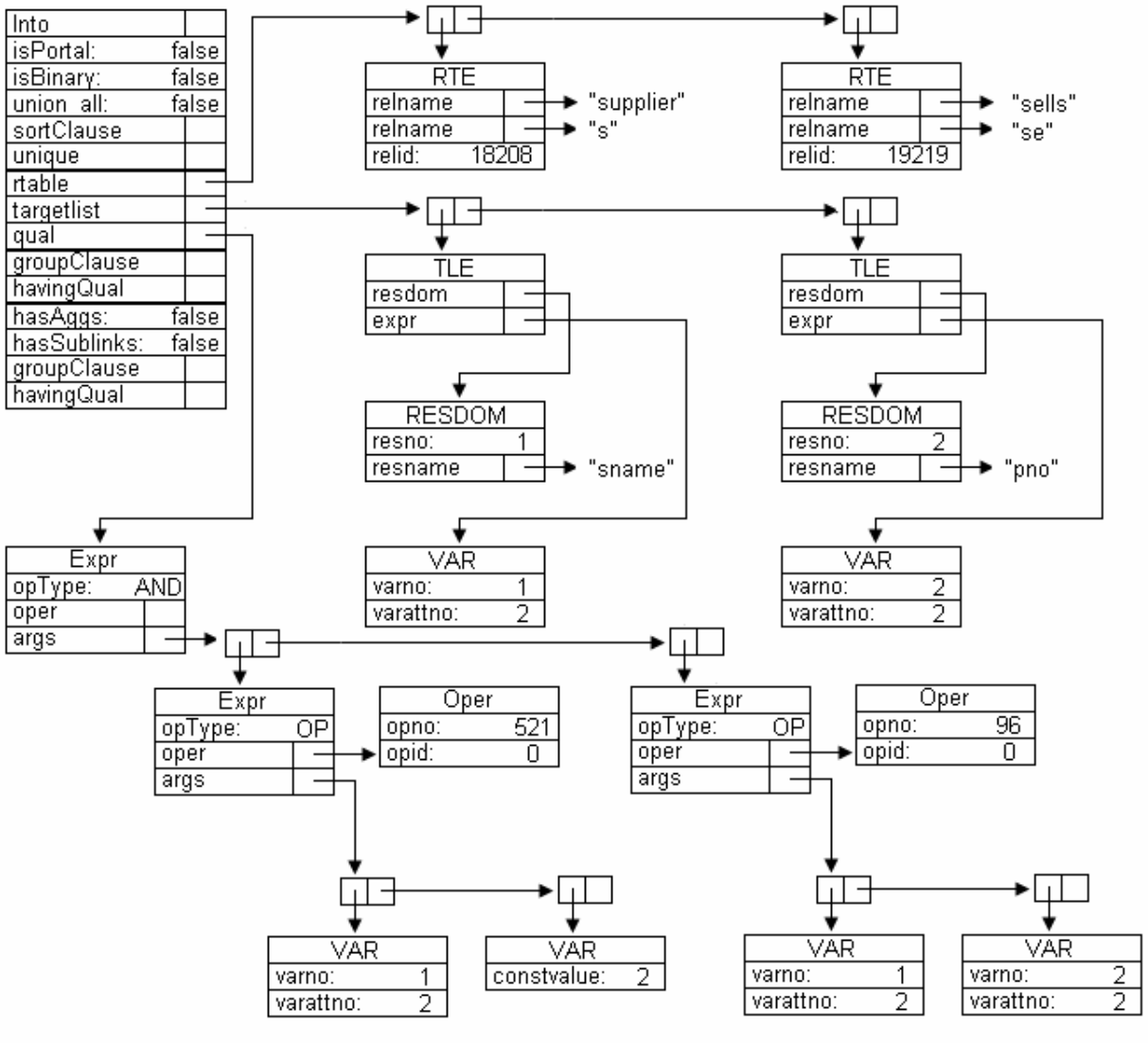
El operador tree se construye para la cláusula WHERE de la consulta SQL, la cual es añadida al campo qual del nodo SelectStms. En la Figura 33 la raíz del operador tree es un nodo A_Expr que representa una operación AND. Este nodo tiene dos sucesores llamados lexpr y rexpr apuntando a dos subárboles. El subárbol apuntado por lexpr representa la cualificación s.son > 2 y el subárbol que apunta al rexpr representa la cualificación s.son = se.son. Para cada atributo, se crea un nodo Attr conteniendo el nombre de la relación y un puntero al nodo Value que contiene el nombre del atributo. Para las constantes que aparecen en la consulta, se crea un nodo Const es creado conteniendo el valor de la constante.

Proceso de transformación. El proceso de transformación toma el nodo tree retornado por la etapa parser como estructura de entrada y lo recorre recursivamente. Si un nodo SelectStmt se encuentra, éste se transforma en un nodo Query, que será el nodo raíz de la nueva estructura de datos. La Figura 34 muestra la estructura de datos transformada.

Se realiza una verificación de la existencia en el sistema, sobre de las relaciones, mencionadas en la cláusula From. Para cada relación que está presente en el catálogo, un nodo RTE se crea, conteniendo el nombre de la relación, el nombre alias y el id de la relación. De aquí en adelante los id’s de la relación se usan para referenciar las relaciones dadas en la consulta. Todos los nodos RTE se coleccionan en la lista range table entry list (Lista de entrada de tablas de rango) que esta conectada el campo rtable del nodo Query. Si un nombre de relación no es conocido por el sistema en la consulta, un error se retorna y el procesamiento de la consulta será abortado.

Una vez se verifican los nombres de las relaciones, se procede a verificar si los nombres de los atributos usados en la consulta, están contenidos en las relaciones dadas. Para cada atributo, se crea un nodo TLE, conteniendo un puntero a un nodo Resdom (el cual contiene el nombre de la columna) y un puntero a un nodo Var. En el nodo Var hay dos números importantes. El campo varno dá la posición de la relación que contiene el atributo actual en el range table entry list. El campo varattno dá la posición del atributo dentro de la relación. Si el nombre de un atributo no se encuentra, se retorna un error y el procesamiento de la consulta se aborta.

Figura 34. Parser tree transformado, de la consulta del ejemplo 3.1



Cada nodo A_Expr se transforma en un nodo Expr. El nodo Attr que representa los atributos, se reemplaza por nodos Var, como se realizó en el campo targetlist. Se realiza una verificación si los atributos que aparecen son válidos y conocidos por el sistema. Si algún atributo no es conocido por el sistema, se retorna error y el procesamiento de la consulta se aborta.

Todo el proceso de transformación, realiza una transformación de la estructura de datos retornada por la etapa parse, para obtener una estructura de mayor manipulación para las etapas restantes del proceso de optimización. Las cadenas de caracteres representando relaciones y atributos en el árbol original se reemplazan por los id's de la relación y nodos Var cuyos campos se refieren a las entradas en el range table entry list. Además en la etapa de transformación, también se realizan verificaciones sobre la valides de los nombres de las relaciones y atributos en el contexto actual.

- **Sistemas Rewrite.** PostgreSQL soporta un poderoso sistema de reglas para la especificación de vistas y vistas actualizables. El sistema de reglas de PostgreSQL consiste en el sistema Rewrite. El sistema query rewrite es un módulo entre la etapa parser y la etapa del optimizador de consultas. En esta etapa se toma el árbol devuelto por la etapa parser y se realza la búsqueda de reglas presentes dentro de la consulta. Si se encuentra alguna regla dentro de la consulta, la estructura de árbol se transforma en una expresión equivalente que incluye las relaciones base. La reescritura de la consulta se realiza en el archivo …/src/backend/rewrite/rewriteHandler.c.

- **Planner/Optimizer.** La Tarea de la etapa de optimización es crear un plan de ejecución óptimo. Primero, se combinan todas las posibles formas de recorrer y unir las relaciones que aparecen en la consulta. Todas los Path's creados llevan al mismo resultado y la tarea del optimizador es estimar el costo de ejecución de cada Path y encontrar cuál de éstos es el mas barato. La generación del plan de ejecución se realiza en el archivo …/src/backend/optimizar/plan/planner.c.

El optimizador decide cuales planes se deben generar, basado en los tipos de índices definidos en las relaciones que aparecen en la consulta. Un plan usando escaneo secuencial siempre se crea. Si un índice es definido para una relación y una consulta contiene una restricción de la forma relación.atributo OPR constante y relación.atributo coincide con la clave del índice y OPR es un operador relacional distinto del operador ≠, se crea otro plan utilizando el índice para recorrer la relación. Si existen otros índices y en las restricciones de la consulta coinciden con las claves de dichos índices, estos serán considerados por el optimizador.

Después de que todos los planes factibles para recorrer una relación simple han sido encontrados, se crean los planes para unir las relaciones. El optimizador considera solamente Joins entre dos relaciones, para las cuales existe una correspondientes cláusula Join (por ejemplo una restricción como where rel1.attr1 = rel2.attr2) en la cualificación de

la cláusula WHERE. Todos los posibles planes son considerados para cada pareja de JOIN tomadas en cuenta por el optimizador.

Las posibles estrategias JOIN consideradas por el optimizador son:

- Join nested: La relación de la derecha se recorre una vez por cada tupla encontrada en la relación de la izquierda. Esta estrategia es fácil de implementar pero puede consumir mucho tiempo.

- Join merge-sort: Cada relación se ordena por los atributos del Join antes de iniciar el Join. Después las dos relaciones se mezclan, tomando en cuenta que ambas relaciones están ordenadas sobre los atributos del Join. Esta clase de Join es más atractiva porque cada relación se recorre una sola vez.

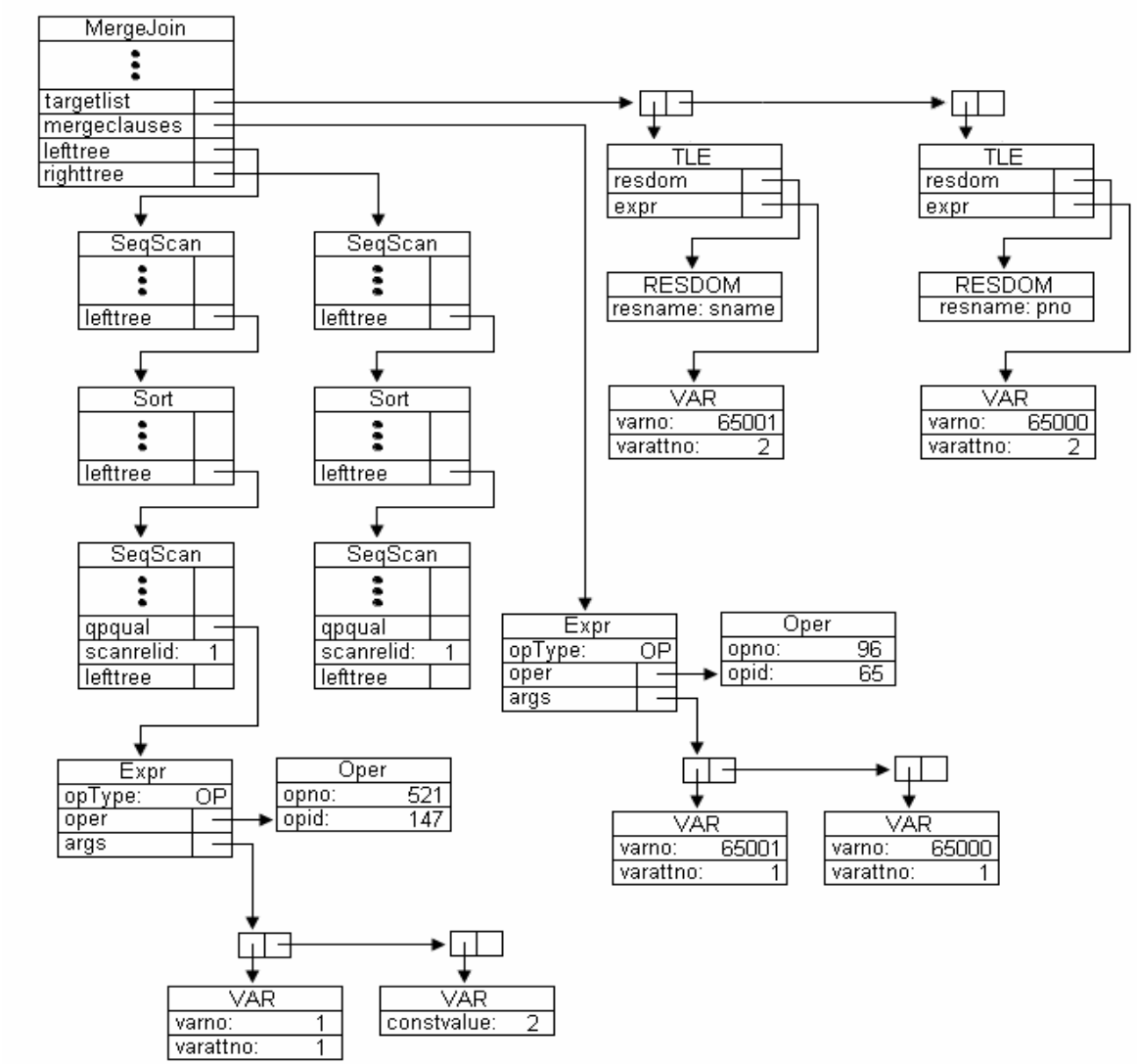
- Join Hash: La relación de la derecha es primero hashed en los atributos del Join. Después la relación de la izquierda se recorre y los valores apropiados de cada tupla encontrada son usados con clave hash para ubicar las tuplas en la relación de la derecha.

La Figura 35. Muestra el plan producido por la consulta en el ejemplo 3.1.

A continuación se da una breve descripción de los nodos que aparecen en el plan. El nodo raíz del Plan, es un nodo MergeJoin con 2 sucesores, uno atado al campo lefttree y el segundo atado al campo righttree. Cada uno de los subnodos representa una relación Join. Un Join merge-sort requiere que cada relación se encuentre ordenada. Por esto, se encuentra un nodo Sort en cada SubPlan. La cualificación adicional dada en la consulta s.sno > 2 se desplaza hacia abajo, lo mas lejos que sea posible y se enlaza al campo qpqual del nodo SeqScan del correspondiente subplan.

La lista enlazada al campo mergeclauses del nodo MergeJoin contiene información acerca los atributos del Join. Los valores 65000 y 65001 para los campos varno en el nodo Var, que aparecen en la lista mergeclauses (está contenida en el campo targelist), significa que no se debe considerar las tuplas del nodo actual sino las tuplas del siguiente nodo en profundidad, por ejemplo los nodos raíces del subplan.

Figura 35. Plan para la consulta del ejemplo 3.1



Otra tarea realizada por el optimizador es establecer el campo operator id en los nodos Expr y Oper. Debido a que PostgreSQL soporta una gran variedad de tipos de datos diferentes y también los definidos por el usuario, es necesario almacenarlas en un sistema de tablas, para poder mantener la enorme cantidad de funciones y operadores. Cada función y operador tiene un único operator id. De acuerdo al tipo de los atributos usados dentro de las cualificaciones se usa el operador apropiado.

Estimación de costo y tamaño. Para podar el espacio de planes, el optimizador debe estimar el costo de ejecución de un plan generado. El optimizador, para estimar el costo, toma en cuenta el tiempo requerido en operaciones CPU y de Entrada/Salida. Cada factor de costo, tiene la siguiente forma:

$$Costo = P + W*T.$$

donde P es el número de páginas examinadas en tiempo de ejecución por el plan y T es el número de tuplas examinadas. P refleja el costo de operaciones de Entrada/Salida, T refleja el costo de operaciones de CPU y W es un factor de peso que indica la importancia relativa del costo de procesamiento en términos de operaciones de CPU y de Entrada/Salida.

Para un recorrido secuencial de una relación, cada página y tupla se debe examinar, P es el número de páginas en la relación. Para un índice secundario, el costo depende del número de páginas y tuplas en el índice de la relación, porque las páginas y tuplas del índice se deben leer primero para determinar dónde examinar la relación principal.

Para cada índice, el número de páginas y tuplas está determinado por la fracción de tuplas en la relación que se espera satisfagan la cláusula de restricción. Esta estimación recibe el nombre de selectividad. La selectividad está en función de una variedad de parámetros, incluyendo el operador de la cláusula de restricción, la constante de restricción, el número de registros de un índice y los valores máximo y mínimo almacenados en el atributo.

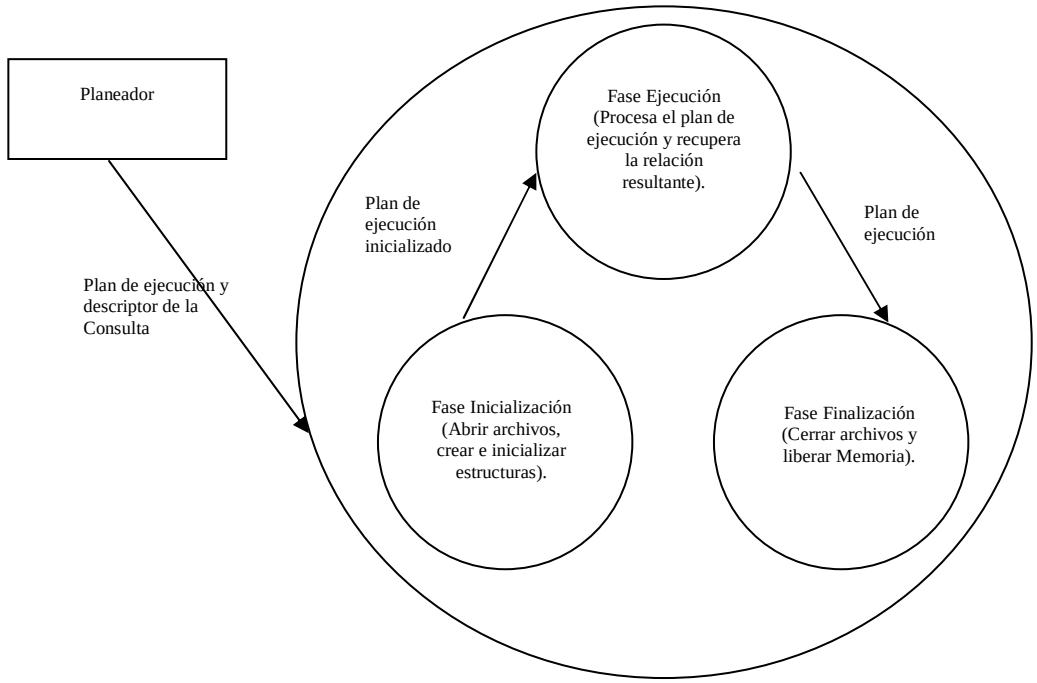
Las fórmulas para la estimación de costo de todas las estrategias Join son funciones del tamaño, en número de páginas y tuplas, de las relaciones del Outer e Inner Join. Para estimar el tamaño en la relación Outer o Inner del Join, el optimizador multiplica el tamaño original de la relación por la selectividad de cada cláusula de restricción aplicable a la relación. Si la cláusula utiliza un índice, la selectividad se calcula, como se indicó anteriormente. Si esto no es posible, el optimizador asocia constantes a estos factores. Si la relación del Outer Join es una relación compuesta, el factor de selectividad está dado por la operación Join. Este factor de selectividad indica la fracción del producto entre las relaciones del Outer e Inner Join, que se espera satisfagan la cláusula Join.

Postgres almacena en el catálogo del sistema, información estadística requerida por las fórmulas de costo y actualiza información el número de páginas y tuplas en una relación, utilizando demonios que se ejecutan en modo background. Estos demonios son implementados utilizando triggers. El valor máximo y mínimo de un atributo, se almacena en la forma de consulta embebida dentro del dato. Esta consulta capturan la información apropiada de la base de datos, y el sistema almacena los datos en cache, así el optimizador no necesita ejecutar repetidamente la consulta.

- **Executor.** Dada una consulta, y luego del análisis sintáctico y semántico de ésta, el planeador de Postgres, apoyado en el optimizador, define un plan de ejecución. Este plan representa lógicamente los pasos necesarios para satisfacer dicha consulta, y físicamente es un árbol cuyos nodos representan operaciones básicas.

Una vez creado el plan de ejecución por parte del planeador, este se ejecuta, es decir, se ejecuta la consulta. Para esto, el ejecutor de consultas de Postgres se divide en tres fases: inicialización, ejecución y finalización (Figura 36).

Figura 36. Fases del ejecutor de Postgres



- **Fase de inicialización.** En esta fase se asignan los recursos necesarios para que el plan de ejecución se lleve a cabo. Básicamente se crean e inicializan estructuras asociadas a los nodos que componen el plan de ejecución, y se abren los archivos correspondientes a las relaciones (tablas) que intervienen en la consulta.

La rutina encargada de realizar la fase de inicialización es ExecutorStart, declarada en el archivo ExecMain.c (…/src/backend/executor/ExecMain.c). Esta tiene como parámetros la descripción de la consulta y el estado de ejecución. La descripción de la consulta contiene el árbol semántico, el plan de ejecución y la acción a realizar (select, update, insert, delete). El estado de ejecución contiene información acerca de la dirección de búsqueda, conjunto de relaciones que intervienen en la consulta y bloques de memoria disponibles, entre otros. Esta función retorna la descripción de los atributos del resultado de la consulta, es decir, el descriptor de la relación resultante.

Al interior de ExecutorStar se llama a la función InitPIan, declarada en el archivo ExecMain.c, la cual en realidad, inicializa el plan de consulta. Esta función recibe los mismos argumentos que ExecutorStart, y también retorna el descriptor de la relación resultante.

La función InitPIan invoca a ExecInitNode, declarada en el archivo ExecProcNode.c (…/src/backend/executor/ExecProcNode.c), con el nodo raíz del plan de ejecución y el estado de la consulta como argumentos. La función ExecInitNode es un gran switch, que dependiendo del tipo de nodo llama a la función encargada de inicializarlo. Dado el caso que un nodo tenga subplanes de entrada y salida, la función también se usa para inicializarlos. De esta forma se inicializa recursivamente todo el plan de ejecución. La rutina ExecInitNode retorna verdadero o falso según se inicialice ó no correctamente el nodo.

Postgres clasifica los nodos (Operadores):

- Control: Result, Append
- Búsqueda: SeqScan, IndexScan
- Unión: NestLoop, MergeJoin
- Materialización: Material, Sort, Unique, Group, Agg, Hash, HashJoin

- Fase de Ejecución. Esta es la fase principal, debido a que realiza la ejecución real de la consulta. La función ExecRun declarada en el archivo ExecMain.c (…/src/backend/executor/ExecMain.c), es la encargada de realizar esta fase.

La función ExecutePlan, declarada en el archivo ExecMain.c, se utiliza por la función ExecutorRun para realizar su tarea. La función ExecutePlan procesa en la dirección especificada el plan de consulta, y retorna tanto el número de tuplas como las tuplas mismas.

La función ExecutePlan invoca a ExecProcNode, ubicada en el archivo ExecProcNode.c, con el nodo raíz del plan de ejecución como argumento. La función ExecProcNode, al igual

que ExecInitNode, es un switch que identifica el tipo de nodo e invoca a la rutina correspondiente. De esta forma, se obtiene la tupla resultante del proceso realizado al interior del nodo. Esta función también se utiliza para ejecutar los subplanes de entrada y salida de un nodo, y es así como se ejecuta recursivamente todo el plan de ejecución.

- Fase de Finalización. Esta fase corresponde al final de la ejecución de cualquier consulta, y se encarga de liberar todos los recursos que se reservaron en la fase de inicialización. La rutina asociada a esta tarea es ExecutorEnd, declarada en el archivo ExecMain.c. Esta rutina tiene los mismos argumentos que ExecutorStart y no retorna nada.

Al interior de ExexutorEnd se llama a la función EndPlan, declarada en el archivo ExecMain.c. La función EndPlan finaliza el plan de consulta (cerrar archivos y liberar memoria de estructuras), recibiendo como parámetros el plan y el estado de ejecución. Esta función no retorna nada.

La función EndPlan llama a la función ExecEndNode, declarada en el archivo ExecProcNode.c. La función ExecEndNode recibe los mismos argumentos que la función EndPlan, y es también un switch que dependiendo del tipo de nodo invoca a la función encargada de finalizarlo. De esta manera, esta función finaliza cualquier tipo de nodo. La función también se utiliza para finalizar los subplanes de entrada y salida de un nodo, y así finalizar recursivamente todo el plan de ejecución con solo finalizar el nodo raíz. La función ExecEndNode solo es llamada si el plan ha sido ejecutado sin errores, ya que después de esta operación, el plan de consulta no puede ser ejecutado nuevamente.

5.2. PROGRAMACIÓN EN EL SERVIDOR.

PostgreSQL le proporciona al usuario la capacidad de diseñar e implementar funciones de propósito especializado o general. Dichas funciones se escriben en un lenguaje de programación anfitrión C y en un lenguaje basado en SQL como PL/PGSQL.

5.2.1. Programación de funciones en PL/PGSQL para Postgresql. Es común que los desarrolladores de aplicaciones subutilicen las prestaciones de las bases de datos relacionales modernas, en ocasiones simplemente por desconocer las ventajas que le ofrecen o por desconocer su manejo. Dentro de PostgreSQL, la base de datos de código abierto más poderosa, se pueden desarrollar funciones en varios lenguajes. El lenguaje PL/pgSQL es uno de los más utilizados dentro de PostgreSQL, debido a que guarda cierta similitud con PL/SQL de Oracle y a su facilidad de uso.

Ventajas de usar PL/pgSQL. SQL es el lenguaje estándar para realizar consultas a un servidor de base de datos. Cada sentencia SQL se ejecuta de manera individual por el servidor, lo cual implica que las aplicaciones cliente deben enviar cada consulta al servidor, esperar a que la procese, recibir los resultados, procesar los datos y después enviar la siguiente sentencia.

Al usar PL/pgSQL es posible realizar cálculos, manejo de cadenas y consultas dentro del servidor de la base de datos, combinando el poder de un lenguaje procedimental y la facilidad de uso de SQL, minimizando el tiempo de conexión entre el cliente y el servidor.

Estructura de PL/pgSQL. El lenguaje PL/pgSQL es estructura en bloques. Todas las palabras clave y los identificadores pueden escribirse mezclando letras mayúsculas y minúsculas. Un bloque se define de la siguiente manera:

```
[<<label>>]
[DECLARE
        declaraciones]
```

COMENTARIOS, CONSTANTES Y VARIABLES 3

```
BEGIN
        Sentencias
END;
```

Pueden existir varios bloques o sub-bloques en la sección de sentencias de un bloque. Los sub-bloques pueden ser usados para ocultar las variables a los bloques más externos. Normalmente una de las sentencias es el valor de retorno, usando la palabra clave RETURN.

Las variables declaradas en la sección que antecede a un bloque se inicializan a su valor por omisión cada vez que se entra al bloque, no solamente al ser llamada la función. Por ejemplo:

```
CREATE FUNCTION estafunc() RETURNS INTEGER AS '
DECLARE
        cantidad INTEGER := 30;
BEGIN
        RAISE NOTICE "Cantidad contiene aquí %",cantidad;
        -- Cantidad contiene aquí 30
        cantidad := 50;
        --
        -- Creamos un sub-bloque
        --
        DECLARE
        cantidad INTEGER := 80;
        BEGIN
        RAISE NOTICE "Cantidad contiene aquí %",cantidad;
        -- Cantidad contiene aquí 80
        END;
        RAISE NOTICE "Cantidad contiene aquí %",cantidad;
```

```
        -- Cantidad contiene aquí 50
        RETURN cantidad;
END;
' LANGUAGE 'plpgsql';
```

No se debe confundir el uso de las sentencias de agrupamiento BEGIN/END de PL/pgSQL con los comandos de la base de datos que sirven para el control de las transacciones. Las funciones y procedimientos disparadores no pueden iniciar o realizar transacciones y Postgres no soporta transacciones anidadas.

VARIABLES Y CONSTANTES

Todas las variables, filas y registros usados en un bloque o en sus sub-bloques deben declararse en la sección de declaraciones del bloque.

La excepción es la variable de un ciclo FOR que itera sobre un rango de valores enteros. Las variables en PL/pgSQL pueden ser de cualquier tipo de datos de SQL, como INTEGER, VARCHAR y CHAR. El valor por omisión de todas las variables es el valor NULL de SQL.

A continuación se muestran algunos ejemplos de declaración de variables:

```
user_id INTEGER;
quantity NUMBER(5);
url VARCHAR;
```

EXPRESIONES

Todas las expresiones usadas en las sentencias de PL/pgSQL son procesadas usando el ejecutor del backend. Las expresiones que parecen contener constantes pueden requerir de una evaluación en tiempo de ejecución (por ejemplo, now para el tipo de dato timestamp) así que es imposible para el analizador sintáctico (parser) de PL/pgSQL identificar los valores de las constantes reales diferentes de NULL. Todas las expresiones son evaluadas internamente ejecutando una sentencia SELECT expresión usando el gestor de SPI. En la expresión, las ocurrencias de los identificadores de las variables se sustituyen por parámetros y los valores reales de las variables se pasan al ejecutor en el arreglo de parámetros. Todas las expresiones usadas en una función de PL/pgSQL son preparadas y almacenadas solamente una vez.

La única excepción a esta regla es una sentencia EXECUTE si se requiere analizar una consulta cada vez que es encontrada. La revisión del tipo realizada por el analizador sintáctico principal de Postgres tiene algunos efectos secundarios a la interpretación de valores constantes.

SENTENCIAS

Cualquier cosa no comprendida por el analizador PL/pgSQL tal como se especifica adelante seria enviado al gestor de la base de datos, para su ejecución. La consulta resultante no devolvería ningún dato.

ASIGNACIÓN

Una asignación de un valor a una variable o campo de fila o de registro se escribe:

```
identifier := expression;
```

SENTENCIAS

Si el tipo de dato resultante de la expresión no coincide con el tipo de dato de las variables, o la variable tienen un tamaño o precisión conocido (como char(29)), el resultado seria amoldado implícitamente por el interprete de bytecode de PL/pgSQL, usando los tipos de las variables para las funciones de entrada y los tipos resultantes en las funciones de salida. Nótese que esto puede potencialmente producir errores de ejecución generados por los tipos de las funciones de entrada.

Una asignación de una selección completa en un registro o fila puede hacerse del siguiente modo:

```
SELECT expressions INTO target FROM ...;
```

target puede ser un registro, una variable de fila o una lista separada por comas de variables y campo de de registros o filas.

Si una fila o una lista de variables se usan como objetivo, los valores seleccionados han de coincidir exactamente con la estructura de los objetivos o se produciría un error de ejecución. La palabra clave FROM puede preceder a cualquier calificador valido, agrupación, ordenación, etc. que pueda pasarse a una sentencia SELECT.

Existe una variable especial llamada FOUND de tipo booleano, que puede usarse inmediatamente después de SELECT INTO para comprobar si una asignación ha tenido éxito.

```
SELECT * INTO myrec FROM EMP WHERE empname = myname;
IF NOT FOUND THEN
RAISE EXCEPTION "employee % not found", myname;
END IF;
```

Si la selección devuelve múltiples filas, solo la primera se mueve a los campos objetivo; todas las demás se descartan.

5.2.2. Creación de las funciones FDU en Postgres. Debido a que Postgresql (al menos hasta la versión 7.3.4) no ofrece soporte para incluir coherentemente funciones externas como funciones agregadas que el backend pueda reconocer y tratar directa y eficientemente, incluye entre sus directorios uno por defecto para que el usuario pueda una vez creadas sus funciones establecer aquí la jerarquía necesaria para su correcto funcionamiento, aunque bien se puede evitar este formalismo es recomendable hacer uso de el. Este es el directorio *Contrib* ubicado en *./postgresql/contrib.*

Es necesario para el funcionamiento de una función escrita en un lenguaje procedural basado en SQL como PL/pgSQL, registrar el lenguaje para su uso en una base de datos especifica. Esto se hace accediendo como usuario postgres (o un usuario con privilegios equivalentes) y dando desde el prompt ($) la instrucción:

CREATELANG PLPGSQL <<dbname>>

Donde *dbname* es el nombre de la Base de Datos específica.

Para que una FDU pueda ejecutarse es necesario además del código fuente, construir los archivos *.sql* donde se especifica la instrucción de instalación para Postgres y las fuentes. Así creación de la función se realiza accediendo a la base de datos, ejecutando los programas monitores de Postgres como psql (que permite introducir, editar y ejecutar comandos SQL interactivamente). Luego, se utiliza el comando de psql "\i" para leer peticiones a partir del archivo *\i /ruta_archivo/nombre_archivo.*

Con esta instrucción se crea la función, y se obtiene el mensaje de confirmación 'CREATE FUNCTION'.

A partir de este momento se puede invocar a la función con los parámetros requeridos. Para nuestro caso se ha dispuesto de un archivo iniciador por decirlo así.

El desarrollo de las funciones definidas por el usuario se ha realizado por conjuntos de a cuerdo a la fase que se busque realizar, así se ha dividido las funciones en seis grupo primero el grupo para el preprocesamiento en donde se realiza las tareas de selección de datos este grupo es manejado por el usuario, el segundo grupo en donde se depura, discretiza y transforma los datos a minar, el tercer grupo construye el modelo de clasificación empleando el algoritmo C4.5 el cuarto grupo construye el modelo usando el algoritmo Mate-Tree, el quinto grupo construye el modelo con el algoritmo Sliq y por último el sexto grupo que contiene las funciones para validar e interpretar el modelo generado.

Después de la instalación de la herramienta y si se prefiere trabajar en modo consola, para crear las funciones aquí construidas se utiliza el comando:

\i. /opt/Matekdd/functions/matekdd.sql.

5.3 IMPLEMENTACIÓN DE FUNCIONES PARA LA FASE DE PREPROCESAMIENTO.

Los archivos que contienen las fuentes de estas funciones están ubicadas en el directorio: /opt/Matekdd/functions/Preprocess/, y son estas funciones las que apoyan la fase de selección ya que el usuario final puede aplicarlas a una tabla para obtener un grupo de datos con características especificas. Corresponden a:

- process_del.sql: Contiene las fuentes de la función process_del().
- process_add.sql: Contiene las fuentes de la función process_add().
- process_dtz.sql: Contiene las fuentes de la función process_dtz().
- process_nlz.sql: Contiene las fuentes de la función process_nlz().
- process_fcd.sql: Contiene las fuentes de la función process_fcd().
- process_rdm.sql: Contiene las fuentes de la función process_rdm().

Para la utilización de las funciones, se ingresa como usuario Postgres o equivalente, a la base de datos a través de la terminal interactiva psql. Una vez dentro se ejecuta la instrucción:

i /opt/Matekdd/functions/*Process/process_del.sql*

i /opt/Matekdd/functions/*Process/process_add.sql*

i /opt/Matekdd/functions/*Process/process_dtz.sql*

i /opt/Matekdd/functions/*Process/process_nlz.sql*

i /opt/Matekdd/functions/*Process/process_fcd.sql*

i /opt/Matekdd/functions/*Process/process_rdm.sql*

Con estas instrucciones se crea las funciones, y obtiene el mensaje de confirmación 'CREATE FUNCTION' para cada una.

A partir de este momento ya es posible invocar a las funciones, haciendo uso de los parámetros requeridos.

5.3.1 Funcionamiento de la FDU process_del(). La función process_del() recibe como parámetros de entrada un tipo de dato VARCHAR que contiene el nombre de la tabla que se esta trabajando y un tipo de dato VARCHAR que contiene el nombre del atributo a eliminar. Ya que no es recomendable modificar la tabla original esta función hace una copia omitiendo el atributo seleccionado para eliminar.

Tabla 20. JugarTenis

Estado	Temperatura	Humedad	Viento	Jugar
Soleado	85	85	Débil	No
Soleado	80	90	Fuerte	No
Nublado	83	78	Débil	Si
Lluvioso	70	96	Débil	Si
Lluvioso	68	80	Débil	Si
Lluvioso	65	70	Fuerte	No
Nublado	64	65	Fuerte	Si
Soleado	72	95	Débil	No
Soleado	69	70	Débil	Si
Lluvioso	75	80	Débil	Si
Soleado	75	70	Fuerte	Si
Nublado	72	90	Fuerte	Si
Nublado	81	75	Débil	Si
Lluvioso	71	80	fuerte	No

Si por ejemplo sobre la tabla *"JugarTenis"* (Tabla 20.). Se aplica la función process_del('Jugar_Tenis','temperatura'), se obtiene como resultado la tabla *'fl_Jugar_Tenis'* (Tabla 21.).

Tabla 21. fl_JugarTenis Resultado de process_del()

Estado	Humedad	Viento	Jugar
Soleado	85	Débil	No
Soleado	90	Fuerte	No
Nublado	78	Débil	Si
Lluvioso	96	Débil	Si
Lluvioso	80	Débil	Si
Lluvioso	70	Fuerte	No
Nublado	65	Fuerte	Si
Soleado	95	Débil	No
Soleado	70	Débil	Si
Lluvioso	80	Débil	Si
Soleado	70	Fuerte	Si
Nublado	90	Fuerte	Si
Nublado	75	Débil	Si
Lluvioso	80	fuerte	No

5.3.2 Funcionamiento de la FDU process_add(). La función process_add() recibe como parámetros de entrada un tipo de dato VARCHAR que contiene el nombre de la tabla que se esta trabajando, un tipo de dato VARCHAR que contiene el nombre del nuevo atributo a adicionar, un tipo de dato VARCHAR que contiene la operación a aplicar, un tipo de dato INTEGER que contiene el número de un primer atributo, un tipo de dato FLOAT que puede contener el número de otro atributo o un número cualquiera para una operación matemática (suma, resta, multiplicación, división) y un tipo de datos INTEGER que contiene la opción del proceso a aplicar. Entonces, si el valor del último parámetro es cero se realiza una operación entre los dos atributos seleccionados y si el valor es igual a uno se realiza la operación entre un atributo y un valor. El resultado de esta va en un atributo que se adiciona de último a una copia de la tabla original.

Si por ejemplo sobre la tabla *"JugarTenis"* (Tabla 20.). Se aplica la función process_add('JugarTenis','div',2,2,'/',1), se obtiene como resultado la tabla *'fl_JugarTenis'* (Tabla 22.).

Tabla 22. fl_JugarTenis Resultado de process_add()

Estado	Temperatura	Humedad	Viento	Jugar	viv
Soleado	85	85	Débil	No	42
Soleado	80	90	Fuerte	No	40
Nublado	83	78	Débil	Si	41
Lluvioso	70	96	Débil	Si	35
Lluvioso	68	80	Débil	Si	34
Lluvioso	65	70	Fuerte	No	32
Nublado	64	65	Fuerte	Si	32
Soleado	72	95	Débil	No	66
Soleado	69	70	Débil	Si	34
Lluvioso	75	80	Débil	Si	37
Soleado	75	70	Fuerte	Si	37
Nublado	72	90	Fuerte	Si	36
Nublado	81	75	Débil	Si	40
Lluvioso	71	80	fuerte	No	35

5.3.3 Funcionamiento de la FDU process_dtz(). La función process_dtz() recibe como parámetros de entrada un tipo de dato VARCHAR que contiene el nombre de la tabla, un tipo de dato INTEGER que contiene el número de rangos dispuestos para discretización y un tipo de dato INTEGER que contiene el número del atributo a discretizar. Así, si el último atributo es igual a cero se realiza la discretización de todos los atributos de tipo numérico que tenga la tabla pero si este parámetro es mayor que cero indicara el número del atributo que se va a discretizar. La discretización consiste en reemplazar los valores continuos por el rango al que pertenece. Este cambio se lo realiza en una copia de la tabla. Además, se genera una tabla *pg_rangos* en la que se guarda la información correspondiente para la interpretación de las reglas.

Si por ejemplo sobre la tabla *"JugarTenis"* (Tabla 20.). Se aplica la función process_dtz('JugarTenis',3,2), se obtiene como resultado la tabla *'fl_Juga_Tenis'* (Tabla 23.).

Tabla 23. fl_JugarTenis Resultado de process_dtz()

Estado	Temperatura	Humedad	Viento	Jugar
Soleado	[78.000,85.000]	85	Débil	No
Soleado	[78.000,85.000]	90	Fuerte	No
Nublado	[78.000,85.000]	78	Débil	Si
Lluvioso	[64.000,71.000)	96	Débil	Si
Lluvioso	[64.000,71.000)	80	Débil	Si
Lluvioso	[64.000,71.000)	70	Fuerte	No
Nublado	[64.000,71.000)	65	Fuerte	Si
Soleado	[71.000,78.000)	95	Débil	No
Soleado	[64.000,71.000)	70	Débil	Si
Lluvioso	[71.000,78.000)	80	Débil	Si
Soleado	[71.000,78.000)	70	Fuerte	Si
Nublado	[71.000,78.000)	90	Fuerte	Si
Nublado	[78.000,85.000]	75	Débil	Si
Lluvioso	[71.000,78.000)	80	fuerte	No

Y si por ejemplo sobre la tabla *"JugarTenis"* (Tabla 20.). Se aplica la función process_dtz('JugarTenis',3,0), se obtiene como resultado la tabla *'fl_Juga_Tenis'* (Tabla 24.).

Tabla 24. fl_JugarTenis Resultado de process_dtz() – 2

Estado	Temperatura	Humedad	Viento	Jugar
Soleado	[78.000,85.000]	[75.333,85,666)	Débil	No
Soleado	[78.000,85.000]	[85,666,96,000]	Fuerte	No
Nublado	[78.000,85.000]	[75.333,85,666)	Débil	Si
Lluvioso	[64.000,71.000)	[85,666,96,000]	Débil	Si
Lluvioso	[64.000,71.000)	[75.333,85,666)	Débil	Si
Lluvioso	[64.000,71.000)	[65.000,75.333)	Fuerte	No
Nublado	[64.000,71.000)	[65.000,75.333)	Fuerte	Si
Soleado	[71.000,78.000)	[85,666,96,000]	Débil	No
Soleado	[64.000,71.000)	[65.000,75.333)	Débil	Si
Lluvioso	[71.000,78.000)	[75.333,85,666)	Débil	Si
Soleado	[71.000,78.000)	[65.000,75.333)	Fuerte	Si
Nublado	[71.000,78.000)	[85,666,96,000]	Fuerte	Si
Nublado	[78.000,85.000]	[65.000,75.333)	Débil	Si
Lluvioso	[71.000,78.000)	[75.333,85,666)	fuerte	No

A parte de la tabla con los datos discretizados esta la tabla *pg_rangos* (Tabla 25.)

Tabla 25. pg_rangos

nro	Atributo	Min	Max	equivalente
1	Temperatura	64.000	71.000	[64.000,71.000)
2	Temperatura	71.000	78.000	[71.000,78.000)
3	Temperatura	78.000	85.000	[78.000,85.000]
4	Humedad	65.000	75.333	[65.000,75.333)
5	Humedad	75.333	85.666	[75.333,85,666)
6	Humedad	85.666	96.000	[85,666,96,000]

5.3.4 Funcionamiento de la FDU process_nlz(). La función process_nlz() recibe como parámetros de entrada un tipo de dato VARCHAR que contiene el nombre de la tabla un tipo de dato INTEGER que contiene el número del atributo a normalizar. Así, si el último atributo es igual a cero se realiza la Normalización de todos los atributos de tipo numérico que tenga la tabla pero si este parámetro es mayor que cero indicara el número del atributo que se va a normalizar. La Normalización consiste en reemplazar los valores continuos por su equivalente en el rango 0 a 1. Este cambio se lo realiza en una copia de la tabla.

Si por ejemplo sobre la tabla "*Jugar_Tenis"* (Tabla 20.). Se aplica la función process_nlz('JugarTenis',3), se obtiene como resultado la tabla *'fl_Juga_Tenis'* (Tabla 26.).

Tabla 26. fl_JugarTenis Resultado de process_nlz()

Estado	Temperatura	Humedad	Viento	Jugar
Soleado	1.000	85	Débil	No
Soleado	0.941	90	Fuerte	No
Nublado	0.976	78	Débil	Si
Lluvioso	0.824	96	Débil	Si
Lluvioso	0.800	80	Débil	Si
Lluvioso	0.765	70	Fuerte	No
Nublado	0.753	65	Fuerte	Si
Soleado	0.847	95	Débil	No
Soleado	0.812	70	Débil	Si
Lluvioso	0.882	80	Débil	Si
Soleado	0.882	70	Fuerte	Si
Nublado	0.847	90	Fuerte	Si
Nublado	0.953	75	Débil	Si
Lluvioso	0.835	80	fuerte	No

Y si por ejemplo sobre la tabla *"JugarTenis"* (Tabla 20.). Se aplica la función process_nlz('JugarTenis',0), se obtiene como resultado la tabla *'fl_JugarTenis'* (Tabla 27.).

Tabla 27. fl_JugarTenis Resultado de process_nlz() – 2

Estado	Temperatura	Humedad	Viento	Jugar
Soleado	1.000	0.885	Débil	No
Soleado	0.941	0.938	Fuerte	No
Nublado	0.976	0.813	Débil	Si
Lluvioso	0.824	1.000	Débil	Si
Lluvioso	0.800	0.833	Débil	Si
Lluvioso	0.765	0.729	Fuerte	No
Nublado	0.753	0.677	Fuerte	Si
Soleado	0.847	0.990	Débil	No
Soleado	0.812	0.729	Débil	Si
Lluvioso	0.882	0.833	Débil	Si
Soleado	0.882	0.729	Fuerte	Si
Nublado	0.847	0.938	Fuerte	Si
Nublado	0.953	0.781	Débil	Si
Lluvioso	0.835	0.833	fuerte	No

5.3.5 Funcionamiento de la FDU process_fcd(). La función process_fcd() recibe como parámetros de entrada un tipo de dato VARCHAR que contiene el nombre de la tabla que se esta trabajando y un tipo de dato VARCHAR que contiene la condición para la selección de registros. Esta función hace una copia de la tabla original y en ella copia las instancias que cumplen la condición indicada y retorna el nombre de la tabla creada.

Si por ejemplo sobre la tabla *"Jugar_Tenis"* (Tabla 20.). Se aplica la función process_fcd('Jugar_Tenis','humedad>70'), se obtiene como resultado la tabla *'fl_JugaTenis'* (Tabla 28.).

Tabla 28. fl_JugarTenis Resultado de process_fcd()

Estado	Temperatura	Humedad	Viento	Jugar	viv
Soleado	85	85	Débil	No	42
Soleado	80	90	Fuerte	No	40
Nublado	83	78	Débil	Si	41
Lluvioso	70	96	Débil	Si	35
Lluvioso	68	80	Débil	Si	34
Soleado	72	95	Débil	No	66
Lluvioso	75	80	Débil	Si	37
Nublado	72	90	Fuerte	Si	36
Nublado	81	75	Débil	Si	40
Lluvioso	71	80	fuerte	No	35

5.3.6 Funcionamiento de la FDU process_rd(). La función process__rdm() recibe como parámetros de entrada un tipo de dato VARCHAR que contiene el nombre de la tabla y junto con un tipo de dato INTEGER que contiene el número del tuplas para el nuevo conjunto de datos.

Si por ejemplo sobre la tabla *"Juga_Tenis"* (Tabla 20.). Se aplica la función process_rdm('JugarTenis',7), se obtiene como resultado la tabla *'fl_JugarTenis'* (Tabla 29.).

Tabla 29. fl_Jugar_Tenis resultado de process_rdm()

Estado	Temperatura	Humedad	Viento	Jugar	viv
Soleado	85	85	Débil	No	42
Nublado	83	78	Débil	Si	41
Lluvioso	65	70	Fuerte	No	32
Nublado	64	65	Fuerte	Si	32
Lluvioso	75	80	Débil	Si	37
Soleado	75	70	Fuerte	Si	37
Lluvioso	71	80	fuerte	No	35

5.4 IMPLEMENTACIÓN DE FUNCIONES PARA LA FASE DE TRANSFORMACIÓN.

Los archivos que contienen las fuentes de estas funciones deben estar ubicadas en el directorio: */opt/Matekdd/functions/Transform*, tienen como propósito depurar, discretizar y transformar los datos para su posterior utilización y corresponden a:

- purify.sql: Contiene las fuentes de la función purify().
- transform.sql: Contiene las fuentes de la función transform().
- filter.sql: Contiene las fuentes de la función filter().

Para la utilización de las funciones, se ingresa como usuario Postgres o equivalente, a la base de datos a través de la terminal interactiva psql. Una vez dentro se ejecuta la instrucción:

\i /opt/Matekdd/functions/Transform/purify.sql

\i /opt/Matekdd/functions/Transform/transform.sql

\i /opt/Matekdd/functions/Transform/filter.sql

Con estas instrucciones se crea las funciones, y obtiene el mensaje de confirmación 'CREATE FUNCTION' para cada una.

A partir de este momento ya es posible invocar a las funciones, haciendo uso de los parámetros requeridos. Cabe resaltar que la función que divide el conjunto de datos en datos de entrenamiento y prueba es la misma función process_rdm() mencionada anteriormente.

5.4.1 Funcionamiento de la FDU purify(). La función purify() recibe como parámetro de entrada un tipo de dato VARCHAR que contiene el nombre de la tabla a la cual se le aplicara el proceso y genera como resultado una tabla *"kdd_list"*. Esta función se encarga de depura la tabla validando los valores nulos y les asigna un valor promedio si es un atributo de tipo numérico y si es un tributo categórico le asigna el que mayor frecuencia obtenga. En el caso de que no haya necesidad de depurar la tabla simplemente realiza una copia de la tabla.

Si por ejemplo sobre la tabla *"JugarTenis3"* (Tabla 30.). Se aplica la función purify('JugarTenis3'), se obtiene como resultado la tabla *'kdd_list'* (Tabla 31.) ya depurada.

Tabla 30. Jugar_Tenis3

Estado	Temperatura	Humedad	Viento	Jugar	viv
Soleado	85	85	Débil	No	42
Soleado	80	90	Fuerte	No	40
Nublado	83	78	Débil	Si	41
Lluvioso	70		Débil	Si	35
Lluvioso	68	80	Débil	Si	34
Lluvioso		70	Fuerte	No	32
Nublado	64	65	Fuerte	Si	32
Soleado	72	95	Débil	No	66
Soleado	69	70	Débil	Si	34
Lluvioso		80	Débil	Si	37
Soleado	75	70	Fuerte	Si	37
Nublado	72	90	Fuerte	Si	36
Nublado	81		Débil	Si	40
Lluvioso	71	80	fuerte	No	35

Tabla 31. kdd_list Resultado de purify()

Estado	Temperatura	Humedad	Viento	Jugar	viv
Soleado	85	85	Débil	No	42
Soleado	80	90	Fuerte	No	40
Nublado	83	78	Débil	Si	41

Lluvioso	70	96	Débil	Si	35
Lluvioso	68	80	Débil	Si	34
Lluvioso	65	70	Fuerte	No	32
Nublado	64	65	Fuerte	Si	32
Soleado	72	95	Débil	No	66
Soleado	69	70	Débil	Si	34
Lluvioso	75	80	Débil	Si	37
Soleado	75	70	Fuerte	Si	37
Nublado	72	90	Fuerte	Si	36
Nublado	81	75	Débil	Si	40
Lluvioso	71	80	fuerte	No	35

5.4.2 Funcionamiento de la FDU transform(). La función transform() recibe como parámetro de entrada un tipo de dato VARCHAR que contiene el nombre del atributo clase.

Recorre la tabla para transformar los valores contenidos. Convierte la tabla *"kdd_list"* en una tabla transformada *"kdd_ent"* que contienen únicamente valores numéricos y organiza los atributo de tal forma que el atributo clase quede al final, Crea también la tabla *'kdd_values'*, que guarda los parámetros de transformación utilizados, necesarios para la conversión cuando sea necesario expresar los valores resultantes finales del proceso de clasificación en términos de los valores originales.

Por ejemplo, sobre la tabla *"kdd_list"* de la Tabla 24 (debidamente depurada) Al aplicar la función transform('jugar'), se obtiene la tabla *"kdd_ent"* ya transformada (Tabla 32.) y la tabla *'kdd_values'* (Tabla 33.).

Tabla 32. kdd_ent

Estado	Temperatura	Humedad	Viento	Jugar
0	2	1	0	0
0	2	2	1	0
2	2	1	0	1
1	0	2	0	1
1	0	1	0	1
1	0	0	1	0
2	0	0	1	1
0	1	2	0	0
0	0	0	0	1
1	1	1	0	1
0	1	0	1	1
2	1	2	1	1
2	2	0	0	1
2	1	1	1	0

Tabla 33. kdd_values

Nro	Atributo	Valor	Discret	nclase
1	Estado	Soleado	0	
2	Estado	Lluvioso	1	
3	Estado	Nublado	2	3
4	temperatura	[64.000,71.000)	0	
5	Temperatura	[71.000,78.000)	1	
6	Temperatura	[78.000,85.000]	2	3
7	Humedad	[65.000,75.333)	0	
8	Humedad	[75.333,85.666)	1	
9	Humedad	[85.666,96.000]	2	3
10	Viento	Débil	0	
11	Viento	Fuerte	1	2
12	Jugar	No	0	
13	Jugar	Si	1	2

Otro ejemplo que se puede utilizar para la función transform() es la tabla *'JugarTenis2'* (Tabla 34.). Al aplicar la función transform('jugar'), se obtiene la tabla en *"kdd_ent"* (Tabla 35.) y la tabla *'kdd_values'* (Tabla 36.).

Tabla 34. JugarTenis2

Estado	Temperatura	Humedad	Viento	Jugar
Soleado	Caliente	Alta	Débil	No
Soleado	Caliente	Alta	Fuerte	No
Nublado	Caliente	Alta	Débil	Si
Lluvioso	Templado	Alta	Débil	Si
Lluvioso	Fresco	Normal	Débil	Si
Lluvioso	Fresco	Normal	Fuerte	No
Nublado	Fresco	Normal	Fuerte	Si
Soleado	Templado	Alta	Débil	No
Soleado	Fresco	Normal	Débil	Si
Lluvioso	Templado	Normal	Débil	Si
Soleado	Templado	Normal	Fuerte	Si
Nublado	Templado	Alta	Fuerte	Si
Nublado	Caliente	Normal	Débil	Si
Lluvioso	templado	Alta	fuerte	No

Tabla 35. kdd_ent 2

Estado	Temperatura	Humedad	Viento	Jugar
0	0	0	0	0
0	0	0	1	0
2	0	0	0	1
1	2	0	0	1
1	1	1	0	1

1	1	1	1	0
2	1	1	1	1
0	2	0	0	0
0	1	1	0	1
1	2	1	0	1
0	2	1	1	1
2	2	0	1	1
2	0	1	0	1
2	2	0	1	0

Tabla 36. Tabla kdd_values 2

Nro	Atributo	Valor	Discret	nclase
1	Estado	soleado	0	
2	Estado	lluvioso	1	
3	Estado	nublado	2	3
4	temperatura	Caliente	0	
5	Temperatura	Fresco	1	
6	Temperatura	Templado	2	3
7	Humedad	Alta	0	
8	Humedad	Normal	1	2
9	Viento	Débil	0	
10	Viento	Fuerte	1	2
11	Jugar	No	0	
11	Jugar	Si	1	2

5.4.3 Funcionamiento de la FDU filter(). La función filter() recibe como parámetro de entrada un tipo de dato VARCHAR que contiene el nombre del atributo clase. Trabaja con la tabla *"kdd_ent"*- Se encarga de validar las contradicciones que se presenten el conjunto de datos. Mas exactamente si encuentra una condición que pertenezca a más de una clase, calcula la probabilidad de ocurrencia en las clases y deja en la tabla los registros que cumplen con la condición y la clase con mayor probabilidad.

5.5 IMPLEMENTACIÓN DEL ALGORITMO C4.5 PARA LA FASE DE MINERÍA

Los archivos que contienen las fuentes de estas funciones están ubicadas en el directorio: */opt/Matekdd/functions/AlgorihtmC45/*, tienen como propósito generar reglas de clasificación aplicando a los datos de una tabla el algoritmo C4.5 y corresponden a:

- c45_algorithm.sql: Contiene las fuentes de la función principal c*45().*
- c45_entro.sql: Contiene las fuentes de la función c45_entr().
- c45_gain.sql: Contiene las fuentes de la función c45_gain().

- c45_tree.sql: Contiene las fuentes de la función c45_tree().

- view_set.sql: Contiene las fuentes de la función view_set().

Para la utilización de la función C***45()***, se ingresa como usuario Postgres o equivalente, a la base de datos a través de la terminal interactiva psql. Una vez dentro se ejecuta la instrucción:

\i /opt/Matekdd/functions/AlgorihtmC45/c45_algorithm.sql

\i /opt/Matekdd/functions/AlgorihtmC45/c45_entro.sql

\i /opt/Matekdd/functions/AlgorihtmC45/c45_gain.sql

\i /opt/Matekdd/functions/AlgorihtmC45/c45_tree.sql

\i /opt/Matekdd/functions//AlgorihtmC45/view_set.sql

Con estas instrucciones se crea las funciones, y obtiene el mensaje de confirmación 'CREATE FUNCTION' para cada una.

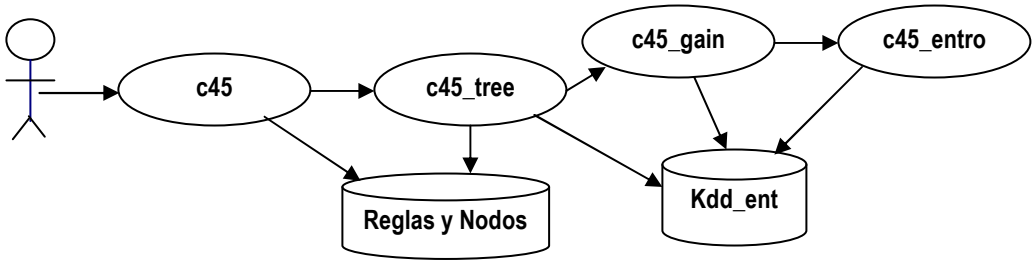
A partir de este momento ya es posible invocar a la función c45() haciendo uso de los parámetros requeridos.

5.5.1 Funcionamiento de la FDU c45(). La función c45() recibe como parámetro un tipo de dato VARCHAR que es el nombre del atributo clase del conjunto de datos. Esta es la función principal del algoritmo y trabaja con una tabla *''kdd_ent''* en donde deben estar los datos de entrenamiento debidamente transformados. También requiere de las sub funciones c45_entro(), c45_gain(), c45_tree() y view_set(). Esta función se encarga primero que todo de crear físicamente las tablas *'c45_rules'*, *'c45_nodes'* como las tablas auxiliares que utiliza en el transcurso del proceso de modelado. Seguidamente llama a la función c45_entro() la cual retorna la entropía de la clase y sigue buscando el atributo que va a ser utilizado para el nodo raíz utilizando la función c45_gain(). Y por último llama a la función c45_tree() quien es la que termina de crear el modelo de clasificación.

Tanto esta función como c45_tree() utilizan una tabla temporal llamada "pg_columna" que actúa como un vector y lleva el orden y el número de los atributos que componen una camino del árbol (o sea, una rama).

En la Figura 37 se muestra las relaciones de las funciones necesarias en el desarrollo del algoritmo C4.5.

Figura 37. Diagrama de interrelación de funciones en C4.5



5.5.2 Funcionamiento de la FDU c45_tree(). La función c45_tree() recibe como parámetro un tipo de dato INTEGER que contiene el número de atributos hasta el momento van conformando una condición.

Esta función es llamada por la función principal que es c45() o por si misma. Empieza insertando el nodo actual en la tabla de nodos *"c45_nodes"* luego valida que este nodo sea una hoja Terminal, si esto es así, construye la reglas recorriendo la tabla auxiliar *"pg_columna"* y la inserta en la tabla de reglas *"c45_rules"*. Si no es un nodo terminal continúa el análisis primero por la izquierda y luego siguiendo por la derecha, tomando los atributos que no han sido utilizados y aquel atributo que tenga la mayor ganancia (para obtener la ganancia de información utiliza la función c45_gain()) es el nodo que sigue en el árbol en seguida llama de nuevo a la función c45_tree().

5.5.3 Funcionamiento de la FDU c45_entro(). La función c45_entro() recibe como parámetro un tipo de dato INTEGER que contiene el número de la opción solicitada, más claramente si se requiere la entropía de la clase o la entropía de un grupo especifico de atributos, y junto con un tipo de datos VARCHAR que contiene el nombre del atributo clase.

Esta función calcula la entropía aplicando la formula de entropía de shannon, indicada en el capítulo 3 para el algoritmo C4.5, y retorna el valor de la entropía que es un dato de tipo NUMERIC.

5.5.4 Funcionamiento de la FDU c45_gain(). La función c45_gain() recibe como parámetro un tipo de dato FLOAT que contiene un valor de entropía, dato necesario para obtener la ganancia, junto un tipo de dato VARCHAR que contiene el nombre del atributo clase de la tabla. Esta función calcula la ganancia de un atributo utilizando la función entropía(), utiliza la formula indicada en el capítulo 3 para el algoritmoC4.5.

5.5.5 Funcionamiento de la FDU view_set(). La función view_set(). Esta función selecciona un subgrupo de datos relacionados al nodo en turno para su posterior análisis. Crea la vista view1 con la misma estructura de la tabla kdd_ent.

5.5.6. Ejemplo de la función c45(). Usando la tabla *'kdd_ent'* transformada (Tabla 32.). Se aplica la función c45('jugar') y como resultado se tiene las tablas *'c45_rules'* (Tabla 37.) y *'c45_nodes'* (Tabla 38.).

Tabla 37. Tabla c45_rules

Id	Atributo	valor
1	1	0
1	3	0
1	5	1
2	1	0
2	3	1
2	5	0
3	1	1
3	4	0
3	5	1
4	1	1
4	4	1
4	5	0
5	1	2
5	5	1

Tabla 38. c45_nodes

Nodo	Padre	Atributo	Valor	Hijo	Hemí	Clase
0	-1	1	-1	-1	-1	-1
1	0	1	0	3	2	-1
2	0	1	1	5	-1	-1
3	1	3	0	-1	4	1
4	1	3	1	-1	-1	0
5	4	4	0	-1	6	1
6	4	4	1	-1	7	0
7	0	1	2	-1	-1	1

La estructura de las tablas *'c45_rules'* y *'c45_nodes'* se muestra en las Tablas 37 y 38 respectivamente. La tabla *'c45_rules'* presenta los nodos relacionados en una misma regla de forma secuencial. Todos los registros que conforman una regla tienen el mismo valor en el atributo id, que indica el número de la regla. El valor constante '-999' indica el caso especial en que el atributo puede ser cualquiera de los posibles valores del atributo clase.

La tabla *'c45_nodes'* especifica las relaciones entre los nodos del árbol, y los valores útiles. Todos los valores están transformados, y por tanto pueden traducirse en resultados reales y coherentes. En este caso el valor constante '-999' tiene el mismo significado que en la tabla *'c45_rules'*.

Esta estructura será igualmente utilizada para mostrar los resultados del modelo obtenido con el algoritmo Mate-Tree que sigue a continuación.

5.6 IMPLEMENTACIÓN DEL ALGORITMO MATE-TREE PARA LA FASE DE MINERÍA

Los archivos que contienen las fuentes de estas funciones están ubicadas en el directorio: */opt/Matekdd/functions/AlgorihtmMate/*, tienen como propósito generar reglas de clasificación aplicando a los datos de una tabla el algoritmo Mate-Tree y corresponden a:

- matetree.sql: Contiene las fuentes de la función principal matetree*()*.
- mate_tree.sql: Contiene las fuentes de la función principal mate_tree*()*.
- mateby.sql: Contiene las fuentes de la función mateby().
- posiciones.sql: Contiene las fuentes de la función controlposicion().
- mate_gain.sql: Contiene las fuentes de la función principal mate_gain*()*.

Para la utilización de la función **m*atetree()***, se ingresa como usuario Postgres o equivalente, a la base de datos a través de la terminal interactiva psql. Una vez dentro se ejecuta las instrucciones:

\i /opt/Matekdd/functions/AlgorihtmMate/mate_gain.sql

\i /opt/Matekdd/functions/AlgorihtmMate/mateby.sql

\i /opt/Matekdd/functions/AlgorihtmMate/posicion.sql

\i /opt/Matekdd/functions/AlgorihtmMate/matetree.sql

\i /opt/Matekdd/functions/AlgorihtmMate/mate_tree.sql

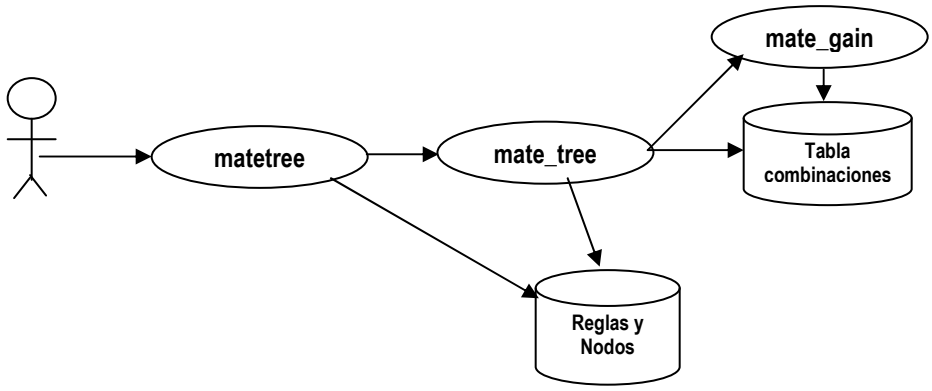
Con estas instrucciones se crea las funciones, y obtiene el mensaje de confirmación 'CREATE FUNCTION' para cada una.

A partir de este momento ya es posible invocar a la función matetree() o cada una de las funciones: mateby(), luego entro() y por último gain(), haciendo uso de los parámetros requeridos.

5.6.1 Funcionamiento de la FDU matetree(). La función matetree() recibe como parámetro un tipo de dato VARCHAR que contiene el nombre del atributo clase de la tabla. Esta función tiene como fin generar un modelo de clasificación. Esta es la función principal la cual utiliza las funciones mateby() y entro() y luego junto las funciones mate_tree() y mate_gain construye el modelo de decisión.

En la Figura 38 se muestra las relaciones de las funciones necesarias en el desarrollo del algoritmo Mate-Tree.

Figura 38. Diagrama de interrelación de funciones en Mate-Tree



A continuación se describe el funcionamiento de cada una de las funciones que componen la implementación del algoritmo Mate-Tree y se muestra el ejemplo paralelamente.

5.6.2 Funcionamiento de la FDU mateby(). La función mateby() recibe como parámetro un tipo de dato VARCHAR que contiene el nombre del atributo clase de la tabla y un tipo de dato INTEGER que contiene el valor mínimo de soporte de prepoda. Esta función tiene como fin generar por cada una de las tuplas todas las posibles combinaciones formadas por los valores no nulos de los atributos decisión y el atributo clase de la tabla tratada y además los cuenta, además debe actuar como un proceso de prepoda.

Por ejemplo, sobre la tabla *'kdd_ent'* (Tabla 35). Al aplicar la función mateby('jugar') sobre ésta tabla se obtiene la tabla *'mate_clase'* (Tabla 39).

Tabla 39. mate_clase

Estado	Temperatura	Humedad	Viento	Jugar	Count
0	0	0	0	1	1
0	0	0		1	1
0	0		0	1	1
0	0			1	1

0	1	0	1	1	1
0	1	0		1	1
0	1	2	0	0	1
0	1	2		0	1
0	1		0	0	1
0	1		1	1	1
0	1			0	1
.					
.					
..					

5.6.3 Funcionamiento de la FDU Mate_gain(). La función mate_gain() trabaja con la tabla *'mate_clase'* (Tabla 39.). Calcula la ganancia de información de los atributos decisión que cumplan unas determinadas característica y devuelve este valor la función mate_tree().

5.6.4 Funcionamiento de la FDU mate_tree(). La función mate_gain(). Es la función que se encarga de seguir los pasos lógicos para encontrar el nodo en tuno.

5.6.5. Ejemplo de la función matetree(). Usando la tabla *'kdd_ent'* transformada (Tabla 32.). Se aplica la función matetree('jugar') y como resultado se tiene las tablas *'mate_rules'* (Tabla 40.) y *'mate_nodes'* (Tabla 41.). Estas tablas utilizan la misma estructura que las tablas generadas por C4.5.

Tabla 40. mate_rules

Id	Atributo	valor
1	1	0
1	3	0
1	5	1
2	1	0
2	3	1
2	5	0
3	1	1
3	4	0
3	5	1
4	1	1
4	4	1
4	5	0
5	1	2
5	5	1

Tabla 41. mate_nodes

Nodo	Padre	Atributo	Valor	Hijo	Hem	Clase
0	-1	1	-1	-1	-1	-1
1	0	1	0	3	2	-1
2	0	1	1	5	-1	-1
3	1	3	0	-1	4	1
4	1	3	1	-1	-1	0
5	4	4	0	-1	6	1
6	4	4	1	-1	7	0
7	0	1	2	-1	-1	1

5.7 IMPLEMENTACIÓN DEL ALGORITMO SLIQ PARA LA FASE DE MINERÍA

Los archivos que contienen las fuentes de estas funciones están ubicadas en el directorio: */opt/Matekdd/functions/*AlgorithmSliq, tienen como propósito generar reglas de clasificación aplicando a los datos de una tabla el algoritmo Sliq y corresponden a:

- sliq_algorithm.sql: Contiene las fuentes de la función principal sliq*().*
- sliq_tree.sql: Contiene las fuentes de la función sliq_tree.
- sliq_gini.sql: Contiene las fuentes de la función sliq_gini().
- sliq_combinar.sql: Contiene las fuentes de la función sliq_combinar().

Para la utilización de la función ***sliq()***, se ingresa como usuario Postgres o equivalente, a la base de datos a través de la terminal interactiva psql. Una vez dentro se ejecuta la instrucción:

\i */opt/Matekdd/functions/*AlgorithmSliq *sliq_algorithm.sql*

\i */opt/Matekdd/functions/*AlgorithmSliq *sliq_tree.sql*

\i */opt/Matekdd/functions/*AlgorithmSliq *sliq_gini.sql*

\i */opt/Matekdd/functions/*AlgorithmSliq *sliq_combinar.sql*

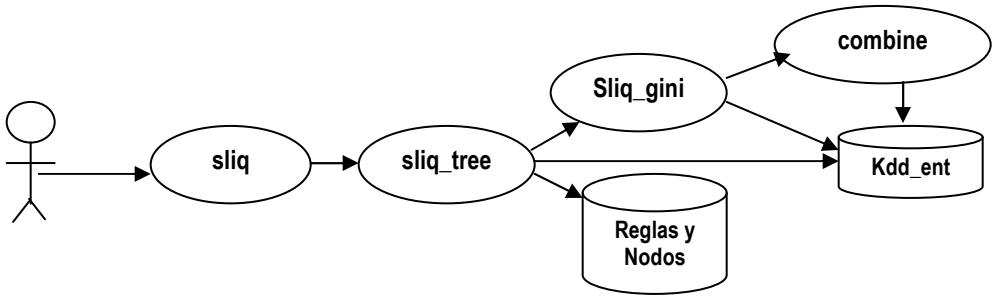
Con estas instrucciones se crea las funciones, y obtiene el mensaje de confirmación 'CREATE FUNCTION' para cada una.

A partir de este momento ya es posible invocar a la función sliq() haciendo uso de los parámetros requeridos.

5.7.1 Funcionamiento de la FDU sliq(). La función sliq() recibe como parámetro un tipo de dato VARCHAR que contiene el nombre del atributo clase de la tabla. Esta función trabaja sobre la tabla *"kdd_ent"* que es la tiene los datos de entrenamiento debidamente depurados y ya que este algoritmo trabaja tanto con atributo categóricos como con atributos numéricos no es necesario discretizar ni transformar su valores. Y empieza creando las tablas *"sliq_rules"* y *"sliq_nodes"* para las reglas y los nodos del modelo respectivamente así como las tablas auxiliares que son utilizadas en el proceso, también llama a la función sliq_gini() con el fin de encontrar el nodo raíz y luego llama a la función sliq_tree() la cual se encarga de el resto del árbol de decisión.

En la Figura 39 se muestra las relaciones de las funciones necesarias en el desarrollo del algoritmo Sliq.

Figura 39. Diagrama de interrelación de funciones en Sliq



A continuación se describe el funcionamiento de cada una de las funciones que componen la implementación del algoritmo Sliq.

5.7.2 Funcionamiento de la FDU sliq_tree(). La función sliq_tree() recibe como parámetro un tipo de dato VARCHAR que contiene el nombre del atributo clase de la tabla y un tipo de dato INTEGER que contiene la opción a aplicar.

Esta función es llamada por la función principal que es sliq() o por si misma. Empieza insertando el nodo actual en la tabla de nodos *"sliq_nodes"* luego valida que este nodo sea una hoja Terminal, si esto es así, construye la reglas recorriendo la tabla auxiliar *"pg_columna"* y la inserta en la tabla de reglas *"sliq_rules"*. Si no es un nodo terminal continúa el análisis usando la función sliq_gini() y sliq_set().

5.7.3 Funcionamiento de la FDU sliq_gini(). La función sliq_gini() recibe como parámetro un tipo de dato VARCHAR que contiene el nombre del atributo clase de la tabla. Esta función tiene como fin encontrar el valor del próximo nodo calculando las ganancias

de cada atributo. Así, si el atributo es numérico calcula la ganancia de los valores que estén entre v_i y v_{i+1}, donde los v_i son los valores ordenados de un atributo, si el atributo es categórico se llama a la función sliq_combinar() y calcula la ganancia de los subconjuntos diferentes de un atributo (es decir, 2^n donde es el número de valores del atributo). De estos se selecciona el de menor ganancia y se ubica la condición en el nodo correspondiente.

5.7.4 Funcionamiento de la FDU sliq_combinar(). La función sliq_combinar(). Esta función tiene como fin generar todas las posibles combinaciones entre los valores de un atributo categórico.

5.7.5 Primer ejemplo de la Función sliq(). Ya que este algoritmo no requiere de transformación de datos se aplicara sobre la tabla *'Jugar_Tenis'* (Tabla 20.) que para nuestro ejemplo será la tabla *'kdd_ent'*. Al aplicar la función sliq('jugar') sobre ésta tabla se obtiene la tabla *'sliq_rules'* (Tabla 42.) y la tabla *"sliq_nodes"* (Tabla 43.).

Tabla 42. sliq_rules 1

N	Id	Atributo	Valor	Com	1tipo
1	1	Estado	Nublado	=	2
1	1	Jugar	Si	=	2
2	1	Estado	Nublado	!=	2
2	1	Temperatura	77.5	<=	1
2	1	Estado	Lluvioso	=	2
2	1	Viento	Débil	=	2
2	1	Jugar	Si	=	2
3	1	Estado	Nublado	!=	2
3	1	Temperatura	77.5	<=	1
3	1	Estado	Lluvioso	=	2
3	1	Viento	Débil	!=	2
3	1	Jugar	No	=	2
4	1	Estado	Nublado	!=	2
4	1	Temperatura	77.5	<=	1
4	1	Estado	Lluvioso	!=	2
4	1	Humedad	82.5	<=	1
4	1	Jugar	Si	=	2
5	1	Estado	Nublado	!=	2
5	1	Temperatura	77.5	<=	1
5	1	Estado	Lluvioso	!=	2
5	1	Humedad	82.5	>	1
5	1	jugar	No	=	2
6	1	Estado	Nublado	!=	2
6	1	Temperatura	77.5	>	1
6	1	jugar	No	=	2

Tabla 43. Tabla sliq_nodes 1

Nodo	Padre	Atributo	Valor	Com	Hijo	Herm	clase
1	-1	Estado	Nublado	=	-1	-1	Si
2	1	Temperatura	77.5	<=	4	3	-1
3	1	Temperatura	77.5	>	-1	-1	No
4	2	Estado	Lluvioso	=	5		
5	4	Viento	Débil	=			
6	5	Jugar	Si	=			
7	5	Jugar	No	=			
8	4	Humedad	82.5	<=			
9	8	Jugar	Si	=			
10	8	Jugar	No	=			

La estructura de las tablas *'sliq_rules'* y *'sliq_nodes'* se muestra en las Tablas 42 y 43 respectivamente. La tabla *'sliq_rules'* presenta los nodos relacionados en una misma regla de forma secuencial. Todos los registros que conforman una regla tienen el mismo valor en el atributo *'id'*, que indica el número de la regla. El campo *'tipo'* contiene el tipo de dato a comparar (1 indica tipo numérico y 2 indica tipo categórico) y el campo '***com***' indica la comparación del caso.

La tabla *'sliq_nodes'* especifica las relaciones entre los nodos del árbol, y los valores útiles. Estos resultados esta en valores originales por lo cual no se debe hacer ninguna post transformación de las tablas.

5.7.6 Segundo ejemplo de la función sliq(). Sobre la tabla *'Jugar_Tenis2'* (Tabla 34.) que para nuestro ejemplo será la tabla *'kdd_ent'*. Al aplicar la función Sliq('jugar') sobre ésta tabla se obtiene la tabla *'sliq_rules'* (Tabla 44.) y la tabla *"sliq_nodes"* (Tabla 45).

Tabla 44 sliq_rules 2

N	Id	Atributo	Valor	Com	1tipo
1	1	Estado	Nublado	=	2
1	1	Jugar	Si	=	2
2	1	Estado	Nublado	!=	2
2	1	Temperatura	Caliente	=	2
2	1	Jugar	No	=	2
3	1	Estado	Nublado	!=	2
3	1	Temperatura	Caliente	!=	2
3	1	Estado	Lluvioso	=	2
3	1	Viento	Débil	=	2
3	1	Jugar	Si	=	2
4	1	Estado	Nublado	!=	2
4	1	Temperatura	Caliente	!=	2
4	1	Estado	Lluvioso	=	2

4	1	Viento	Débil	!=	2
4	1	Jugar	No	=	2
5	1	Estado	Nublado	!=	2
5	1	Temperatura	Caliente	!=	2
5	1	Estado	Lluvioso	!=	2
5	1	Humedad	Normal	=	2
5	1	Jugar	Si	=	2
6	1	Estado	Nublado	!=	2
6	1	Temperatura	Caliente	!=	2
6	1	Estado	Lluvioso	!=	2
6	1	Humead	Normal	=	2
6	1	jugar	No	=	2

Tabla 45. sliq_nodes 2

Nodo	Padre	Hijo	Atributo	Valor	Com
1	-1	0	Estado	Nublado	=
2	1	1	Jugar	Si	=
3	1	2	Temperatura	Caliente	=
4	3	1	Jugar	No	=
5	3	2	Estado	Lluvioso	=
6	5	1	Viento	Débil	=
7	6	1	Jugar	Si	=
8	6	2	Jugar	No	=
9	5	2	Humedad	Normal	=
10	9	1	Jugar	Si	=
11	9	2	jugar	No	=

5.8 IMPLEMENTACIÓN DE LAS FUNCIONES PARA LA FASE DE EVALUACIÓN.

Los archivos que contienen las fuentes de estas funciones están ubicadas en el directorio: *i /opt/Matekdd/functions/V*alid, tienen como propósito probar, traducir e interpretar un modelo de clasificación y corresponden a:

- test_rules.sql: Contiene las fuentes de la función test_rules().
- test_sliq.sql: Contiene las fuentes de la función test_sliq().
- translate_rules.sql: Contiene las fuentes de la función translate_rules().
- translate_inte.sql: Contiene las fuentes de la función translate_inte()..
- arm_rules.sql: Contiene las fuentes de la función arm_rules().

Para la utilización de las funciones, se ingresa como usuario Postgres o equivalente, a la base de datos a través de la terminal interactiva psql. Una vez dentro se ejecuta la instrucción:

\i */opt/Matekdd/functions/V*alid */test_rules.sql*

\i */opt/Matekdd/functions/V*alid */test_sliq.sql*

\i */opt/Matekdd/functions/V*alid */translate_rules.sql*

\i */opt/Matekdd/functions/V*alid */translate_inte.sql*

\i */opt/Matekdd/functions/V*alid */arm_rules.sql*

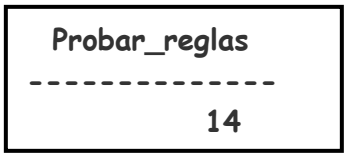
Con estas instrucciones se crea las funciones, y obtiene el mensaje de confirmación 'CREATE FUNCTION' para cada una.

A partir de este momento ya es posible invocar a las funciones, haciendo uso de los parámetros requeridos.

5.8.1 Funcionamiento de la FDU test_rules(). La función test_rules() recibe como parámetro un tipo de dato VARCHAR que contiene el nombre del modelo ('c45'/'mate') a validar (solo se aplica a los modelos de los algoritmos C4.5 y Mate-Tree), y es la cargada de probar el modelo armando las reglas y consultado su coincidencia en la tabla de prueba *'kdd_pru'* y devolver el número de incidencia de estas reglas en la tabla, es decir, el número de tuplas o registros que fueron correctamente clasificados.

Retomando el ejemplo del modelo construido con el algoritmo C4.5 en donde se obtuvo como resultado las tablas de reglas y de nodos *'c45_rules'* (Tabla 37) y *'kdd_nodes'* (Tabla 38) respectivamente. Aplicando la función probar_reglas() el resultado es el mostrado en la Figura 40. (Este modelo fue probado con el mismo conjunto de datos que se uso para el entrenamiento).

Figura 40. Resultado de test_rules()



5.8.2 Funcionamiento de la FDU translate_rules(). La función translate_rules() recibe como parámetro un tipo de dato INTEGER que contiene el número de rangos de utilizado para el proceso de discretización junto con un tipo de dato VARCHAR que contiene el

nombre del modelo (‘c45’/’mate’) a traducir (solo se aplica a los modelos de los algoritmos C4.5 y Mate-Tree), y es la encargada de transformar a su estado inicial los valores de las tablas utilizando las tablas *‘kdd_values’* también llama a la función interpretar() que utilizando la tabla *‘kdd_rangos’* interpreta los valores anteriormente discretizados.

Continuando con el modelo C4.5 y junto con las tablas de rangos y valores de transformación *‘pg_rangos’* de la Tabla 25 y *‘kdd_values’* de la Tabla 36. Si Se aplica la función traducir se tiene como resultado las misma tabla *‘c45_rules’* (Tabla 37.) y *‘c45_nodes’* (Tabla 38.) pero traducidas a sus valores originales (Tablas 46 y 47 respectivamente).

Tabla 46. c45_rules del modelo C4.5 traducida

Id	Atributo	valor
1	Estado	Soleado
1	Humedad	<= 80.5
1	Jugar	Si
2	Estado	Soleado
2	Humedad	>80.5
2	Jugar	No
3	Estado	Lluvioso
3	Viento	Débil
3	Jugar	Si
4	Estado	Lluvioso
4	Viento	Fuerte
4	Jugar	No
5	Estado	Nublado
5	Jugar	Si

Tabla 47. c45_nodes del modelo C4.5 traducida

Nodo	Padre	Atributo	Valor	Clase
0	-1	Estado	-1	-1
1	0	Estado	Soleado	-1
2	1	Humedad	<=80.5	Si
3	1	Humedad	>80.5	No
4	0	Estado	Lluvioso	-1
5	4	Viento	Débil	Si
6	4	Viento	Fuerte	No
7	0	Estado	Nublado	si

Si Se aplica la función translate_rules() sobre la tabla “mate_rules” (Tabla 40.) del modelo Mate-Tree se tiene la tabla “mate_rules” (Tabla 48.) traducida a sus valores originales.

Tabla 48. mate_rules del modelo Mate-Tree traducida

Id	Atributo	valor
1	Estado	Soleado
1	Humedad	<= 80.5
1	Jugar	Si
2	Estado	Soleado
2	Humedad	>80.5
2	Jugar	No
3	Estado	Lluvioso
3	Viento	Débil
3	Jugar	Si
4	Estado	Lluvioso
4	Viento	Fuerte
4	Jugar	No
5	Estado	Nublado
5	Jugar	Si

5.8.3 Funcionamiento de la FDU arm_rules(). La función arm_rules() recibe como parámetro un tipo de dato VARCHAR que contiene el nombre del atributo clase y junto con un tipo de dato VARCHAR que contiene el nombre del modelo ('c45'/'mate') de clasificación. Esta función arma cada una de las reglas como su nombre lo indica y devuelve reglas del tipo *condicion1 AND condicion2.. --> Decisión*

Si Se aplica la función arm_rules() a la tabla de reglas 'c45_rules' (Tabla 48.), se obtiene como resultado las reglas que se muestran en la Figura 41..

Figura 41. Reglas de clasificación del modelo C4.5

armar_reglas

---

1 – estado = soleado AND humedad <= 80.5 → jugar = Si
2 – estado = soleado AND humedad > 80.5 → jugar = No
3 – estado = lluvioso AND viento = debil → jugar = Si
4 – estado = lluvioso AND viento = fuerte → jugar = No
5 – estado = nublado → jugar = Si

Si Se aplica la función arm_rules() a la tabla de reglas 'mate_rules' (Tabla 49), se consigue como resultado las reglas que se muestran en la Figura (42).

Figura 42. Reglas de clasificación del modelo Mate-Tree

armar_reglas

1 – estado = lluvioso AND viento = debil → jugar = Si
2 – estado = lluvioso AND viento = fuerte → jugar = No
3 – estado = nublado → jugar = Si
4 – estado = soleado AND humedad <= 80.5 → jugar = Si
5 – estado = soleado AND humedad > 80.5 → jugar = No

5.8.4 Funcionamiento de la FDU test_sliq(). La función test_sliq() recibe como parámetro un tipo de dato VARCHAR que contiene el nombre del atributo clase, y es la cargada de probar el modelo armando las reglas y consultado su coincidencia en la tabla de prueba *'kdd_pru'* y retorna las reglas del modelo junto con el número de incidencia de estas reglas en la tabla, es decir, el número de tuplas o registros que fueron correctamente clasificados.

Retomando el ejemplo del modelo construido con el algoritmo Sliq en donde se obtuvo las tablas de reglas y de nodos *'sliq_rules 1'* (Tabla 42.) y *'sliq_nodes 1'* (Tabla 43.) respectivamente. Ejecutando la función probarsliq() el resultado es el mostrado en la Figura 43.

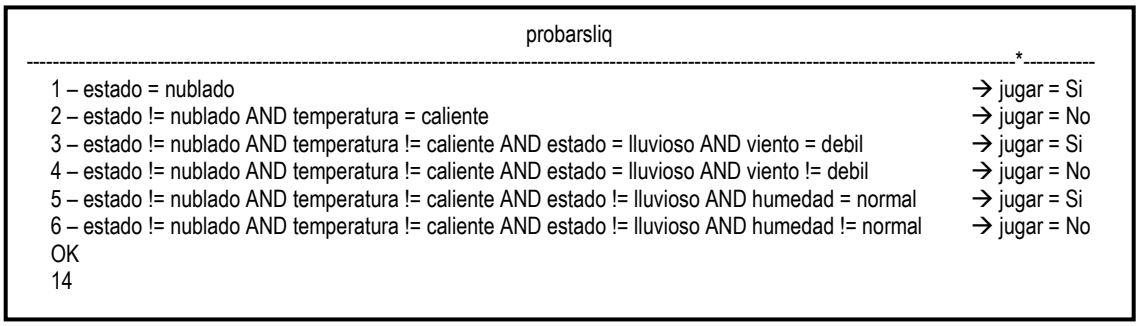
Figura 43. Resultado 1 de probarsliq

probarsliq

1 – estado = nublado → jugar = Si
2 – estado != nublado AND temperatura <= 77.5 AND estado = lluvioso AND viento = debil → jugar = Si
3 – estado != nublado AND temperatura <= 77.5 AND estado = lluvioso AND viento != debil → jugar = No
4 – estado != nublado AND temperatura <= 77.5 AND estado != lluvioso AND humedad <= 82.5 → jugar = Si
5 – estado != nublado AND temperatura <= 77.5 AND estado != lluvioso AND humedad > 82.5 → jugar = No
6 – estado != nublado AND temperatura > 77.5 → jugar = No
OK
14

Y si Se aplica la función probarsliq() a la tabla *'sliq_rules 2'* (Tabla 44) el resultado es el mostrado en la Figura 44.

Figura 44. Resultado 2 de probarsliq



6. PRUEBAS Y RESULTADOS

En el presente capítulo se describe, detalladamente, una fase fundamental en el desarrollo de la herramienta: El diseño de pruebas y análisis de los resultados obtenidos de los algoritmos implementados C4.5 [2], [4], [10]; Mate-Tree [36], [37] y Sliq [26].

6.1. DISEÑO DE PRUEBAS

Para estimar el rendimiento de los algoritmo implementados, se realizan varias pruebas usando un equipo Intel Pentium IV de 64 bits a 3,0 Ghz, memoria RAM de 2Gb, disco duro SERIAL ATA DE 160 Gb, Sistema Operativo FEDORA CORE 5

Para poder aplicar y validar las técnicas de Data Mining en general, y en particular aquellas de Clasificación que constituyen el enfoque del presente trabajo, se necesita un gran volumen de datos. Para evaluar los algoritmos de clasificación se usan repositorios especializados para éste tipo de algoritmo; se utilizaron las bases de datos JugarTenis.data (2Kb con 14 transacciones), Titanic.data (120 Kb, con 2201 transacciones) y los conjuntos de datos reales Sisben (hasta 1.466 Kb de 1000 a 15000 transacciones) y udenar (1.366 Kb con 20.328 transacciones).

La etapa de prueba de los algoritmos de clasificación tiene como objetivo:

1. Verificar que los resultados generados después de aplicados los algoritmos coincidan con los resultados que teóricamente generados teóricamente.

2. Comparar los tiempos de ejecución de cada uno de los algoritmos (únicamente proceso de Data Mining) con respecto al volumen de registros.

3. Comparar los resultados obtenidos con el algoritmo Mate-Tree con respecto a C4.5 y Sliq.

4. Medir la capacidad con la que cuenta la herramienta.

Las pruebas se realizaron sobre un conjunto de datos con un número pequeño de registros (JugarTenis, Titanic), ya que estos ejercicios son conocidos y con ello se verifica la confiabilidad de la respuesta y luego proceder con conjuntos de datos grandes, con el fin de conocer el comportamiento de los algoritmos con respecto al tiempo y hacer una evaluación mas aplicada a la realidad.

6.2. CARACTERISTICAS DE LAS BASES DE DATOS

6.2.1. Base de datos jugar tenis

Esta base de datos contiene información de 14 casos de decisión, con 5 atributos (atributos nominales).

- Número de transacciones: 14
- Número de atributos: 5
- Información de los Atributos

Tabla 49. Atributos BD JugarTenis

Nro	Nombre del Atributo	Valores del Dominio
1	Estado	Lluvioso, Nublado, Soleado
2	Temperatura	Caliente, Fresco, Templado
3	Humedad	Alta, Normal
4	Viento	Débil, Fuerte
5	Jugar ***(Clase)***	No, Si

6.2.2. Base de datos titanic

Esta base de datos contiene información de 2201 casos de decisión, con 4 atributos (atributos nominales).

- Número de transacciones: 2201
- Número de atributos: 4
- Información de los Atributos

Tabla 50. Atributos BD Titanic

Nro	Nombre del Atributo	Valores del Dominio
1	Clase	Cero, Uno, Dos, Tres

2	Edad	Adulto, Niño
3	Sexo	Hombre, Mujer
4	Sobrevivió ***(Clase)***	n, y

6.2.3. Base de datos zoologico

- Número de transacciones: 101.
- Número de Atributos: 17
- Información de los atributos

Tabla 51. Tabla Zoológico

Nro	Nombre del atributo	Tipos de valores del dominio
1	Pelo	Bpchar
2	Plumas	Bpchar
3	Huevos	Bpchar
4	Leche	Bpchar
5	Volador	Bpchar
6	Acuático	Bpchar
7	Cazador	Bpchar
8	Dientes	Bpchar
9	Esqueleto	Bpchar
10	Respira	Bpchar
11	Venenoso	Bpchar
12	Aletas	Bpchar
13	Patas	Bpchar
14	Cola	Bpchar
15	Domestico	Bpchar
16	TamaoGato	Bpchar
17	Class	Bpchar Valores enteros en el intervalo [1..7]

6.2.4. Base de datos sisben

- Número de transacciones: 15.000.
- Número de Atributos: 10
- Información de los atributos

Tabla 52. Tabla Sisben

Nro	Nombre del atributo	Tipos de valores del dominio
1	Codinstitucion	Varchar
2	Codsexo	Varchar
3	Codtipoafiliado	Varchar
4	Codrelacioncabeza	Varchar
5	**Codgrupopoblacional (Clase)**	**Varchar**
6	Codnivelsisbern	Varchar
7	Codciudad	Varchar
8	Codtipocontrato	Varchar
9	Codgrupoetnico	Varchar
10	Discapacidad	Varchar
11	Codzona	varchar
12	Fechaafiliacion	Varchar
13	fechaNacimiento	varchar

- Para las pruebas realizadas se selecciono un subconjunto de datos con los siguientes atributos:

Tabla 53. Tabla subconjunto de Sisben

Nro	Nombre del atributo	Tipos de valores del dominio
1	Codinstitucion	Varchar
2	Codsexo	Varchar
3	Codtipoafiliado	Varchar
4	Codrelacioncabeza	Varchar
5	**Codgrupopoblacional (Clase)**	**Varchar**
6	Codnivelsisbern	Varchar
7	Codciudad	Varchar
8	Codtipocontrato	Varchar
9	Codgrupoetnico	Varchar
10	Discapacidad	Varchar

6.2.5. Base de datos udenar

- Número de transacciones: 20.328.

- Número de Atributos: 26

- Información de los atributos

Tabla 54. Tabla Udenar

Nro	Nombre del atributo	Tipos de valores del dominio
1	Edad	Varchar
2	Edad_ing	Varchar
3	Fecha_ing	Varchar
4	Ingresos	varchar
5	Val_matricula	Varchar
6	Nestrato	Varchar
7	Naturaleza	Varchar
8	Jornada	Varchar
9	Calendario	Varchar
10	Sexo	Varchar
11	Ponderado	Varchar
12	Ocu_madre	Varchar
13	Mas_una_a_cargo	Varchar
14	Vive_con_familia	Varchar
15	Tiene_hermanos_u	Varchar
16	Especial	varchar
17	Padre_vive	varchar
18	Madre_vive	Varchar
19	Estado_civil	Varchar
20	tipo_residencia	Varchar
21	zona_nac	Varchar
22	Semestred	Varchar
23	cod_facultad	Varchar
24	Claseal (clase)	varchar
25	Claserend (clase)	varchar
26	Clasepromedio (clase)	Varchar

- Para las pruebas realizadas se selecciono un subconjunto de datos con los siguientes atributos:

Tabla 55. Tabla subconjunto de Udenar

Nro	Nombre del atributo	Tipos de valores del dominio
1	Edading	Varchar
2	Nestrato	Varchar
3	Naturaleza	Varchar
4	Sexo	Varchar
5	Ponderado	Varchar
6	tipo_residencia	Varchar
7	zona_nac	Varchar
8	Semestred	Varchar
9	cod_facultad	Varchar
10	**Clasepromedio**	**Varchar**

Tabla 56. Tabla Discretizacion BD Udenar

EDAD DE INGRESO	DISCRETIZACION	CANTIDAD
Menores e iguales a 18	A	9054
Mayores de 18 y menores que 22	B	7370
Mayores e iguales que 22 y menores de 26	C	2594
Mayores e iguales que 26	D	1311
NATURALEZA	**DISCRETIZACION**	**CANTIDAD**
OFICIAL	OFICIAL	13571
NO OFICIAL	PRIVADO	6757
PONDERADO	**DISCRETIZACION**	**CANTIDAD**
Menores a 30	A	43
Mayores e iguales que 30 y menores que 50	B	2689
Mayores e iguales que 50 y menores que 70	C	16601
Mayores e iguales que 70 y menores que 100	D	1623
TIPO_RESIDENCIA	**DISCRETZACION**	**CANTIDAD**
No_propia	0	5181
Propia	1	14995
Propia pagando cuotas	2	308
Anrrendando o Anticues	3	472
SEMESTRED	**DISCRETIZACION**	**CANTIDAD**
Semestre 1 hasta semestre 4	1	8859
Semestre 5 hasta semestre 8	2	3539
Semestre 9 hasta semestre 10	3	1390
Semestre (11-15) (egresados) – 20 (Demorados en egresar)	4	1383

Semestre 110 (graduados)	5	5157
FACULTAD	**DISCRETIZACION**	**CANTIDAD**
ARTES	1	2097
CIENCIAS AGRICOLAS	2	1532
DERECHO	3	1208
CIENCIAS ECONOMICAS Y ADMINISTRATIVAS	4	2424
INGENIERIA	5	2549
CIENCIAS PECUARIAS	6	1715
CIENCIAS HUMANAS	8	4009
EDUCACION	18	793
INGENIERIA AGROINDUSTRIAL	22	537
CIENCIA DE LA SALUD	32	390
CLASE PROMEDIO	**DISCRETIZACION**	**CANTIDAD**
Menor a 2.0	A	2391
Mayor o igual a 2.0 hasta 3	B	2934
Mayor o igual a 3.0 hasta 3.5	C	5166
Mayor o igual a 3.5 hasta 4.0	D	6850
Mayor o igual a 4.0 hasta 5.0	E	3615

6.3. PRUEBAS DE CONFIABILIDAD Y EFICIENCIA

Este tipo de prueba se hace con el propósito de verificar que los resultados de los algoritmos sean correctos y poder hacer una comparación entre los resultados (reglas de clasificación) de los algoritmos C4.5 y Sliq con Mate-Tree.

Es necesario aclarar que los datos utilizados para generar los modelos de clasificación pasan primero por una serie de procesos para su depuración, transformación, y filtrado.

Figura 45. Resultados de JugarTenis con C4.5

```
1. Estado = Lluvioso AND Viento = Débil          --> Jugar = Si
2. Estado = Lluvioso AND Viento = Fuerte         --> Jugar = No
3. Estado = Nublado                              --> Jugar = Si
4. Estado = Soleado AND Humedad = Alta           --> Jugar = No
5. Estado = Soleado AND Humedad = Normal         --> Jugar = Si
```

Figura 46. Resultados de JugarTenis con Mate-Tree

1. Estado = Lluvioso AND Viento = Débil	--> Jugar = Si
2. Estado = Lluvioso AND Viento = Fuerte	--> Jugar = No
3. Estado = Nublado	--> Jugar = Si
4. Estado = Soleado AND Humedad = Alta	--> Jugar = No
5. Estado = Soleado AND Humedad = Normal	--> Jugar = Si

Figura 47. Resultados de JugarTenis con Sliq

1. Estado = Nublado	--> Jugar = Si
2. Estado = Soleado AND Humedad = Alta	--> Jugar = No
3. Estado = Soleado AND Humedad = Normal	--> Jugar = Si
4. Estado = Lluvioso AND Viento = Débil	--> Jugar = Si
5. Estado = Lluvioso AND Viento = Fuerte	--> Jugar = No

Tabla 57. Porcentaje de Confiabilidad con JugarTenis

Algoritmo	Instancias	Bien Clasificadas
C4.5	14	14
Mate-Tree	14	14
Sliq	14	14

Figura 48. Resultados de Titanic con C4.5

Reglas	
1. sexo = hombre AND edad = adulto	--> sobrevivio = n
2. sexo = hombre AND edad = niño AND clase = dos	--> sobrevivio = y
3. sexo = hombre AND edad = niño AND clase = tres	--> sobrevivio = n
4. sexo = hombre AND edad = niño AND clase = uno	--> sobrevivio = y
5. sexo = mujer AND clase = cero	--> sobrevivio = y
6. sexo = mujer AND clase = dos	--> sobrevivio = y
7. sexo = mujer AND clase = tres	--> sobrevivio = n
8. sexo = mujer AND clase = uno	--> sobrevivio = y

Figura 49. Resultados de Titanic con Mate-Tree

Reglas	
1. sexo = hombre AND edad = adulto	--> sobrevivio = n
2. sexo = hombre AND edad = niño AND clase = dos	--> sobrevivio = y
3. sexo = hombre AND edad = niño AND clase = tres	--> sobrevivio = n
4. sexo = hombre AND edad = niño AND clase = uno	--> sobrevivio = y
5. sexo = mujer AND clase = cero	--> sobrevivio = y
6. sexo = mujer AND clase = dos	--> sobrevivio = y
7. sexo = mujer AND clase = tres	--> sobrevivio = n
8. sexo = mujer AND clase = uno	--> sobrevivio = y

Figura 50. Resultados de Titanic con Sliq

Reglas	
1- Class = tres	→ sobrevivio = n
2- Class = cero AND sexo = hombre	→ sobrevivio = n
3- class = cero AND sexo = mujer	→ sobrevivio = y
4- class = dos AND sexo = mujer	→ sobrevivio = y
5- class = uno AND sexo =mujer	→ sobrevivio = y
6- class = dos AND sexo = hombre AND edad =adulto	→ sobrevivio = n
7- class = dos AND sexo = hombre AND edad = nino	→ sobrevivio = y

Tabla 58. Porcentaje de Confiabilidad con titanic

Algoritmo	Bien Clasificadas	Porcentaje de Confiabilidad
C4.5	1740	79.1 %
Mate-Tree	1740	79.1 %
Sliq	1617	73.4 %

6.4. PRUEBAS DE RENDIMIENTO

Para realizar las pruebas de rendimiento de los algoritmos, se seleccionaron los conjuntos de datos Titanic, zoológico, sisben, udenar. El objetivo principal es comparar el tiempo de procesamiento de los algoritmos con respecto al número de atributos y de instancias entre los algoritmos, así como medir la capacidad de la herramienta con BD grandes.
Las pruebas de rendimiento se ejecutan teniendo en cuenta dos aspectos, primero rendimiento mirando el comportamiento de los algoritmos en la medida en que se

incrementa el número de instancias, y el segundo aspecto a evaluar es el comportamiento en la medida en que la Base de Datos crece en número de atributos.

El tiempo de procesamiento de las función mateby será medido separadamente de la función matetree ya esta funciona como un proceso de pre poda.

6.4.1. Pruebas con la Base de Datos Titanic. El conjunto de datos de titanic se duplica hasta obtener el número de registros que se requieren, esto para observar el comportamiento de los algoritmos cuando el número de registros aumenta. El tiempo expresado en las Tablas está dado en segundos.

Tabla 59. Tiempo de procesamientos de mateby con titanic

No. Instancias	Mateby
2201	0,645
4402	1,011
8804	2,422
17608	4,630
35216	10,585
70432	18,914

Tabla 60. Tiempo de procesamientos con Titanic

No. Instancias	C4,5	Mate-Tree	Sliq
2201	2,098	0,529	2,125
4402	2,431	0,600	3,163
8804	2,527	0,596	3,963
17608	2,897	0,609	6,706
35216	3,818	0,666	13,858
70432	5,609	0,713	29,415

6.4.2. Pruebas con la Base de Datos Zoológico. Es necesario probar los algoritmos con una base de datos flexible en cuanto al manejo de número de atributos y evaluar el rendimiento de cada algoritmo y aprovechar para comparar los resultados en cuanto a número de reglas y porcentaje de clasificación. Se elige esta Base ya que por su estructura permite evaluar la escalabilidad de acuerdo al número de atributos. Para ello se comienza trabajando con una copia de la base de datos original con los 4 primeros atributos más la clase y consiste en aumentar cada atributo en su orden. El tiempo expresado en las Tablas está dado en segundos.

Tabla 61. Tiempo de procesamientos de mateby con Tabla Zoológico

Att	Mateby
3	0,096
4	0,156
5	0,327
6	0,719
7	1,638
8	4,260
9	9,764
10	24,415
11	69,664
12	69,664
13	208,593
14	1958,096

Tabla 62. Tiempo de procesamientos con Tabla Zoológico

Atributos	C4,5	Mate-Tree	Sliq
3	0,116	0,193	0,491
4	0,189	0,333	0,775
5	0,262	0,451	0,975
6	0,483	0,728	1,416
7	0,661	1,457	1,881
8	1,209	1,238	2,630
9	1,418	1,284	2,911
10	2,068	1,912	3,920
11	2,290	3,433	4,375
12	3,170	6,306	5,991
13	3,640	15,197	5,665
14	3,816	40,510	7,941
15	5,035		6,367
16	5,037		5,967

6.4.3. Pruebas con la Base de Datos Sisben. Para esta prueba se seleccionaron aleatoriamente seis sub conjuntos de datos con diferente número de instancias, se aplica los algoritmos tratados en este proyecto y se mide el coste computacional de los mismos. El tiempo expresado en las Tablas está dado en segundos.

Tabla 63. Resultados con la tabla Sisben

No. Registros	C4.5	Mate-Tree	Sliq
1000	0,97	3,86	4,93
2000	0,97	4,34	6,69
5000	2,20	6,19	15,66
8000	2,18	10,87	27,73
10000	4,02	25,47	32,07
15000	7,41	45,74	46,25

Tabla 64. Comparación porcentaje de clasificación tabla Sisben

No. Registros	C45	MATE-TREE	SLIQ
1000	89,90	89,90	99,90
2000	99,80	99,80	99,95
5000	99,66	99,66	88,68
8000	99,78	99,78	89,28
10000	99,63	99,63	88,30
15000	99,73	99,73	88,93

Tabla 65. Porcentaje de clasificación en la BD Sisben

Registros	C4.5	Mate-Tree	Sliq
1000	99,60	99,60	90,80
2000	99,50	99,50	99,50
5000	99,78	99,78	88,80
8000	99,30	99,30	99,28
10000	99,25	99,25	98,62
15000	99,33	99,33	99,13

6.4.4. Pruebas con la Base de Datos Udenar. La prueba consiste seleccionar un número de instancias de forma aleatoria. Se medirá el coste de procesamiento y el porcentaje de clasificación. . El tiempo expresado en las Tablas está dado en segundos.

Tabla 66. Resultados de mateby con la tabla Udenar con aumento de atributos

No. Atributos	Mateby
4	7,281
5	20,086
6	50,344
7	155,422
8	233,133
9	1697,441

Tabla 67. Resultados con la tabla Udenar con aumento de atributos

No. Atributos	C4,5	Mate-Tree	Sliq
4	2,091	0,208	36,185
5	10,805	0,369	36,295
6	58,732	3,685	35,901
7	80,517	7,586	103,430
8	108,838	20,720	294,417
9	340,838	156,018	1214,948

Tabla 68. Comportamiento de los algoritmos con Udenar en variación de instancias

No. Instancias	C4.5	Mate-Tree	Sliq
5000	6,37	0,45	0,37
10000	13,60	0,92	6,47
15000	16,70	1,47	12,57
20328	19,20	1,67	18,67

6.5. PRUEBAS DE COMPARACIÓN DE RENDIMIENTO ENTRE LAS ARQUITECTURAS MEDIANA Y FUERTEMENTE ACOPLADA

Teniendo en cuenta que una de las partes más costosas del algoritmo Mate-Tree es la primitiva Mate by, se diseño esta prueba para medir y comparar al comportamiento de la primitiva medianamente acoplada vs fuertemente acoplada y establecer sus límites de rentabilidad en cuanto al manejo de número de atributos.

Por otro lado se busca establecer que influencia tienen las funciones entro y gain en el proceso de clasificación con el algoritmo Mate-Tree, puesto que inicialmente se considera que los tiempos de procesamiento de estas funciones es despreciable.

6.5.1 Pruebas de la primitiva Mate by con la Base de Datos Zoológico. La metodología de la prueba es la misma, iniciando con una copia de la Base de Datos Zoológico original y como en las pruebas anteriores partiendo con los cinco primero atributos y 101 instancias, y se sigue incrementando el número de atributos. El tiempo de ejecución esta dado en segundos

Tabla 69. Resultados Mate by fuertemente Vs medianamente acoplada

No. Atributos	Mateby Medianamente Acoplado	Mateby Fuertemente Acoplado
4	0,1980	0,0989
5	0,3540	0,2150
6	0,7080	0,5648
7	1,8700	0,7147
8	4,3000	0,7148
9	10,5490	3,2942
10	25,0600	7,1928
11	61,7310	16,0551
12	169,9790	40,4997
13	241.2150	122,2415

6.6. ANÁLISIS DE RESULTADOS

En esta sección se compararan y analizaran los resultados obtenidos en las pruebas de confiabilidad, eficiencia y rendimiento que se realizaron con los algoritmos C4.5, Mate-Tree y Sliq.

6.6.1 Análisis de la prueba de confiabilidad y eficiencia. Al ejecutar el algoritmo C4.5 con la BD JugarTenis se obtuvo como resultado un modelo de clasificación compuesto por 5 reglas:

1. Estado = Lluvioso AND Viento = Débil → Jugar = Si
2. Estado = Lluvioso AND Viento = Fuerte → Jugar = No
3. Estado = Nublado → Jugar = Si
4. Estado = Soleado AND Humedad = Alta → Jugar = No
5. Estado = Soleado AND Humedad = Normal → Jugar = Si

Estas clasifican en su totalidad los registros de la BD, este mismo modelo fue el resultado de ejecutar los algoritmos Mate-Tree y Sliq al conjunto de datos. Se puede comprobar fácilmente que los modelos son óptimos, por lo que se puede asegurar que los modelos generados por estos son confiables.

Por otro lado las pruebas realizadas con la BD Titanic mostraron que en bases de datos más grandes el algoritmo Sliq resulta ser menos óptimo puesto que el modelo generado tiene un porcentaje de clasificación más bajo en comparación con los modelos generados por los algoritmos C4.5 y Mate-Tree generaron modelos de clasificación con 8 reglas.

Los tres modelos generados por C4.5 y Mate-Tree tienen el mismo porcentaje de confiabilidad de clasificación 79.1% sobre el total de instancias del conjunto de datos. El porcentaje de confianza no es óptimo pero esto se debe a que en esta BD hay casos de decisión contradictorios, por tanto el modelaje se realiza sobre un conjunto de datos previamente filtrados quedando para clasificar un total de 1.740 registros depurados.

Por lo anterior es evidente que los modelos generados por los algoritmos son confiables, y de aplicarse en conjuntos de datos mas grandes en los cuales el procesamiento manual es complejo, se demuestra que los resultados obtenidos son confiables, esto se reflejaría en el porcentaje de confiabilidad que se obtenga, claro que es decisión del analista si el modelo es lo suficientemente óptimo o no para efectos de toma de dediciones.

6.6.2 Análisis de la prueba de rendimiento. En la prueba con la Base de Datos Titanic, todos los registros que se agregaron con el fin de aumentar la cardinalidad de la tabla, pertenecen a las mismas divisiones de clases, así, el número de ítems clase obtenidos con el primer conjunto de transacciones es el mismo que se obtiene al procesar el último conjunto, es decir que no hay aumento en el número de categorías clase y por ende las reglas que se obtienen son exactamente las mismas, solamente se quiso observar el comportamiento ante este fenómeno.

Los resultados de las pruebas realizadas con la tabla Titanic se detallan en la figura 51, en ella es fácil comparar el tiempo que cada algoritmo empleo creando el modelo sobre la tabla con volúmenes de datos que se incrementan en 2201 registros para cada prueba. Y en la figura 52 se observa el comportamiento de la función mateby con el mismo fenómeno.

Figura 51. Rendimiento de los algoritmos con Titanic

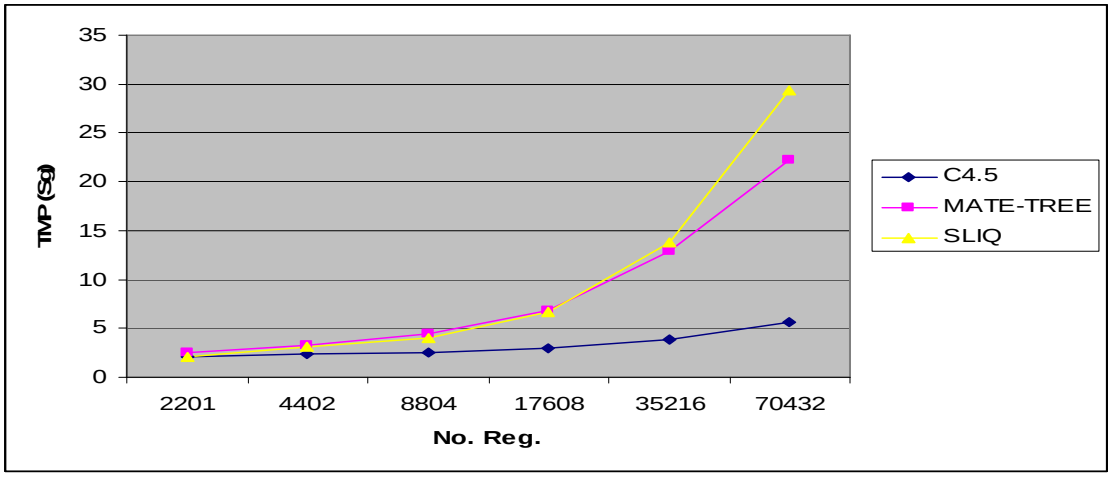
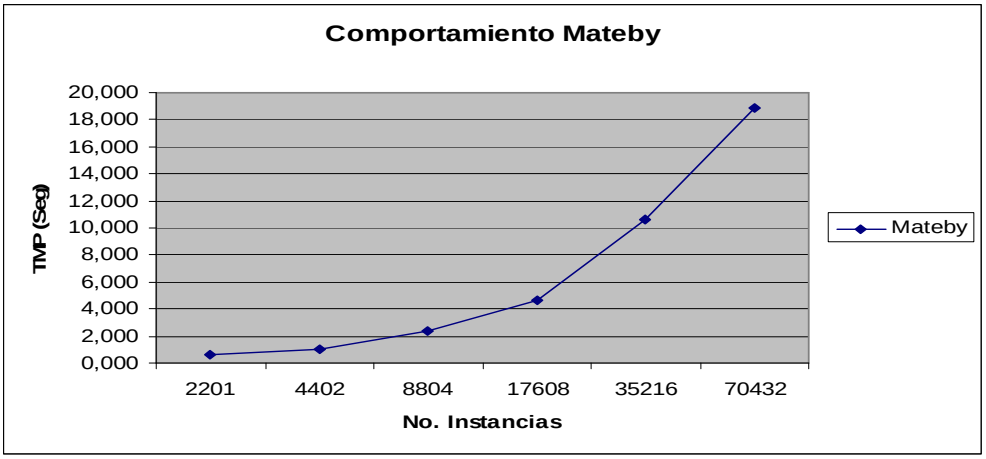


Figura 52. Comportamiento de mateby con Titanic



Es evidente que el comportamiento de la función mateby es exponencial al número de atributos lo que la hace de un coste computacional muy alto. Con una tabla de 4 atributos, de los cuales el cuarto es el atributo clase, Sliq toma mas tiempo en clasificar contrario a Mate-Tree y C5.4, (Figura 51.). Entre C4.5 y Mate-Tree el que emplea menos tiempo es C4.5 de manera que para una tabla con estas características el algoritmo resulta más eficiente.

Los resultado generales al aplicar el algoritmo Mate-Tree, muestran que los tiempos de ejecución están directamente relacionados con el número de atributos y las distintas categorías del atributo clase. El tiempo de ejecución del algoritmo crece significativamente de acuerdo al grado y cardinalidad de la tabla.

El tiempo de ejecución de los algoritmos sobre conjuntos de datos pequeños es relativamente bajo. La diferencia en tiempo de ejecución se hace más notoria en la medida en que el conjunto de datos es mayor, es decir, el tiempo de ejecución crece proporcionalmente con respecto al tamaño del conjunto de datos. La complejidad del algoritmo Mate-Tree, Sliq y de C4.5 esta directamente relacionados con: la cantidad de instancias en el conjunto de datos, el número de atributos y las categorías de los mismos.

A la hora de evaluar la escalabilidad de los algoritmos teniendo en cuenta el número de atributos, gráficamente se describe el comportamiento de la Figura 51 y Figura 52 en donde se ve el coste computacional de mateby y los algoritmos respectivamente. En la Figura 50 se demuestra nuevamente que el rendimiento de mateby es inversamente proporcional al número de atributos tratados. Y en la Figura 51 se observa que los algoritmos no siguen un patrón con respecto a las anteriores pruebas, en esta el tiempo de la función del algoritmo Mate-tree tiende a ser exponencial al número de atributos.

Figura 53. Comportamiento de mateby con Zoológico

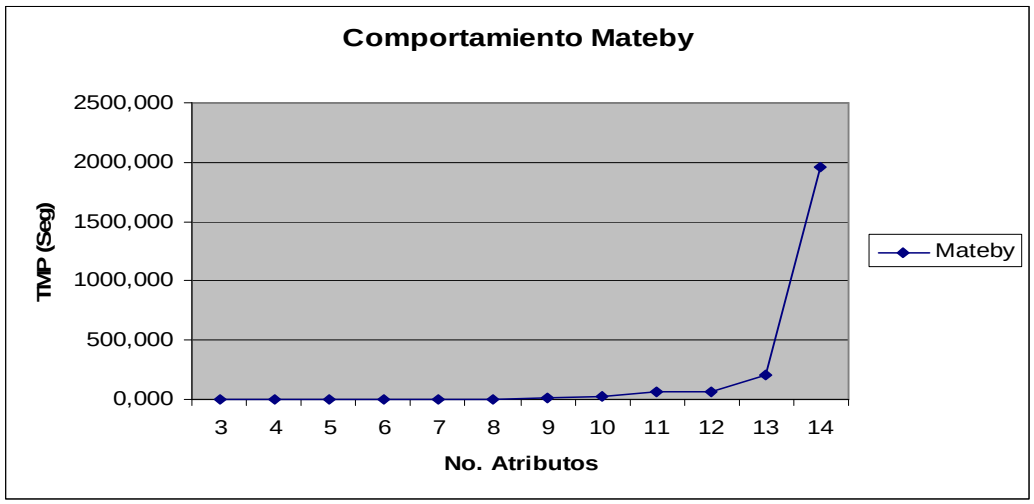
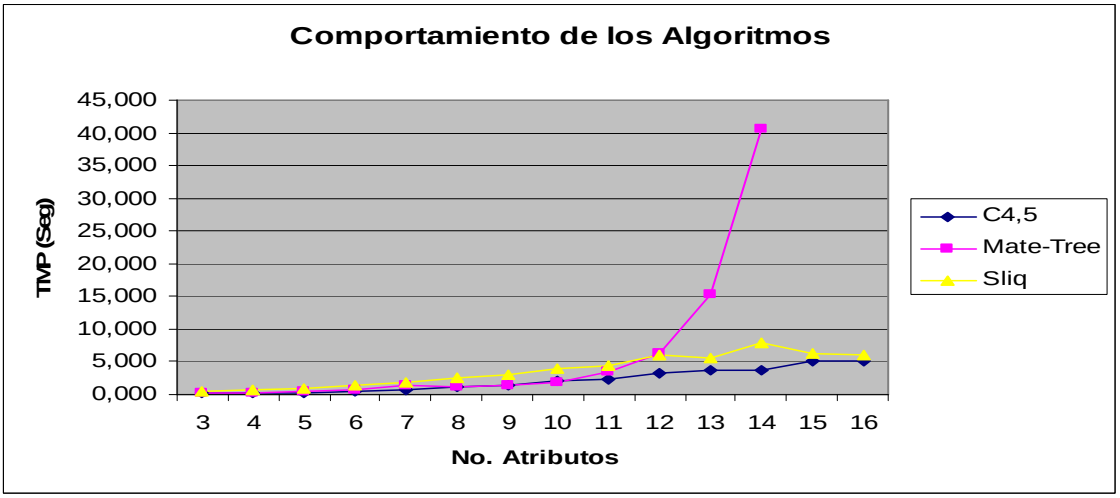
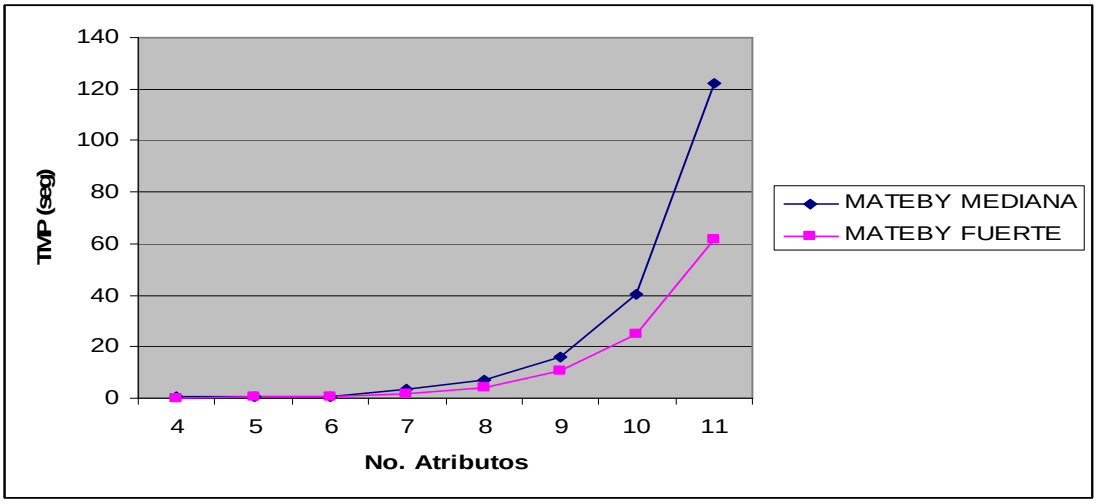


Figura 54. Gráfica de comparación de rendimiento de los algoritmos con la tabla Zoológico



Se afirma que el tiempo de procesamiento del algoritmo es directamente proporcional al número de atributos, con la Base de Datos zoológico se puede manipular la cantidad de atributos como se hizo en las pruebas anteriores, con los atributos condición empezando con cuatro atributos condición y luego incrementado un atributo cada vez, sin inconvenientes, los tres algoritmos procesaron por completo la tabla

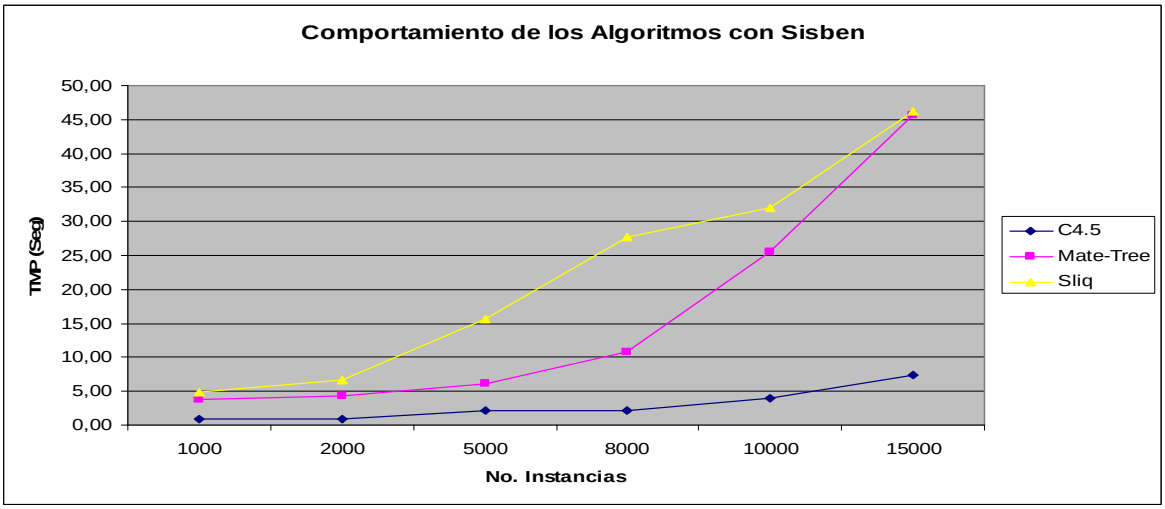
Figura 55. Gráfica de comparación de rendimiento de la primitiva Mate by con la tabla Zoológico



En la Figura 55 se observa que es la primitiva la que gasta el mayor tiempo en el proceso completo de clasificación con Mate-Tree. El tiempo que consume la primitiva crece exponencialmente cuando los atributos pasan de nueve, podría trabajarse de forma óptima hasta diez atributos, en adelante se incrementan los costos del algoritmo.

Siguiendo con las pruebas esta vez con la BD del Sisben (Figura 56), aquí se ve que el tiempo de procesamiento de los tres algoritmos va aumentando de forma considerable a medida que aumenta el número de instancias.

Figura 56. Gráfica de comparación de rendimiento tabla Sisben



El tiempo de C4.5 con esta Base de Datos es mínimo en comparación al utilizado por Mate-Tree y sobre todo por Sliq.

Siguiendo con las prueba, esta vez con la BD de Udenar las Figuras 57 y 58 en donde se observa el comportamiento de los algoritmos a medida que varia el número de atributos y el número de instancias, la primera prueba se realizo con el total de instancias del conjunto de datos y la segunda con el número de atributos (10) seleccionados previamente.

Es evidente que en la primera prueba los tres algoritmos van aumentando el coste computacional a medida que aumenta e número de atributos, en este caso Sliq es al que mas se le nota el cambio

En la Figura 58 se observa que Sliq tiene un comportamiento directamente proporcional al número de instancias, c4.5 desde un principio tiene un coste bastante alto en comparación con los otros dos y se mantiene así en todas las pruebas y Mate-tree por último mantiene un comportamiento estable y aceptable.

Figura 57. Gráfica de comparación de rendimiento tabla Udenar en variación de atributos

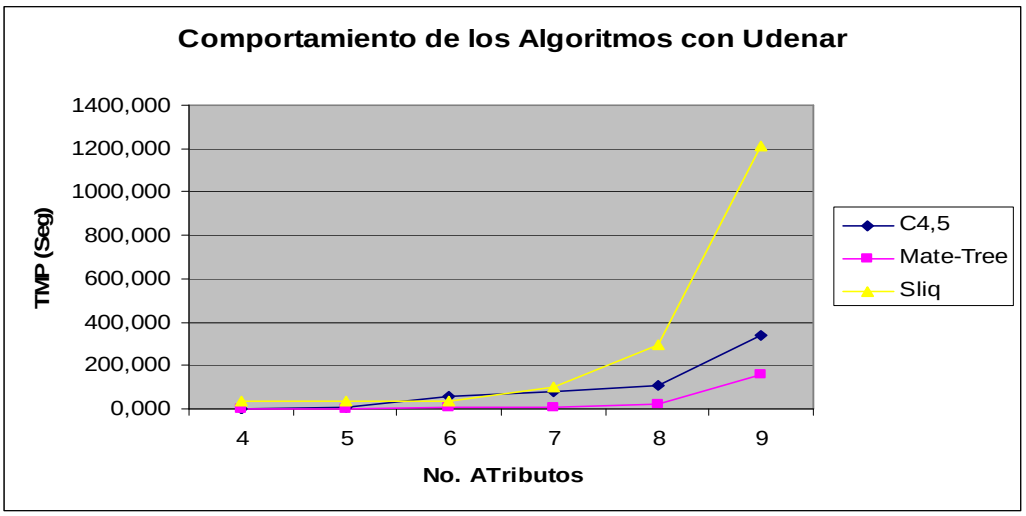
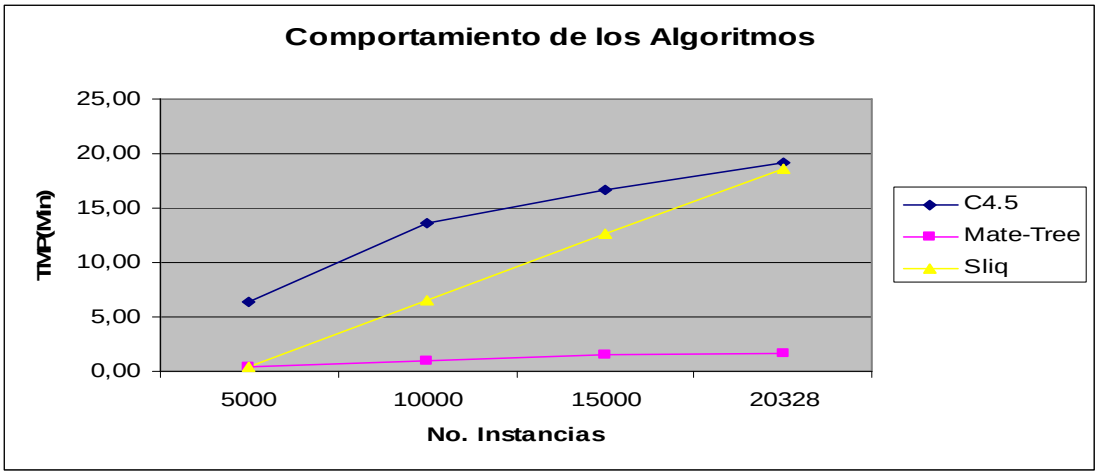
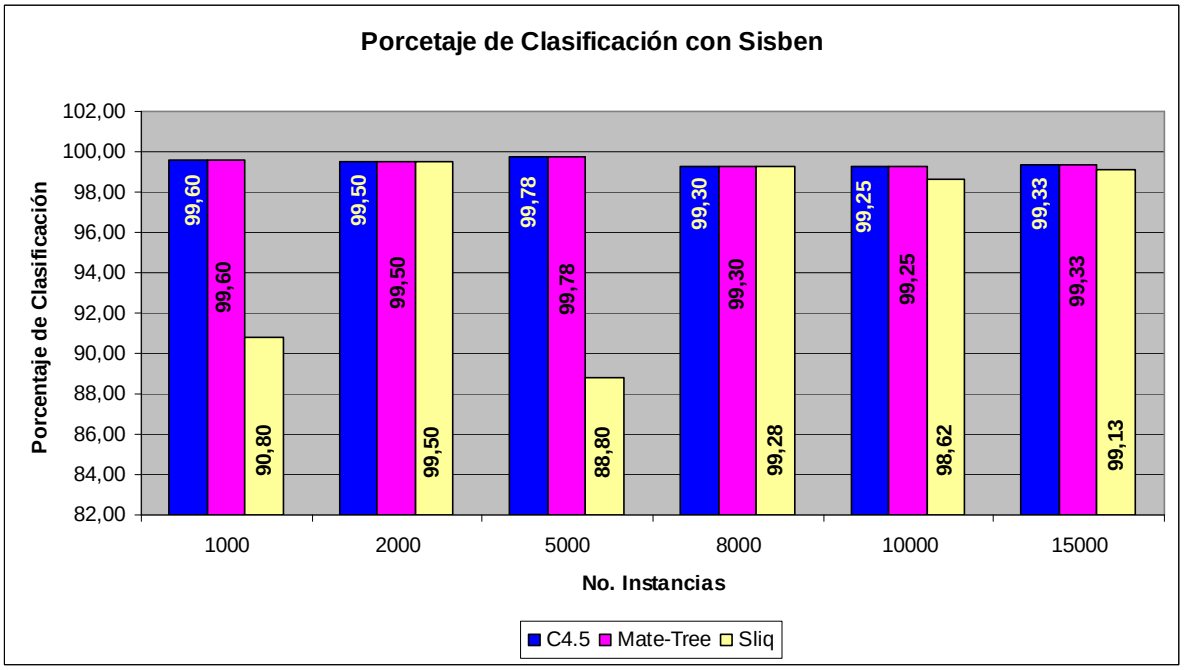


Figura 58. Gráfica de comparación de rendimiento tabla Udenar en variación de instancias



Siguiendo con la Figuras 59 que muestran el porcentaje de clasificación de los algoritmos con las pruebas realizadas en la BD del Sisben:

Figura 59. Porcentaje de clasificación en Sisben

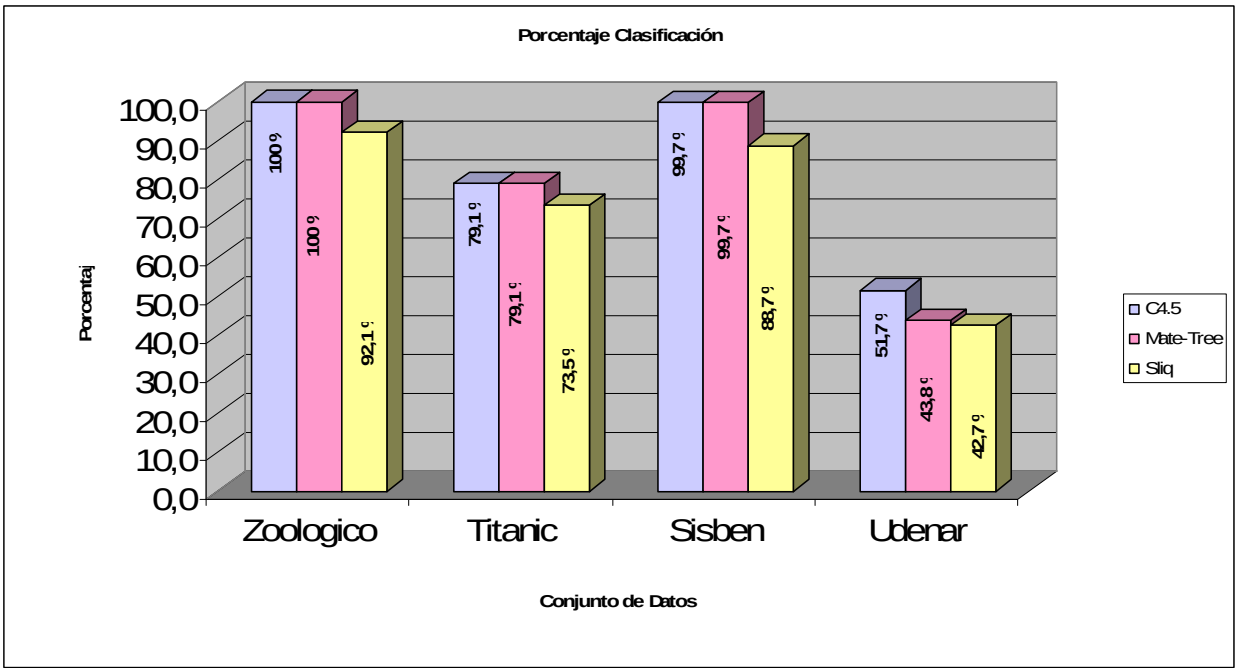


En esta gráfica resalta primero que todo que los algoritmos C4.5 y Mate-Tree tienen el mayor porcentaje de clasificación y además muy cercanos mientras Sliq varía en todas las pruebas pero en general tiene unos resultados aceptables.

Para terminar la fase de pruebas sigue la Figura 60 en donde se observa los porcentajes de Clasificación que obtienen los modelos generados por los algoritmos en los diferentes conjuntos de datos.

Esta confirma que los modelos de Mate-tree y C4.5 son los mas óptimos en la mayoría de los casos un detalle a resaltar es que el porcentaje de los tres algoritmos con la BD de Udenar es bástate bajo, esto se debe meramente al conjunto de datos tratado, es decir, la selección previa de atributos que se hizo no es aceptable o suficientes para generar modelos con este conjunto de datos.

Figura 60. Porcentaje de clasificación en los diferentes conjuntos de datos



7. CONCLUSIONES

Como resultado de este proyecto se desarrolló una herramienta de para el Descubrimiento de Reglas de Clasificación en bases de datos medianamente acoplada con el Sistema Gestor de Bases de Datos PosgreSQL, desarrollada bajo plataforma GNU,

Se analizaron herramientas para descubrimiento de conocimiento como Weka [27], enfocándonos en el método de Clasificación por medio de árboles de decisión, se estudio detalladamente el funcionamiento los algoritmos de Clasificación ***C4.5*** [2]**,** [5], [6], [11]; ***Sliq*** [26] y el nuevo algoritmo propuesto por Timaran ***Mate-Tree*** [36], [37], también se estudio la forma de implementar funciones de propósito específico dentro del servidor del sistema gestor de bases de datos Postgres. Utilizando la capacidad de Postgres de definición de funciones definidas por el usuario (FDU) a través de lenguajes procedurales.

Teniendo en cuenta las fases del proceso de Descubrimiento de Conocimiento en Bases de Datos y el funcionamiento de cada uno de los algoritmos, se desarrollaron funciones FDU para las fases de preprocesamiento de la base de datos como adicionar, eliminar, discretizar y filtrar datos, transformación que incluye la depuración y transformación de la base de datos, minería de datos con los algoritmos C4.5, Mate-Tree y Sliq; validación de los modelos y para la interpretación de los resultados. Estas FDUs se implementaron utilizando el lenguaje PL/pgSQL. Además se desarrollo una interfaz gráfica intuitiva, fácil de manejar que permite al usuario interactuar con el SGBD PostgreSQL y mirar los modelos creados de forma más comprensible, así como ejecutar sus scripts para cargar sus propias bases de datos sin necesidad de entrar al SGBD PostgresSQL, haciendo que la herramienta sea mucho mas práctica..

Se realizaron las pruebas de los algoritmos y se evaluaron los resultados de éstas. Para el plan de pruebas, se utilizaron conjuntos de datos especializados para este tipo de algoritmos disponibles en http://www.ics.uci.edu/~mlearn/databases/, así como también, bases de datos de nuestro entorno como la del Sisben y Udenar.

Después de las etapas de prueba del proyecto, se puede decir:

- Sobre el desempeño del algoritmo de clasificación Mate-Tree, se puede asegurar que es igualmente confiable que C4.5, Sliq por su parte, con resultados diferentes es igualmente confiable con la diferencia que es menos óptimo modelando los conjuntos de datos.

- El rendimiento en tiempo de procesamiento de los algoritmos C4.5, Mate-Tree y Sliq

- esta directamente relacionado con el número instancias, de atributos, categorías de los atributos.

- La confiabilidad de los modelos creados por los algoritmos es alta, cuanto mayor número de instancias se tiene, hay mayor probabilidad de que el modelo sea confiable siempre y cuando se depure la base de datos y se minimicen los datos contradictorios, de una buena técnica de preprocesamiento depende el éxito del algoritmo con sus resultados pues el índice de confiabilidad será mas alto.

- Uno de los objetivos que inicialmente se planteo fue el de evaluar y comparar el rendimiento y desempeño de cada algoritmo, pero se afirma que depende directamente de la Base de Datos que se este procesando; los algoritmos se comportan de determinada forma según las características de los registros y atributos y aunque C4.5 resulta ser menos costoso, Mate-Tree en sus tiempos de ejecución difiere de forma mínima hasta un determinado número de atributos estableciéndose entonces un limite de ocho atributos para Mate-Tree, mientras que Sliq siendo el mas costoso es el mas eficiente pues genera menos cantidad de reglas lo cual hace al modelo mas óptimo son índices de confiabilidad altos.

- El desempeño del nuevo algoritmo Mate-Tree es óptimo siempre y cuando se tenga en cuenta que el tiempo del desempeño de la primitiva mateby es exponencial frente al número de atributos que conformen el conjunto de datos.

- Por su parte Sliq tiene una forma de clasificar diferente, entonces no puede compararse, pero en cuanto a confiabilidad y escalabilidad supera a los anteriores, pero siempre resulta más costoso.

- La configuración de hardware es un punto importante a tener en cuenta a la hora de minar, cuanto mas complejas las características de la base de datos se debe pensar en una maquina lo suficientemente dotada en términos de memoria y velocidad, ya que esto influye notoriamente en el desempeño de los algoritmos en bases de datos grandes, que son los que en la práctica se busca minar.

- No se puede parametrizar el comportamiento de los algoritmos ya que este es relativo al conjunto de datos.

- Finalmente, se cuenta con una herramienta eficiente con la capacidad de extraer reglas de clasificación en una arquitectura medianamente acoplada.

- Una vez que se han descrito los resultados más relevantes que se han obtenido durante la realización de este trabajo, se sugiere una serie de puntos de partida para futuros trabajos de minería de datos:

- Para obtener resultados óptimos al minar una Base de Datos, es indispensable tener en cuenta sus características para así determinar con que algoritmo se va a crear el modelo de decisión, se sugiere:

- Implementar mas tareas de descubrimiento de conocimiento, así como procesos para preprocesamiento, filtrado y depuración del conjunto de datos.

- Optimizar el funcionamiento de la primitiva Mateby.

- Implementar procesos de pre-poda y post-poda para la optimización de los modelos generados.

- Mejorar la interfaz gráfica para brindar mayor facilidad.

- Difundir la región el uso de la minería de datos para obtener beneficios en el ámbito estratégico empresarial.

- Realizar mas pruebas con repositorios reales para analizar el comportamiento de la herramienta.

- Mate-Kdd será una herramienta de soporte a la electiva de Minería de Datos

- Finalmente, este trabajo permitió aplicar los conocimientos adquiridos en el programa de Ingeniería de Sistemas y sobretodo los recibidos en la electiva Base de Datos III, en la cual se abordó temas relacionados a las técnicas de Minería de datos y del descubrimiento de conocimiento en base de datos (KDD).

REFERENCIAS BIBLIOGRAFICAS

[1] Guerrero, M., Díaz, M., Cerquera, C., Armero S., Implantación De Primitivas Sql Para El Descubrimiento De Reglas De Asociación Y Clasificación Al Interior Del Motor Del Sistema Gestor De Bases De Datos Postgresql. San Juan de Pasto, 2005. Trabajo de grado (Ingeniero de Sistemas). Universidad de Nariño. Facultad de Ingeniería. Programa de Ingeniería de Sistema.

[2] Acuña Edgar (2003) . Notas de reconocimiento estadístico de patrones, Capítulo IX Clasificación Usando Árboles de Decisión. Departamento de Matematicas Universidad de Puerto Rico, Recinto Universitario de Mayaguez. Disponible en Internet: http://math.uprm.edu/~edgar/lec8015.html.

[3] Agrawal Rakesh, Imielinski Tomasz, Swami Arun. Database Mining: A Performance Perspective. IBM Almaden Research Center.

[4] Bogado Verónica S. Arruzazabala Mariana C. Trabajo de Investigación: Descubrimiento de Conocimiento de Conocimiento en bases de datos (KDD), 2003. Disponible en Internet: http://exa.unne.edu.ar/depar/areas/informatica/

[5] Berzal Galiano Fernando. Capítulo I: Data Mining & Knowledge Discovery in Databases (KDD). Disponible en Internet: elvex.ugr.es/berzal.html

[6] Berzal Galiano Fernando. Capítulo 5: Clasificación, Aprendizaje Supervisado. Disponible en Internet: elvex.ugr.es/berzal.html

[7] Chapman Pete (NCR), Clinton Julian (SPSS), Kerber Randy (NCR), et al. CRISP-DM 1.0 Step-by-step data mining guide. Disponible en Internet: www.crisp-dm.org

[8] Chen M., Han J., Yu P., Data Mining: An Overview from Database Perspective. Disponible en Internet: citeseer.ist.psu.edu/correct/5126

[9] DAEDALUS (2005), Fases de un Proyecto de Mineria de Datos, DAEDALUS, Data Decisión and Lenguaje, S.A. Disponible en Internet: www.DAEDALOS.org

[10] Data Mining. Disponible en Internet: www.monografias.com/trabajos-/datamining/datamining.shtml

[11] Data Mining y el Descubrimiento del Conocimiento, Disponible en Internet: sisbib.unmsm.edu.pe/bibvirtaldata/publicaciones/indata/Vol7_n2/Pdf/a13.pdf

[12] Freitas, Alex A. A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery. Postgraduate Program in Computer Science, Pontificia Universidade Catolica do Parana Rua Imaculada Conceicao, 1155. Curitiba - PR. 80215-901. Brazil.

[13] Fernández Enrique José. Trabajo de Investigación: Análisis de Clasificadores Bayesianos. Diciembre 2004.

[14] Fayyad Usama. Piatetsky-Shapiro Gregory, Smyth Padhraic. Article: From Data Mining to Knowlwdgw Discovery in Databases. Disponible en Internet: www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf

[15] Giu_ridal Giovanni, Chul Wesly w., Hanssens2 Dominique M. Mining Classification Rules from Databases with Large Number of Many-Valued attributes.

[16] Guerra Hernández Alejandro. Aprendizaje Automático: Arboles de Decisión. Universidad Veracruzana, Facultad de Física e Inteligencia Artificial Maestría en Inteligencia Artificial, Marzo 15, 2004. Disponible en Internet: www.uv.mx/aguerra/teaching/MIA/MachineLearning/clase07.pdf

[17] Hongjun Lu, Hongyan Liu. Decision Tables: Scalable Classification Exploring RDBMS Capabilites.

[18] Han, J., Kamber, M., Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, San Francisco, 2001.

[19] Isoft S.A., Alice Disponible en Internet: www.alice-soft.com.

[20] El Proceso KDD, Disponible en Internet: www.dsic.upv.es/~jorallo/-cursoDWDM/dwdm-III-2.pdf

[21] El Proceso KDD, Disponible en Internet: www.dsic.upv.es/~jorallo/cursoDDM/dwdm-III-Fayyad

[22] Data Mining & Knowledge Discovery in Databases (KDD). Disponible en Internet: elvex.ugr.es/etexts/spanish/kdd/KDD.html

[23][FONG86] Zelaine Fong; The Design and implementation of the Postgres Query Optimizer; University of California, Berkeley, Computer Science Department, Agosto 1986.

[24] Kubski, Mariana Inés (2004). Trabajo de Investigación: Minería de Datos. Licenciatura en Sistemas de Información. Universidad Nacional del Norodeste. Corrientes – Argentina. Disponible en Internet: http://exa.unne.edu.ar/depar/areas-/informatica/SistemasOperativos.

[25] Larrañaga Pedro e Inza Iñaki Capítulo 11 "Árboles de clasificación". Departamento de Ciencias de la Computación e Inteligencia Artificial Universidad del País Vasco–Euskal Herriko Unibertsitatea. Disponible en Internet: http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t11arboles.pdf.

[26] MEHTA, M., AGRAWAL, R., RISSANEN. SLIQ: A Fast Scalable Classifier for Data Mining, 1996.

[27] Molina López José Manuel, García Herrero Jesús. Trabajo de investigación: Técnicas De Análisis De Datos, Aplicaciones Prácticas Utilizando Microsoft Excel Y Weka. Universidad CARLOS III de Madrid, 2004

[28] Morales Manzanares Eduardo (2005). Descubrimiento de Conocimiento en Bases de Datos (KDD). Disponible en Internet: dns1.mor.itesm.mx/~emorales-/Cursos/KDD03/principal.html

[29] Moreno García María N. Quintales Luis A. Miguel, et al. Trabajo de investigación: "Aplicación de Técnicas de Minería de Datos en la construcción y validación de modelos predictivos y asociativos a partir de especificaciones de requisitos de software". Universidad de Salamanca, Departamento de Informática y Automática.

[30] Oren Nahum, Tali Orad. Mining Association and Inverse Association Rules in Large Database, Holon Academic Institute of Technology Department of Computer Science, January 2001.

[31] Ortiz Vega James. Implementación de un Operador de Clasificación Basado en la Técnica de Árboles de Decisión para el Manejador de Bases de Datos Postgres. Santiago de Cali, 2001. Trabajo de grado (Ingeniero de Sistemas). Universidad del Valle. Facultad de Ingeniería. Escuela de Ingeniería de Sistemas y Computación.

[32] Quiadrillon Corp., Q-Yield, Disponible en Internet: www.quadrillon.com/-qyield.shtm, 2001

[33] Ribadas Peña Francisco José. Modelos de razonamiento y Aprendizaje. Marzo 2005.

[34] SPSS, Clementine, Disponible en Internet: www.spss.com/clementine, 2001

[35] Timarán, R., Arquitecturas de Integración del Proceso de Descubrimiento de Conocimiento con Sistemas de Gestión de bases de datos: un Estado del Arte, en revista Ingeniería y Competitividad, Universidad del Valle, Volumen 3, No. 2, Cali, diciembre de 2001.

[36] Timarán, R., Nuevos operadores algebraicos y Primitivas SQL para el Descubrimiento de Conocimiento en Bases de Datos, tesis doctoral, Universidad del Valle, Diciembre 2002.

[37] Timarán, R., Millán, M., Machuca, F., New Algebraic Operators and SQL Primitives for Mining Association Rules, in proceedings of the IASTED International Conference on Neural Networks and Computational Intelligence (NCI 2003), International Association of Science and Technology for Development, Cancun, Mexico, mayo 2003.

[38] Timaran, R., Guerrero, M., Díaz, M., Cerquera, C., Armero S., Implementación de de primitivas SQL para el descubrimiento de reglas de asociación en una arquitectura fuertemente acoplado con PostgreSQL, XXXI conferencia latinoamericana de informática CLEI 2005, Cali, octubre 2005.

[39] Timarán R., Descubrimiento de Conocimiento en Bases de Datos: Una Visión general, en memorias del Primer Congreso Nacional de Investigación y Tecnología en Ingeniería de Sistemas, Universidad del Quindío, Armenia, Octubre de 2002

[40] Stefan Simkovics.Enhancement of the ANSI SQL Implementation of PostgreSQL, 1998.

[41] WIKIMEDIA 2005. Minería de Datos. A WIKIMEDIA Project, 16 NOV 2005. Disponible en Internet: es.wikipedia.org/wiki/Data_Mining

ANEXO A

INSTRUCCIONES DE INSTALACION DE PostgreSQL Y LAS FDU

Este anexo describe la instalación del código fuente de PostgreSQL, las funciones FDU y la forma correcta de usarlas.

PROCESO DE INSTALACIÓN DE POSTGRESQL

- Se ingresa como usuario root y se desplaza al directorio </usr/local/src/> donde se ubicaran las fuentes de postgresql

  $ su
  # cd /usr/local/src/
  # cp /cdrom/postgresql-7.4.3.tar.gz.

- Se desempaqueta el archivo y se ingresa en el directorio generado.

  # tar -xvfz postgresql-7.4.3.tar.gz
  # cd postgresql-7.3.4/

- Antes de iniciar la instalación se configura el código fuente de la siguiente forma:

  # ./configure

- Para compilar e instalar, se digitan las siguientes secuencias de comandos:

  # gmake
  # gmake install

- Se crea el súper-usuario postgres.

  # adduser postgres

- Se crea el directorio de datos y de generación de archivos de seguimiento, y se le asigna permisos de propietario al usuario postgres

mkdir /usr/local/pgsql/data

chown postgres:postgres /usr/local/pgsql/data

- Se inician las bases de datos básicas para el correcto funcionamiento de PostgreSQL y se ejecuta el servidor de PostgreSQL. Para esto, es necesario identificarse como usuario postgres.

  # su – postgres

  # /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data

  # /usr/local/pgsql/bin/postmaster –s –i -D /usr/local/pgsql/data

- Se crea la base de datos a utilizar, se crea el lenguaje plpgsql en la base de datos y se inicia la terminal interactiva de postgresql.

  # /usr/local/pgsql/bin/createdb <nombre_base_datos>

  # /usr/local/pgsql/bin/createlang plpgsql <nombre_base_datos>

  # /usr/local/pgsql/bin/psql <nombre_base_datos>

CREACIÓN DE LAS FUNCIONES FDUs

- Como usuario postgres se inicia la terminal interactiva de postgresql y se crean las funciones dentro de la base de datos

  # /usr/local/pgsql/bin/psql <nombre_base_datos>

  # \i /usr/local/src/postgresql-7.3.4/contrib/kdd/Functions/matekdd.sql

OBTENER Y VALIDAR EL MODELO C4.5

select * from trans_c45_mate(‘<nombre_tabla>’, ’<nombre_att_clase>’, <rango_ discretizacion>,<num_instacias_prueba>);

select * from c45(‘<nombre_att_clase>’);

select * from test_rules(‘c45’);

select * from translate_rules(<rango_discretizacion>, ’c45’, ’<nombre_att_clase>’);

select * from arm_rules(‘<nombre_att_clase>’, ’c45’);

OBTENER Y VALIDAR EL MODELO MATE-TREE

select * from trans_c45_mate(‘<nombre_tabla>’, ’<nombre_att_clase>’, <rango_ discretizacion>,<num_instacias_prueba>);

select * from matetree(‘<nombre_att_clase>’);

select * from test_rules(‘mate’);

select * from translate_rules(<rango_discretizacion>, ’mate’, ’<nombre_att_clase>’);

select * from arm_rules(‘<nombre_att_clase>’, ’mate’);

OBTENER Y VALIDAR EL MODELO SLIQ

select * from trans_sliq(‘<nombre_tabla>’, ’<nombre_att_clase>’, <num_instacias_prueba>);

select * from sliq(‘<nombre_att_clase>’);

select * from test_sliq(‘<nombre_att_clase>’);

select * from arm_rules(‘<nombre_att_clase>’,’sliq’);

ANEXO B

MANUAL DEL USUARIO

MATE-KDD: Una Herramienta Genérica para el Descubrimiento de Reglas de Clasificación Medianamente Acoplada al SGBD PostgreSQL.

TABLA DE CONTENIDO

INTRODUCCIÓN

En este manual se encontrará con la información necesaria acerca MATE-KDD: Una Herramienta Genérica para el Descubrimiento de Reglas de Clasificación Medianamente Acoplada al SGBD PostgreSQL, instalación en su sistema.

DE QUÉ TRATA EL PROGRAMA

El programa MATE-KDD, es una herramienta creada para contribuir con el desarrollo de actividades de minería de datos, es decir que con esta herramienta se pretende poner a disposición de los usuarios tres algoritmos para obtener reglas de clasificación a partir de una base de datos diseñada en PostgresSQL, con lo cual puede comparar resultados y rendimiento de cada uno de los algoritmos.

Esperamos sea de gran utilidad.

aleydacabrera@yahoo.es aleyda@mail.udenar.edu.co

claudiami83@yahoo.es claudiami83@gmail.com

1. INSTALACIÓN DEL PROGRAMA

Para la instalación de MATE-TREE se debe tener en cuenta los pasos que se describen a continuación.

- Se ingresa como usuario root y se desplaza al directorio </usr/local/src/> donde se ubicaran las fuentes de postgresql

  $ su
  # cd /usr/local/src/
  # cp /cdrom/postgresql-KDD.tar.gz.

- Se desempaqueta el archivo y se ingresa en el directorio generado.

  # tar xvfz postgresql-KDD.tar.gz
  # cd postgresql-7.3.4

- Antes de iniciar la instalación se configura el código fuente de la siguiente forma:

  # ./configure

- Para compilar e instalar, se digitan las siguientes secuencias de comandos:

  # gmake
  # gmake install

- Se crea el súper-usuario postgres.

  # adduser postgres

- Se crea el directorio de datos y de generación de archivos de seguimiento, y se le asigna permisos de propietario al usuario postgres.

  # mkdir /usr/local/pgsql/data

  # chown postgres:postgres /usr/local/pgsql/data

- Se inician las bases de datos básicas para el correcto funcionamiento de PostgreSQL y se ejecuta el servidor de PostgreSQL. Para esto, es necesario identificarse como usuario postgres.

  # su – postgres
  $ /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
  $ /usr/local/pgsql/bin/postmaster –i –s -D /usr/local/pgsql/data >logfile 2>&1 &

- Una vez se tiene instalado y configurado nuestro Postgres, procedemos a instalar La herramienta en el directorio **<</opt/>>**

  # cd /opt/
  # cp /cdrom/Matekdd.tar.gz.
- Se desempaqueta el archivo y se ingresa en el directorio generado.

  # tar –xvfz Matekdd.tar.gz.

- Antes de ejecutar por primera vez la herramienta se debe cambiar los permisos del directorio **/models** y su contenido.

  # chmod –777 /opt/Matekdd/models

Para ejecutar la herramienta se ingresa al directorio **<<.../dist>>**

  # cd /opt/Matekdd/dist
  # java –jar Matekdd.jar

2. CONSIDERACIONES DEL SISTEMA OPERATIVO

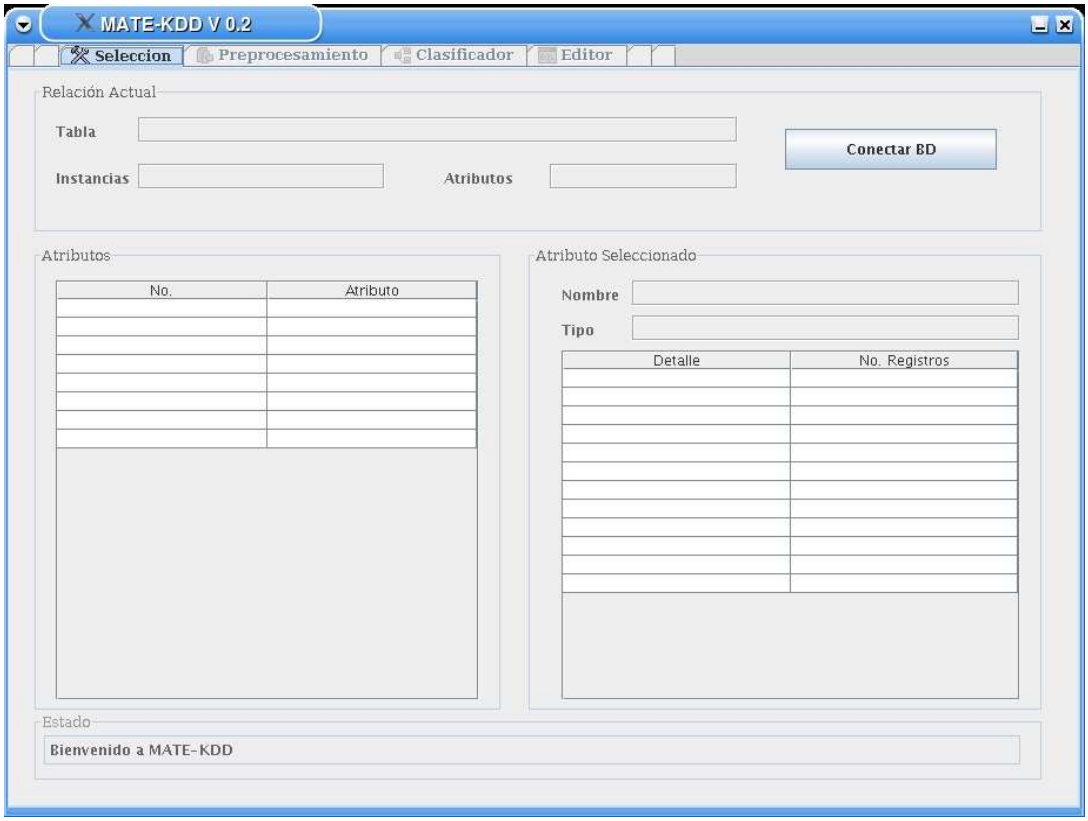
El proyecto se desarrollo bajo filosofía de Software Libre, la herramienta funciona sobre una plataforma Linux, la distribución que se utilizo fue Slakware 10.

Se debe tener instalado Postgresql-7.3.4 que se encuentra incluida en el CD de instalación. En caso de preferir utilizar otra versión de Postgresql es necesario copiar el directorio "kdd" al directorio "contrib" de la otra versión instalada de Postgresql.

3. AMBIENTE DEL PROGRAMA

Esta herramienta está diseñada para facilitar el proceso de la Minería de datos mediante algoritmos de clasificación como son SLIQ, C4.5 y MATE-TREE.

El diseño, es bastante amigable y predecible para una persona conocedora del proceso de Minería de Datos, ya que la aplicación organiza cada uno de los procesos que se siguen secuencialmente para minar una base de datos, para ello encontramos en una ventana diferente la aplicación de cada fase (selección de la BD, preprocesamiento, modelación con determinado algoritmo y resultados), interfaces fáciles de manejar y perfectamente validadas de tal forma que se puede llevar a cabo el proceso de minería de principio a fin sin dificultad.



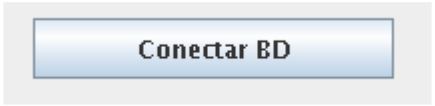
3.1 PESTAÑAS DEL PROGRAMA

En esta parte de la pantalla se visualiza las funciones de la aplicación: Selección, Preprocesamiento, Clasificador y Editor. Cada pestaña permite que naveguemos por la aplicación y en cada item encontramos los pasos principales para minar, empezando por seleccionar la tabla que contiene los datos para procesar, manipulación de la misma, según criterio del analista con elfin de obtener resultados validos, aplicación del modelo para finalmente visualizar las reglas obtenidas y el árbol del modelo. Se implemento una opción importante que permite la interacción directa del analista con el SGBD PostgreSQL, este es el editor.

3.1.1 Opción de Selección

En la opción de Selección nos conectamos con el SGBD para ubicar la tabla que vamos a minar.

3.1.1.1 Conexión a la BD



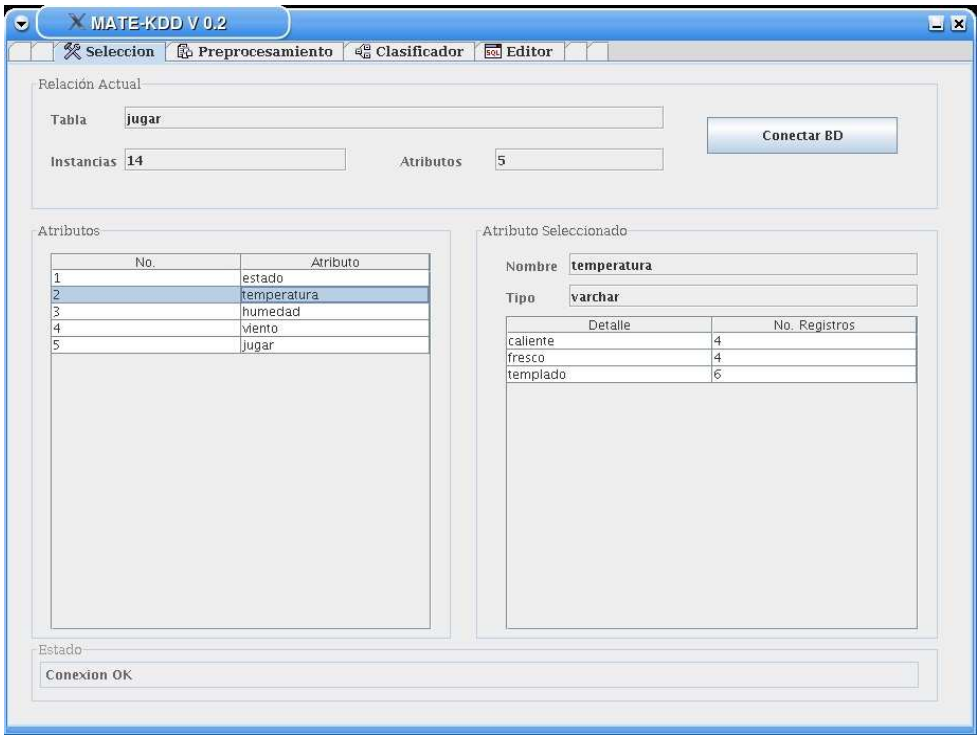
Con este Botón es posible desplegar la lista en la que se elege la Base de Datos que se encuentre Creada en Postgres a la cual accedemos con un nombre de Usuario y una contraseña, el nombre de la base de datos y Host, una vez seguidos estos pasos en el recuadro blanco Rotulado como "Seleccionar Tabla" se visualizan la tablas que tiene creada la base de datos y seleccionamos la deseada para trabajar. También permite borrar tablas que no queramos o ya no necesitemos de nuestra Base de Datos.

Simplemente haciendo clic sobre el botón *"Aceptar"* obtendremos una descripción detallada de la tabla, sus atributos, tipos de datos. Se tiene dos secciones en la ventana principal en la que se visualiza la información de los datos con el fin de conocer sus características para empezar a trabajar y determinar si vamos a aplicar algún cambio en la opción de preprocesamiento.

En el área de trabajo es posible diferenciar la siguiente información:

Relación Actual	Muestra el nombre de la tabla, así como el número de atributos e instancias que la componen. También esta los botones para abrir o guardar una tabla.
Atributos	Aquí se muestran los nombres de los atributos de la tabla.
Atributo Seleccionado	Al seleccionar con un clic un atributo de *"Atributos"* en esta parte se muestra las características de ese atributo.

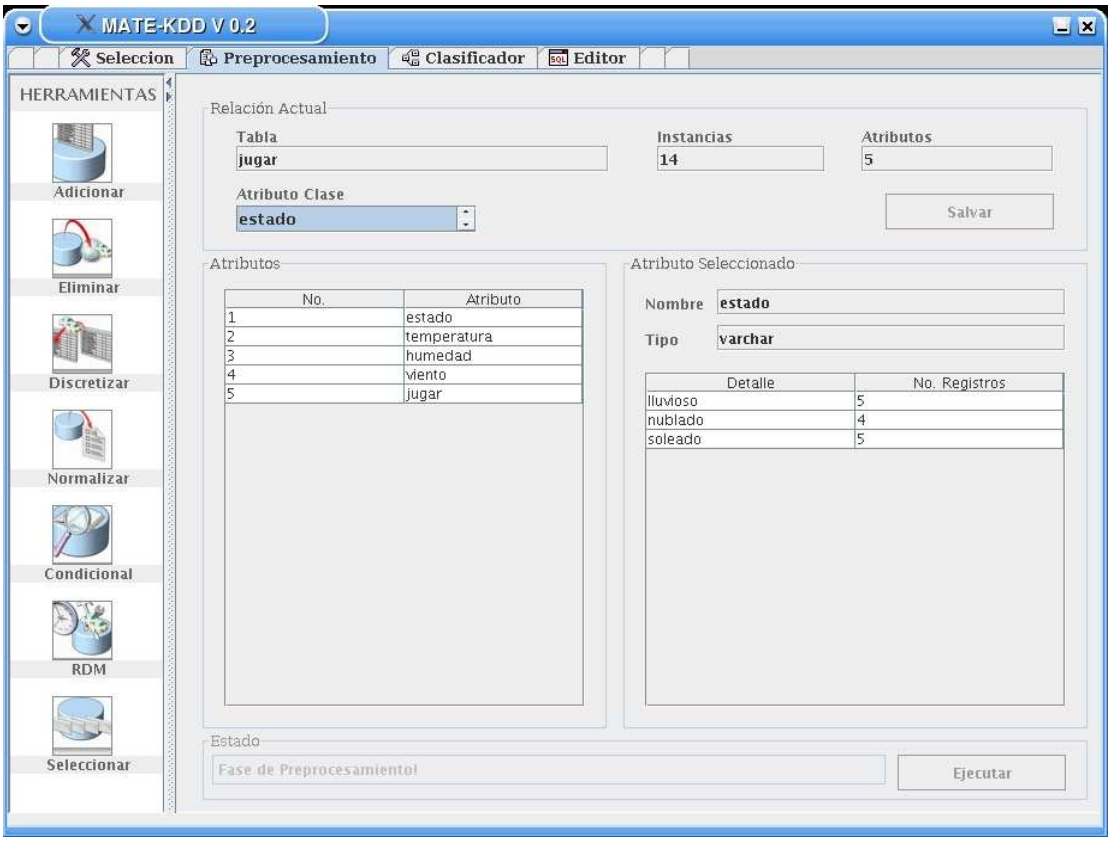
Se observar que en la ventana se detalla la tabla seleccionada, sus atributos, tipos de datos, cantidades.



3.1.2 Opción Preprocesamiento

Mate-kdd permite aplicar en la etapa de preprocesamiento algunas técnicas de depuración y discretización que son indispensables para poder procesar los datos y de esta forma obtener unos patrones confiables.

Se cuenta con algunos filtros que permiten manipular la tabla a criterio del analista para depurar datos ruidosos, Discretizar atributos numéricos que es muy importante según el algoritmo que deseemos aplicar, esto tonel fin de realizar transformaciones a partir de la tabla original de la forma en que el analista vea pertinente, una vez se aplica cualquiera de los filtros se crea una tabla con la nueva configuración a la que podemos dar un nombre. Es en esta fase en donde se debe seleccionar el atributo clase.



Adicionar: Esta herramienta permite adicionar un atributo a la tabla, éste puede ser el resultado de una operación matemática entre dos atributos de la misma o un atributo con un número.

Normalizar: Esta herramienta permite normalizar los datos numéricos que se encuentran en base 10 a rangos entre 0 y 1.

Discretizar: Con esta Herramienta podemos Discretizar un conjunto de atributos numéricos en rangos de datos.

Condicional: mediante esta herramienta podemos filtrar la tabla con una condición booleana (***or*** o ***and***).

RDM: Nos permite seleccionar ramdomicamente un número de datos de la tabla original, el analista especifica cuantos datos quiere

Cuando se tiene seleccionada la tabla para minar podemos entonces aplicar alguno de los filtros antes mencionados dando clic en el botón "Ejecutar".

3.1.2.1 Adicionar

En esta ventana podemos dar un nombre al atributo que vamos a adicionar, seleccionamos el primer atributo la operación matemática que aplicaremos y el segundo atributo o un valor para efectuar el proceso. Al dar clic en el botón "Aceptar", se construye la instrucción SQL para su posterior ejecución.

3.1.2.2 Eliminar

En la ventana de Eliminar atributo podemos seleccionar el atributo que deseamos eliminar de la tabla Al dar clic en el botón "Aceptar", se construye la instrucción SQL para su posterior ejecución.

3.1.2.3 Discretizar

Para Discretizar una tabla o un atributo debemos tener en cuenta que los atributos deben ser numéricos para discretizarse; introducimos el número de rangos que deseamos para formar los grupos de datos que no puede ser mayor que 10, seleccionamos "Discretizar Tabla" o "Discretizar Atributo" según necesitemos y Se aplica el proceso.

3.1.2.4 Normalizar

Para Normalizar una tabla o un atributo procedemos igual que para Discretizar, seleccionamos si el proceso se va a realizar sobre la tabla o sobre un atributo para seleccionar.

3.1.2.5 Condicional

Esta opción nos permite hacer un filtro muy especial, podemos condicionar un atributo mediante una condición lógica.

3.1.2.6 Seleccionar

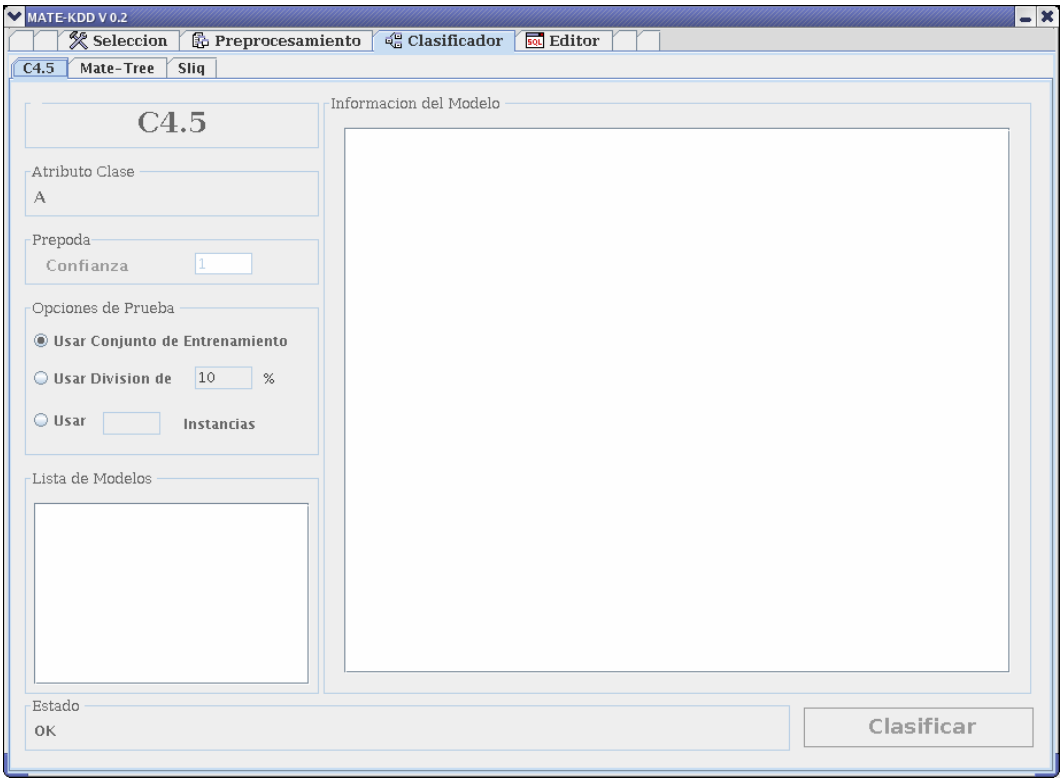
Esta opción permite escoger los atributos que el analista desee de una tabla para clasificar solo con ellos.

3.1.3 Opción Clasificador

Aquí se tiene la posibilidad de seleccionar el algoritmo C4.5, Mate-Tree o Sliq para aplicarlo.

3.1.3.1 C4.5

Al seleccionar la pestaña C4.5 aparece esta pantalla en donde se presenta el proceso de clasificación utilizando el algoritmo C4.5.



Cuando nos disponemos a modelar con cualquiera de los tres algoritmos implementados podemos configurar el los conjuntos para entrenamiento y prueba del modelo:

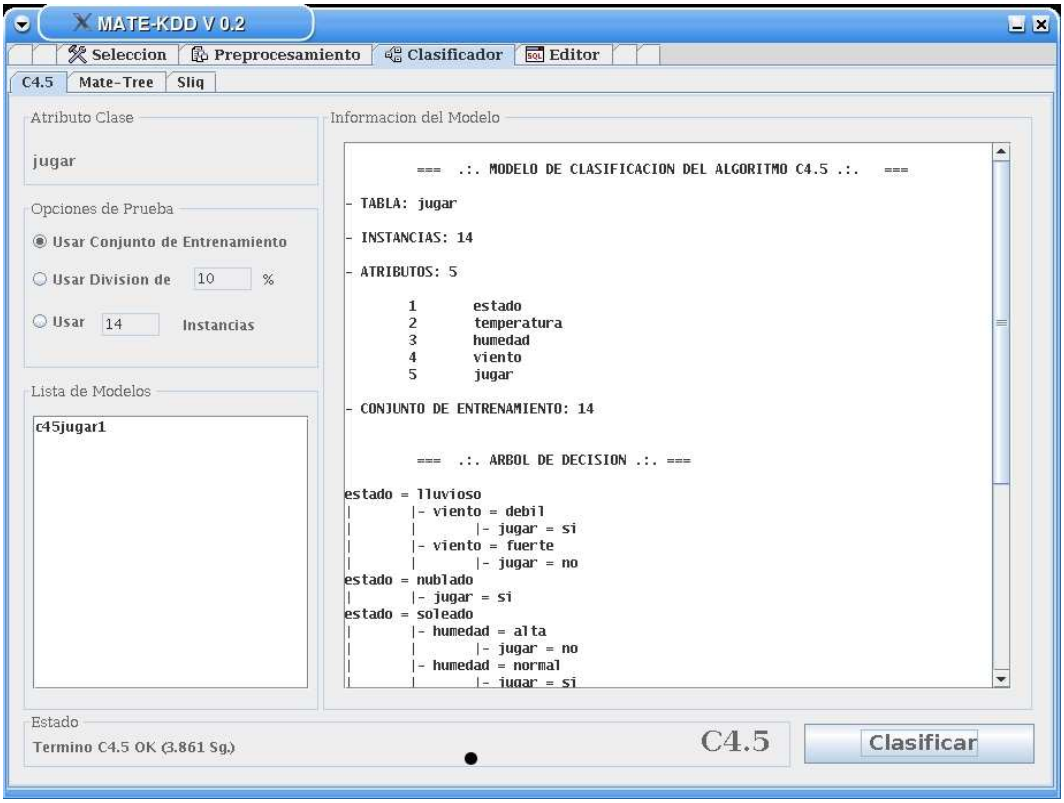
- *Usar conjunto de entrenamiento:* De esta forma se utiliza todo el conjunto de datos para la construcción y para la prueba del modelo.

- *Usar Division de X %:* De esta forma se divide el conjunto de datos para entrenamiento y prueba.

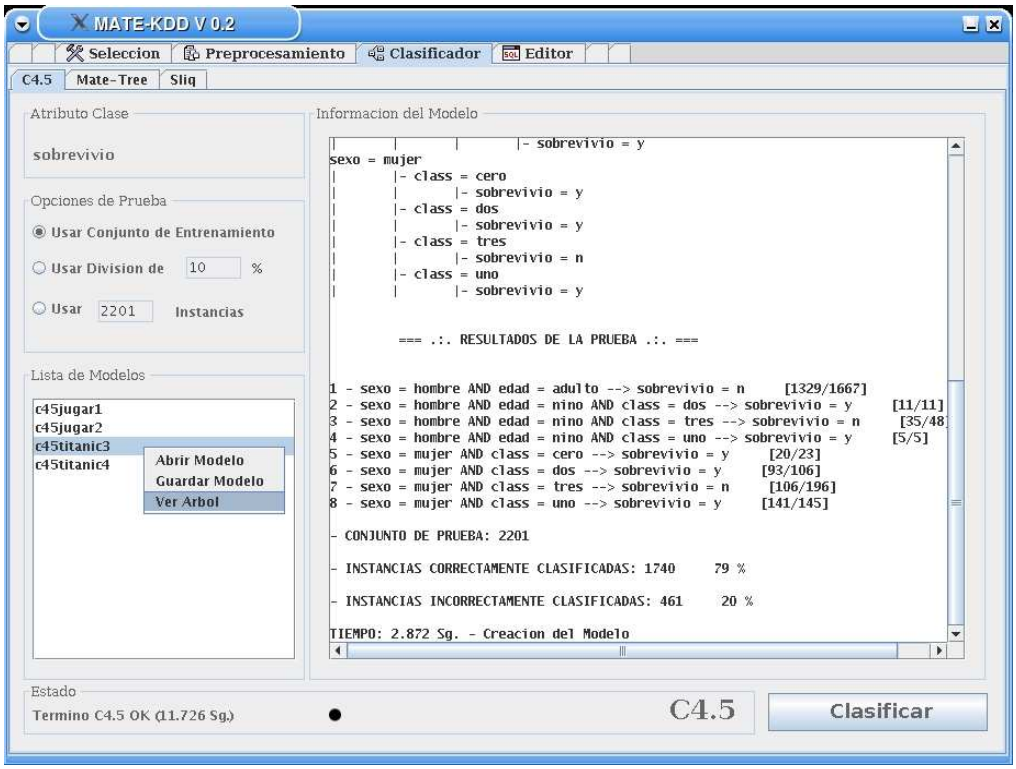
- *Usar X Instancias:* De esta forma se define cuantos registros van a componer el conjunto de datos para la prueba del modelo.

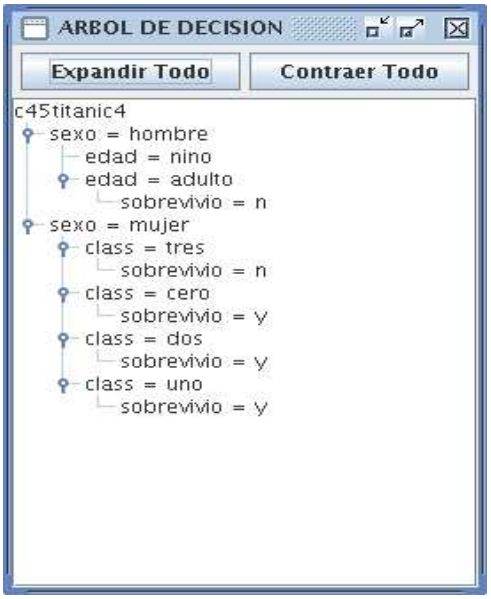
Una vez se haya configurado la opción de prueba, se inicia el modelaje con clic sobre el botón "Clasificar".

El modelo creado se visualiza en el recuadro blanco al lado derecho de la pantalla, y en el recuadro pequeño al lado izquierdo de la ventana se muestra en detalle el modelo.



Podemos guardar el modelo dando clic derecho sobre el nombre que aparece por defecto en el recuadro "Lista de modelos".

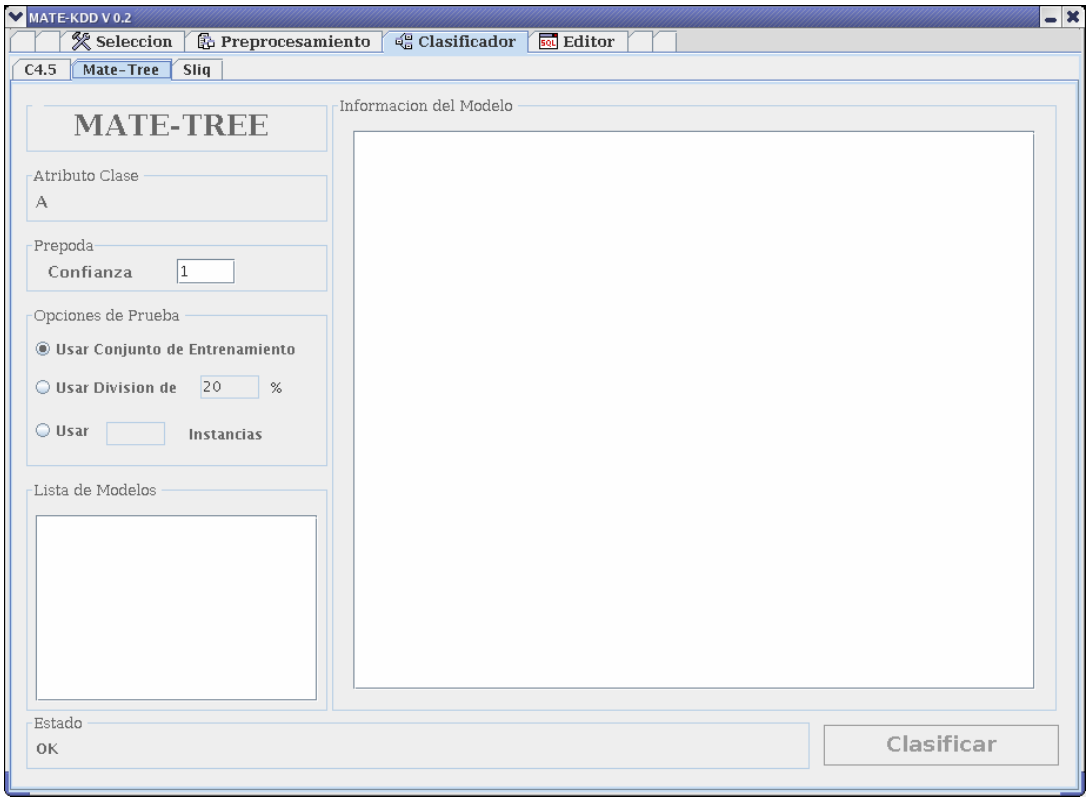

MATE-KDD V 0.2
Seleccion
Preprocesamiento
Clasificador
Editor
C4.5
Mate-Tree
Sliq
Atributo Clase
sobrevivio
Opciones de Prueba
Usar Conjunto de Entrenamiento
Usar Division de 10 %
Usar 2201 Instancias
Lista de Modelos
c45jugar1
c45jugar2
c45titanic3
c45titanic4
Abrir Modelo
Guardar Modelo
Ver Arbol
Informacion del Modelo
|- sobrevivio = y
sexo = mujer
|- class = cero
|- sobrevivio = y
|- class = dos
|- sobrevivio = y
|- class = tres
|- sobrevivio = n
|- class = uno
|- sobrevivio = y
=== .:. RESULTADOS DE LA PRUEBA .:. ===
1 - sexo = hombre AND edad = adulto --> sobrevivio = n [1329/1667]
2 - sexo = hombre AND edad = nino AND class = dos --> sobrevivio = y [11/11]
3 - sexo = hombre AND edad = nino AND class = tres --> sobrevivio = n [35/48]
4 - sexo = hombre AND edad = nino AND class = uno --> sobrevivio = y [5/5]
5 - sexo = mujer AND class = cero --> sobrevivio = y [20/23]
6 - sexo = mujer AND class = dos --> sobrevivio = y [93/106]
7 - sexo = mujer AND class = tres --> sobrevivio = n [106/196]
8 - sexo = mujer AND class = uno --> sobrevivio = y [141/145]
- CONJUNTO DE PRUEBA: 2201
- INSTANCIAS CORRECTAMENTE CLASIFICADAS: 1740 79 %
- INSTANCIAS INCORRECTAMENTE CLASIFICADAS: 461 20 %
TIEMPO: 2.872 Sg. - Creacion del Modelo
Estado
Termino C4.5 OK (11.726 Sg.)
C4.5
Clasificar


ARBOL DE DECISION
Expandir Todo
Contraer Todo
c45titanic4
sexo = hombre
edad = nino
edad = adulto
sobrevivio = n
sexo = mujer
class = tres
sobrevivio = n
class = cero
sobrevivio = y
class = dos
sobrevivio = y
class = uno
sobrevivio = y

La opción "Abrir Modelo" permite escoger y visualizar en la pantalla un modelo que ya se haya creado y guardado anteriormente.

3.1.3.2 Mate-Tree

En la pestaña Mate-Tree se tiene que configurar la tabla para clasificar los datos para entrenamiento y prueba del modelo. Para aplicar el modelo los datos deben estar discretizados en caso de que hallan atributos numéricos, con el botón *"clasificar"* Se aplica el algoritmo.

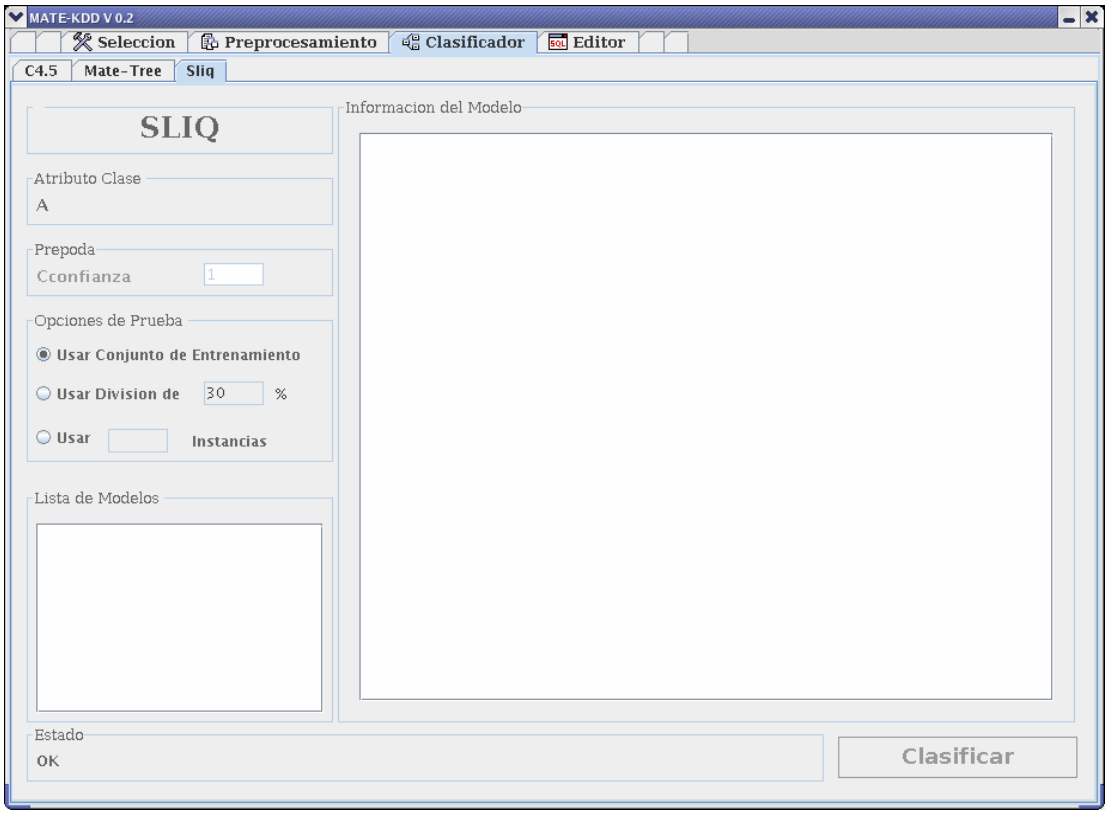


Podemos guardar el modelo, visualizar el árbol y si ya se tiene modelos guardados anteriormente podemos abrirlos y revisar.

3.1.3.3 **Sliq**

Al seleccionar la pestaña Sliq nos encontramos con la pantalla en donde se presentan los diferentes procesos que permiten el proceso clasificación utilizando el algoritmo Sliq.

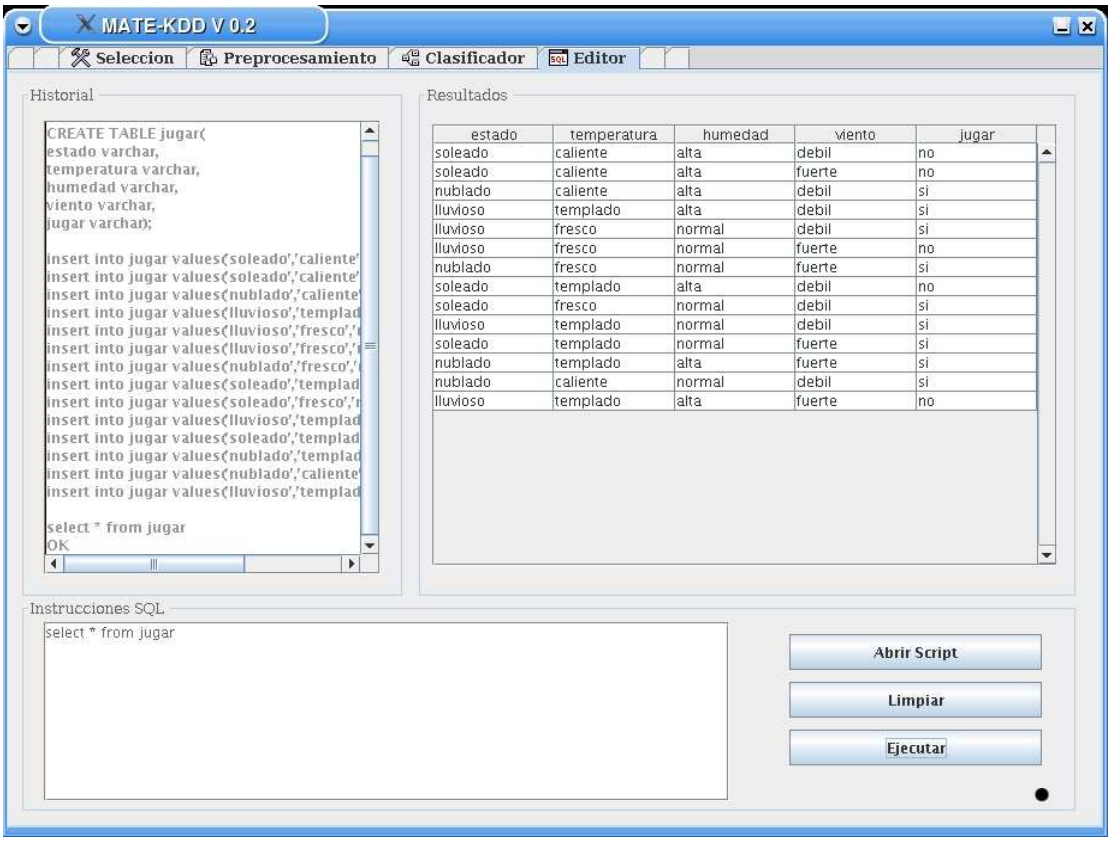
Para este algoritmo no es necesario realizar una discretización previa de los atributos numéricos a menos que el analista lo considere necesario, ya que Sliq trabaja con atributos numéricos, Y se da clic en el botón "Clasificar" para inicial el modelaje.



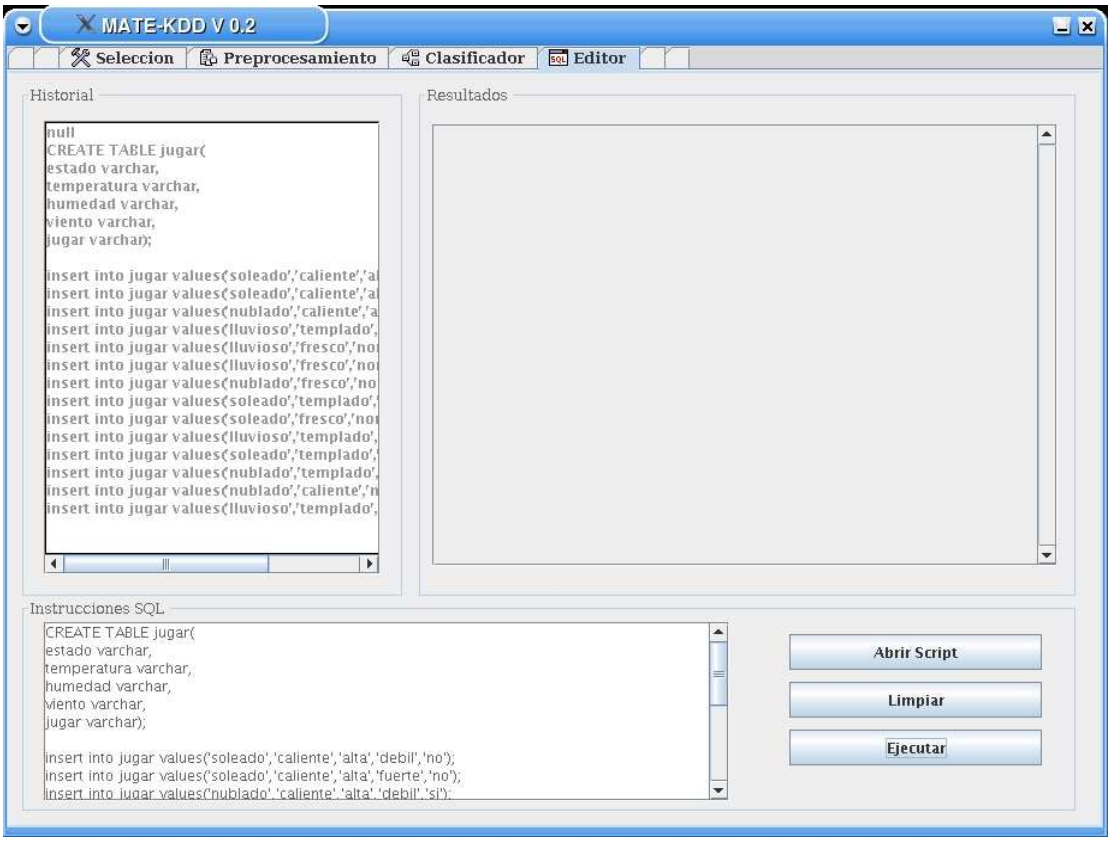
Los modelos creados los podemos guardar, abrir y mirar el árbol de cada modelo, igual que en los anteriores algoritmos, esto con el fin de visualizar los diferentes modelos resultado de aplicar cada algoritmo a una tabla en particular.

3.1.4 Editor

El editor es una interfaz que nos permite ejecutar sentencias SQL en el SGBD Postgresql, y tiene un àrea en la que visualizamos el resultado de la instrucción si es una consulta.



Otra opción muy útil en el editor es la de abrir y ejecutar Scripts, así que no se tiene que ir a la Base de datos Postgres pues desde la aplicación podemos crear tablas y cargar datos para comenzar a utilizar la aplicaron.



Como se observo a lo largo de este manual, la aplicación es fácil de utilizar debido a su interfaz amigable, podemos seguir cada paso del proceso de Minería en detalle y hacer una seguimiento hasta el final, además que con el editor podemos interactuar con el SGBD desde la aplicación sin tener que ir a Postgres si se tiene los scripts de las tablas que queremos minar.