MODELO DE DETECCIÓN DE FRAUDE BASADO EN EL DESCUBRIMIENTO SIMBÓLICO DE REGLAS DE CLASIFICACIÓN EXTRAÍDAS DE UNA RED NEURONAL

WILFREDY SANTAMARÍA RUÍZ

Cód. 299742

"Tesis de grado para optar el título de Magíster en
Ingeniería de Sistemas y Computación"

Director:

Ph.D ELIZABETH LEÓN GUZMÁN

UNIVERSIDAD NACIONAL DE COLOMBIA

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS E INDUSTRIAL

Bogotá D.C., 2010

Agradecimientos

"Este trabajo de grado se lo dedico especialmente a mis padres, mis hermanos, mi sobrino camilo y a todas aquellas personas que creyeron en mí y que me apoyaron para culminar esta carrera. También se lo dedico a la Universidad y a todos los profesores que personal y profesionalmente hicieron de mí una persona preparada y con disposición a asumir retos en el transcurso de mi vida profesional."

El autor expresa sus agradecimientos a:

PhD. ELIZABETH LEÓN GUZMÁN, Profesora de la Universidad Nacional y Director de este proyecto, por su valiosa orientación y apoyo incondicional.

PhD. JONATAN GOMEZ, Profesor de la Universidad Nacional, por su valiosa orientación en el transcurso de la maestría.

MsC. LUIS ROBERTO OJEDA, Profesor de la Universidad Nacional, por su apoyo y colaboración.

Índice general

Índice de figuras

Índice de Tablas

Abstract

This project presents a fraud detection model using data mining techniques such as neural networks and extracting symbolic rules from training artificial neural networks. The proposal of this model arises from the interest of designing and developing a fraud detection tool, in order to help business experts to more easily examine and verify the results for decision making. Related techniques are chosen above, given its good performance of classification and robustness to noise. The proposed model was tested on a set of data from a Colombian organization for sending and remittance payments, in order to identify patterns associated with fraud detection. Similarly the results of the techniques used in the model were compared with other mining techniques such as decision trees, for that purpose a software prototype was developed to test the model, which was integrated into RapidMiner tool that can be used as a tool for academic software.

KEY WORDS: fraud detection, data mining techniques, neural network, extracting symbolic rules, experts business.

Resumen

El presente proyecto presenta un modelo de detección de fraude empleando técnicas de minería de datos tales como redes neuronales y extracción simbólica de reglas de clasificación a partir de la red neuronal entrenada. La propuesta de este modelo surge del interés de diseñar y desarrollar una herramienta de detección de fraude, con el fin de ayudar a los expertos del negocio a examinar y verificar más fácilmente los resultados obtenidos para apoyar la toma de decisiones. Se eligen las técnicas relacionadas anteriormente, dado su buen desempeño de clasificación y robustez al ruido. El modelo propuesto se probo sobre un conjunto de datos de una organización colombiana para el envío y pago de remesas, con el fin de identificar patrones ligados a la detección de fraude. De igual forma los resultados de las técnicas utilizadas en el modelo, se compararon con otras técnicas de minera como los árboles de decisión, para ello un prototipo de software se desarrollo para probar el modelo, el cual fue integrado a la herramienta de RapidMiner , que puede ser usado como una herramienta de software académico.

PALABRAS CLAVE: Detección de fraude, técnicas de minería de datos, redes neuronales, descubrimiento simbólico de reglas, expertos.

Capítulo 1

Introducción

El propósito de esta tesis es proporcionar un prototipo de modelo de detección de fraude empleando minería de datos. Específicamente, se plantea el uso de la técnica de extracción simbólica de reglas de clasificación provenientes de una red neuronal "MultiLayerPerceptron" entrenada para la caracterización de patrones de fraude, con el fin de ayudar al experto del negocio a examinar y verificar más fácilmente los resultados obtenidos para apoyar la toma de decisiones. El modelo de detección de fraude propuesto integra técnicas de aprendizaje de máquinas, tales como redes neuronales y agrupamiento. De igual forma se desarrolló un prototipo de herramienta de software con fines académicos basado en el modelo propuesto.

El prototipo de herramienta construido sirve como base para futuras investigaciones y desarrollos en el área, ya que permite la integración de nuevas técnicas de aprendizaje de máquinas de manera sencilla. La implementación del algoritmo de extracción de reglas se integro a una herramienta libre de minería de datos (RapidMiner v 4.2).

MOTIVACIÓN

La tarea de detección de fraude no es un tema fácil de resolver dada las múltiples modalidades (lavado de activos, fraude en tarjeta de crédito, pólizas de seguro, etc) y a la evolución que ha tenido con la aparición de nuevas tecnologías. Para detectar fraude, es necesario hacer un análi-

sis profundo sobre la información para sospechar que una transacción puede ser fraudulenta. Los primeros intentos que hicieron las entidades financieras para detectar fraude, fueron sistemas basados en aplicación de reglas que alertaban si las transacciones cumplían los criterios definidos como inusuales en una sentencia SQL [GR94]; estas reglas son obtenidas haciendo análisis histórico sobre los datos y se configuran de acuerdo al comportamiento analizado.

En la actualidad muchas compañías a nivel mundial, utilizan técnicas de minería de datos, para reconocer patrones de comportamiento de las transacciones fraudulentas o de la utilización "normal" de los clientes para detectar transacciones "sospechosas". Dentro de las técnicas que han dado mejor desempeño de clasificación según [KLSH04, STD07, SZC02] se encuentran las redes neuronales dado su alto nivel de exactitud en la predicción y su robustez al ruido en los datos, sin embargo sus mayores críticas son: el tiempo de aprendizaje resulta costoso y su forma de clasificación es una caja negra y en muchas aplicaciones es altamente deseable el poder extraer las reglas de clasificación, con el fin de que expertos interpreten y verifiquen fácilmente los resultados obtenidos. En los últimos años se han desarrollado algoritmos que permiten la extracción de reglas de una red neuronal previamente entrenada, brindando un mayor grado de información para comprender los resultados de clasificación generados por la red. Para esto, se han empleado técnicas como agrupamiento y árboles de decisión [LSL96, SL00, Fu94b] con buena aceptación, las cuales se han probado experimentalmente sobre problemas de clasificación en bases de datos artificiales como: Iris, identificación de características de un tipo de robot(MONK's Problems), pronóstico de hepatitis y enfermedades del corazón.

Dado lo anterior, se plantea la necesidad de diseñar e implementar un modelo de detección de fraude basado en el descubrimiento simbólico de reglas de clasificación provenientes de una red neuronal entrenada que permita la caracterización de patrones para la detección de fraude, validado sobre un conjunto de datos real de una organización colombiana para el envío y pago de remesas, con el fin de ayudar al experto del negocio a examinar y verificar más fácilmente los resultados obtenidos para apoyar la toma de decisiones.

OBJETIVOS

General

Diseñar e implementar un modelo de detección de fraude basado en el descubrimiento simbólico de reglas de clasificación extraídas de una red neuronal.

Específicos

1. Diseñar e implementar una red neuronal que sirva como modelo de clasificación y predicción.

2. Diseñar e implementar un modelo que permite extraer de una red neuronal reglas que faciliten la interpretación de esta.

3. Generar un modelo de detección de fraude a partir de la combinación de una red neuronal entrenada y de la extracción simbólica de reglas de clasificación.

4. Validar el modelo propuesto de detección de fraude con un conjunto de datos real y comparar con otras técnicas de minería de datos como los árboles de decisión.

APORTES Y PUBLICACIONES

La contribución de esta tesis incluye lo siguiente:

1. Revisión del estado del arte en cuanto a las técnicas utilizadas en la detección de fraude.
2. Un prototipo de sistema para la detección de fraude basado en redes neuronales y en la extracción simbólica de reglas de clasificación de estas.
3. Recopilación y descripción detallada del algoritmo de extracción de reglas "M of N".
4. Implementación y adición al software de Rapidminer v.4.2, el algoritmo de extracción de reglas "M of N".
5. Validación y aplicación del modelo de detección de fraude propuesto, a un conjunto de datos de una compañía del sector real en Colombia.

Dentro de las principales publicaciones efectuadas se tiene:

1. *Análisis del Proceso de Giros Internacionales y Control de Lavado de Activos Mediante Minería de Datos*. Revista Tendencias en Ingeniería de Software e Inteligencia Artificial – volumen 2, Julio de 2008. En este artículo se presenta un caso concreto del uso de la minería de datos en una organización colombiana. Específicamente se realiza un análisis al proceso de envío y pago de remesas, al igual que la compra y venta de divisas con el fin de identificar patrones ligados al lavado de activos. Inicialmente se realizan actividades de pare-procesamiento que conlleven a la obtención de datos de alta calidad para la posterior aplicación de diversas técnicas de minería. Dentro de las técnicas de minería escogidas se consideran varas técnicas de clasificación para obtener modelos que permitan mitigar el lavado de activo.

ORGANIZACIÓN DEL DOCUMENTO

Este documento está diseñado de tal forma que los conceptos aplicados no se presentan en detalle sino en forma general, en su lugar se hacen referencias a la literatura que presenta una descripción más completa de dichos conceptos. Esta tesis esta organizada en:

Capítulo 2. Contiene las bases teóricas que soporta el trabajo presentado. Este comprende minería de datos, técnicas para la detección de fraude, aprendizaje mediante la construcción de redes neuronales y extracción de reglas a partir de una red neuronal entrenada.

Capítulo 3. Describe el modelo propuesto de detección de fraude y el prototipo de la herramienta implementada. Para ello se describe el algoritmo de red neuronal propuesto al igual que la técnica de extracción de reglas seleccionada para el modelo.

Capítulo 4. Presenta los resultados del modelo desarrollado, el cual fue probado sobre un conjunto de datos para el envío y pago de remesas de una organización Colombiana, para ello se adelanto varios experimentos con diferentes parámetros de configuración para el entrenamiento de la red neuronal y el proceso de extracción de reglas. De igual forma se hace una comparación de los resultados obtenidos en el proceso de clasificación (red neuronal vs árboles de decisión

y Naive Bayes) y de extracción de reglas (Algoritmo M of N vs Arboles de decisión y reglas de asociación-JRIP-).

Capítulo 5. Presenta las conclusiones del trabajo realizado y algunas ideas para trabajos futuros.

Anexo A. Presenta los diferentes modelos construidos en ”RapidMiner” para el modelo de detección de fraude propuesto, y una guía rápida de la instalación de RapidMiner y la forma de adicionar el operador de aprendizaje “WSR-Rules” que implementa el algoritmo ”M of N” desarrollado en esta tesis.

Anexo B. Presenta el detalle de la implementación efectuada a nivel de diagramas UML de clases del algoritmo de extracción de reglas.

Finalmente, en el Anexo C se presenta un glosario para aclarar los conceptos tratados en este documento.

Capítulo 2

Estado del Arte

En este capítulo se establecen las bases teóricas del proyecto al igual que el conocimiento previo que hay que tener en cuenta para el desarrollo de este; con el fin de hacer más compresible su lectura, en el anexo C se presenta un glosario para aclarar conceptos. El capítulo comienza planteando el problema de la detección de fraude, luego presenta las principales técnicas utilizadas para la detección de fraude, comenzando con las técnicas tradicionales (procesos estadísticos y conocimiento basado en reglas SQL) y posteriormente, se presentan las técnicas de minería de datos, las cuales han permitido encontrar nuevos patrones en base a la información almacenada. Seguidamente, se exponen las características de la red neuronal, y se presenta como una técnica de solución al problema de detección de fraude, acompañada de las técnicas de extracción de reglas a partir de la red neuronal entrenada, que permiten una mejor comprensión de los resultados obtenidos por parte de los expertos. Finalmente se presentan las técnicas más usadas para evaluar y comparar algoritmos de clasificación.

2.1. El Problema de la Detección de Fraude

El problema en la detección de fraude, radica en el análisis de perfiles que permitan conocer el comportamiento de un cliente, con el fin de detectar anomalías. En CRM (Customer Resource Management), el análisis en la información de un cliente, implica una cadena de datos como

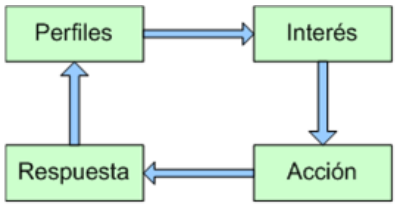

Figura 2.1: Cadena de datos CRM para el análisis de clientes

se muestra en la figura 2.1.

La cadena consiste en cuatro clases de datos[PKB01]:

1. *Datos de perfil*. Datos que representan información histórica del cliente tal como: nombre, profesión, edad, etc.

2. *Datos de Interés*. Datos que representan las tendencias de interés del cliente en los productos de la compañía.

3. *Datos de Acción*. Datos que representan las transacciones entre el cliente y la compañía.

4. *Datos de Respuesta*. Datos que representan la información de servicio al cliente.

En la práctica,la construcción de perfiles de cliente incluye cuatro pasos:

1. Limpieza de datos, para eliminar datos redundantes, con el fin de tener un análisis efectivo de detección de fraude.

2. Selección y Extracción de características, que permitan descubrir indicadores, correspondientes a cambios en comportamientos que indiquen fraude.

3. Modelamiento, para determinar patrones de fraude por un clasificador.

4. Monitoreo y predicción de fraude, con el fin de emitir alarmas.

De los cuatro pasos anteriores, el modelamiento y predicción de fraude son los más importantes, y tienen amplia discusión en el campo del aprendizaje de máquinas. Adicionalmente, una de las dificultades en la detección de fraude, es que típicamente la mayoría de los datos son legítimos (99 %).

2.2. Técnicas para la Detección de Fraude

La detección de fraude no es un tema trivial, las metodologías usadas por los "falsificadores" no son las mismas de hace algunos años; cuando las entidades identifican un patrón de comportamiento, los "falsificadores" ya están pensando en otras alternativas. Actualmente las herramientas para la detección de fraude se pueden clasificar en dos categorías:

1. Técnicas tradicionales y
2. Técnicas de Minería de datos.

2.2.1. Técnicas Tradicionales

Los métodos tradicionales de detección de fraude consisten en una combinación de investigadores y herramientas que reportan alarmas de posibles sospechosos; para ello se utilizan técnicas como:

1. Identificación de clientes que coinciden en listas de control como: OFAC[1], BOE[2], DATACREDITO[3], etc., emitidas por entes internacionales o nacionales.
2. Sistemas basados en la aplicación de reglas que constan de sentencias SQL, definidas con la ayuda de expertos. Esta estructura puede detectar sumas acumulativas de dinero, ingresadas a una cuenta en un corto periodo de tiempo, como un día.
3. Métodos de clasificación estadísticos, como el análisis de regresión de datos, para detectar comportamientos anómalos de cambio en una cuenta, dada una serie de transacciones que efectúa un cliente en un lapso de tiempo[KSM07, BH02].

[1]The Office of Foreign Assets Control of the US Department of the Treasury. Véase http://www.treas.gov/offices/enforcement/ofac.

[2]The Bank Of England as the Treasury's agent for the purpose of administering financial sanctions. Véase http://www.hm-treasury.gov.uk/financialsanctions

[3]DataCrédito es la Central de Información Crediticia líder en el mercado andino con presencia en Colombia, Venezuela y Ecuador. Véase http://www.datacredito.com.co

4. Análisis de relaciones. Este análisis permite encontrar relaciones entre elementos de información como transacciones, cuentas y participantes. Esta técnica requiere un esquema supervisado [BH02].

2.2.2. Técnicas de minería de datos

Debido a los grandes volúmenes de datos que una organización puede llegar a tener, el obtener conocimiento a partir de estos no es una tarea fácil. Con este fin, investigadores han dado origen a campos de investigación como el descubrimiento de conocimiento en bases de datos (Knowledge Discovery in Database - KDD), y en especial la fase de minería de datos (Data Mining) como se ilustra en la figura 2.2.

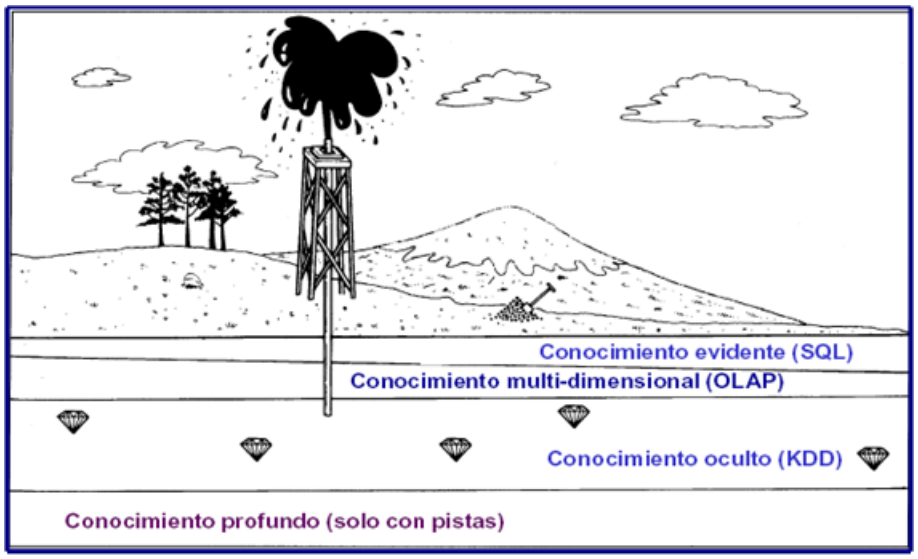

Figura 2.2: Tipos de conocimiento

La minería de datos ofrece un rango de técnicas que permiten identificar casos sospechosos, basados en modelos. Estos modelos se pueden clasificar en:

1. *Modelos de datos inusuales.* Estos modelos, pretenden detectar comportamientos raros en un dato respecto a su grupo de comparación, o con el mismo, por ejemplo la consignación de altas sumas de dinero en efectivo. Para este caso, se puede emplear técnicas de análisis de Agrupamiento (Clustering), seguido de un análisis de detección de ”Outlier”.

2. *Modelos de relaciones inexplicables.* A través de este tipo de modelos, se desea encontrar relaciones de registros que tienen iguales valores para determinados campos, resaltando el hecho que la coincidencia de valores debe ser auténticamente inesperado, desechando similitudes obvias como el sexo, la nacionalidad. Por ejemplo, la transferencia de fondos entre dos o más compañías con la misma dirección de envío. Para este caso se pueden aplicar técnicas de Agrupamiento (Clustering) para encontrar grupos sospechosos y reglas de asociación.

3. *Modelos de características generales de Fraude.* Con estos modelos se pretende, una vez detectado ciertos casos, hacer predicciones de futuros ingresos de transacciones sospechosas. Para estas predicciones usualmente se emplean técnicas de regresión, árboles de decisión y redes neuronales.

De igual forma, taxonómicamente la minería de datos se puede dividir en dos clases: descriptiva y predictiva (clasificación) según [HK06] como se presenta en la figura 2.3

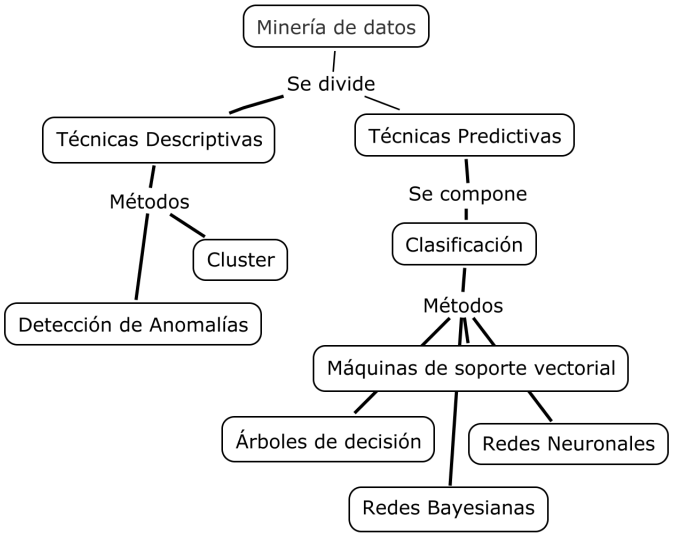

Figura 2.3: Taxonomía: Técnicas de Minería de Datos para Detección de Fraude

2.2.2.1. Técnicas de Minería Descriptiva

El objetivo de este tipo de minería, es encontrar patrones (correlaciones, tendencias, grupos, trayectorias y anomalías) que resuman relaciones en los datos [CHY96]. Dentro de las principales técnicas descriptivas encontramos:

1. *Detección de Anomalías (Outlier)*: La meta principal en la detección de Anomalías, es encontrar objetos que sean diferentes de los demás. Frecuentemente estos objetos son conocidos como "Outlier" [HDXH06, HS03]. La detección de anomalías también es conocida como detección de desviaciones [PFF00], porque objetos anómalos tienen valores de atributos con una desviación significativa respecto a los valores típicos esperados. Aunque los "Outlier" son frecuentemente tratados como ruido o error en muchas operaciones, tales como Agrupamiento, para propósitos de detección de fraude, son una herramienta valiosa para encontrar comportamientos atípicos en las operaciones que un cliente realiza en una entidad financiera. En términos de salida, las técnicas actuales de detección de "Outlier" se clasifican en:

   a) *Técnicas basadas en Modelos* [DG93]. Se basan en el campo de la estadística; dada la premisa de conocer la distribución de los datos. Entre estas técnicas se resalta: Método de Incertidumbre y Método de "convex hull".

   b) *Técnicas basadas en proximidad*[CP95, PWM02]. Esta técnica se fundamenta en el manejo de distancias entre objetos, entre mayor sea la distancia del objeto respecto a los demás, éste es considerado como un "Outlier". Entre los principales métodos se encuentra: la distancia de Mahalanobis y la distancia Euclidiana.

   c) *Técnicas basadas en densidad*. Se hace uso de la estimación de densidad de los objetos, para ello, los objetos localizados en regiones de baja densidad, y que son relativamente distantes de sus vecinos se consideran anómalos. Entre los principales métodos se encuentra: SHV[PWM02]. (Smallest half-volumen),LOF[HS03](Local Outlier Factor).

Este método de minera de datos, generalmente es de aprendizaje no supervisado, ya que en la mayoría de los casos, no se conoce la clase, para ello se asigna una calificación a cada instancia que refleja el grado con el cual la instancia es anómala.

2. *Agrupamiento (Clustering):* El análisis de clúster es un proceso que divide un grupo de objetos, de tal forma que los miembros de cada grupo son similares de acuerdo a alguna métrica. El agrupamiento de acuerdo a la similitud, es una técnica muy poderosa, la clave para esto es trasladar alguna medida intuitiva de similitud dentro de una medida cuantitativa, como se ilustra en la figura 2.4.

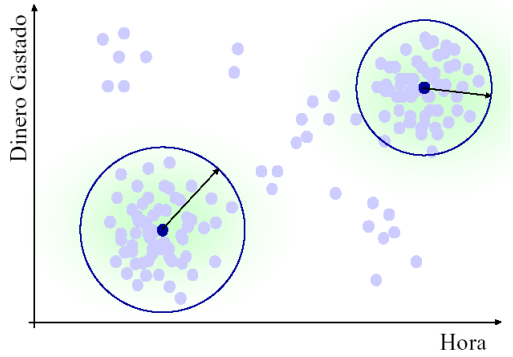

Figura 2.4: Agrupamiento por clúster

Las técnicas de Agrupamiento son utilizadas comúnmente para hacer segmentación, y su gran aplicación está en estrategias de mercadeo, mediante las cuales se determinan conjuntos de clientes que poseen el mismo comportamiento, para hacer llegar ofertas especialmente diseñadas al perfil de dichos clientes. Las técnicas de segmentación permiten identificar claramente el comportamiento de un grupo de casos que difiere de otros grupos o conjuntos, sin embargo algunos autores[MR05] plantean que por lo general, los clúster son resultados difíciles de entender. Algunas veces, se puede utilizar un árbol de decisión a la salida del clúster, para explicar con precisión el comportamiento o características de los casos que conforman el clúster.

Los algoritmos de clúster funcionan con una metodología basada en la construcción inicial de un gran clúster, y luego la subdivisión del mismo hasta encontrar grupos

de muestras muy cercanas, otros por el contrario, parten asumiendo que cada registro es un clúster, y luego empiezan a agrupar registros hasta que se consolidan clúster no superpuestos más grandes. Entre los diferentes tipos de clúster se tienen [TSK05, ZB06]:

a) *Clúster bien separados.* Esta definición idealista parte del hecho que todos los objetos de un grupo deben ser suficientemente similares.

b) *Clúster basados en el centro.* Un clúster es un conjunto de objetos en el que un objeto está más cerca al centro del clúster, que al centro de otro clúster.

c) *Clúster contiguos.* Un clúster es un conjunto de puntos, donde un punto en el clúster está más próximo a otro punto o puntos del clúster, que a cualquier otro punto que no pertenezca al clúster.

d) *Clúster basados en densidad.* Este tipo de agrupamiento, se basa en el hecho de tener grupos en regiones de alta densidad, separados por regiones de baja densidad.

e) *Clúster de propiedad o Conceptual.* Son clúster que tienen propiedad compartida o representan un concepto particular, es decir, hay puntos en común entre dos grupos.

Entre los principales algoritmos usados en el análisis de clúster se encuentra:

a) *Algoritmo K-means*[MR05]. Este algoritmo se fundamenta en clúster basados en el centro, en términos de un centroide, el cual usualmente es la media de un grupo de puntos, y típicamente aplica a objetos en espacios continuos n-dimensionales. En esta técnica se debe especificar el número de clúster que se desea encontrar.

b) *Algoritmo DBSCAN*[TSK05]. Se basa en clúster de densidad, en los cuales los grupos se localizan en las regiones de alta densidad, y son separados por regiones de baja densidad. Este algoritmo genera de manera automática el número de clúster. Los puntos en baja densidad son considerados como ruido y se ignoran.

2.2.2.2. Técnicas de Minería de Clasificación

El objetivo de este tipo de técnicas, es clasificar un valor particular de un atributo basado en otros atributos. El atributo a clasificar es comúnmente llamado ”clase” o variable dependiente, mientras que los atributos usados para hacer la predicción se llaman variables independientes[TSK05]. Dentro de las principales técnicas encontramos:

1. *Árboles de decisión*: De las técnicas de aprendizaje, son el método más fácil de utilizar y entender. Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica, de tal manera que la decisión final a tomar, se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta sus hojas[Qui86]. Se utilizan comúnmente cuando se necesitan detectar reglas del negocio que puedan ser fácilmente traducidas al lenguaje natural o SQL, o en la construcción de modelos de clasificación. Existen dos tipos de árboles: los de clasificación, mediante los cuales un registro es asignado a una clase en particular, reportando una probabilidad de pertenecer a esa clase , y los árboles de regresión, que permiten estimar el valor de una variable numérica objetivo.

   El funcionamiento general de un árbol se basa en la aplicación de premisas que pueden ser cumplidas, o no, por un registro; el registro pasa a través del árbol de premisa en premisa hasta que se evalúa totalmente o hasta que encuentra un nodo terminal, como se aprecia en la figura 2.5. Las premisas pueden ser vistas como una serie de preguntas sobre las variables de entrada al modelo, tales como ingresos mayores a 500?, sexo masculino o femenino?, etc.,; cada registro, que contiene dentro de si las variables de entrada, describe un camino dentro del árbol por el cual pasa hasta obtener una calificación o una clasificación según sea el caso. Los caminos que describe el árbol para llegar a los nodos terminales, representan el conocimiento adquirido y permiten la extracción de reglas de clasificación de la forma SI - ENTONCES .

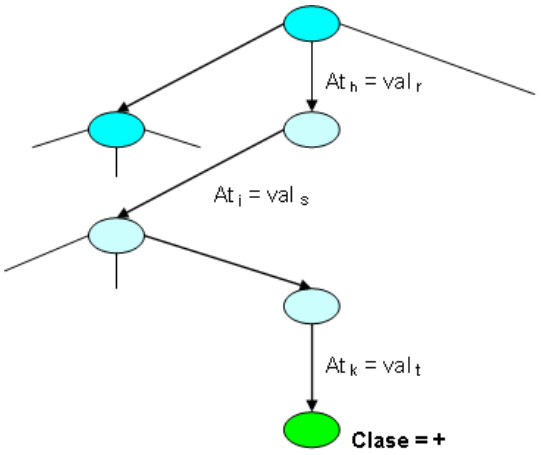

Figura 2.5: Esquema general de un árbol decisión

Según el tema de estudio, los árboles pueden crecer tanto que resultan difíciles de interpretar, o muy cortos que arrojan respuestas obvias o insuficientes. La mayoría de los algoritmos y herramientas en el mercado permiten la configuración de los parámetros como el tamaño mínimo de nodos, dado que cada uno de los nodos del árbol corresponden a una pregunta sobre una variable especifica, los árboles de decisión no pueden descubrir reglas que impliquen relaciones entre variables En la literatura han aparecido numerosos algoritmos de aprendizaje de árboles de decisión, entre los más populares se encuentran:

a) *CART* [Breiman, 1984]. Se basa en el lema "divide y vencerás" [ZB06], son métodos que construyen árboles binarios basados en el criterio de partición GINI y que sirven para clasificación como para regresión. La poda se basa en una estimación de la complejidad del error.

b) *ID3.* Propuesto por Quinlan en 1986[Qui86], el ID3 es considerado el árbol de decisión más simple, usa la ganancia de información como criterio de separación. El árbol crece hasta encontrar un nodo final. No emplea procedimientos de poda, ni manejo de valores perdidos.

c) *C4.5.* Es la evolución del ID3, presentado por Quinlan en 1993[BF01]

2. *Redes Neuronales:* Las redes neuronales consisten en "neuronas" o nodos interconectados que se organizan en capas. Por lo regular, los modelos neuronales constan de tres capas: de entrada, oculta y de salida, como se observa en la figura 2.6(tomada de Darren Dancey [DMB04]). Cada neurona evalúa los valores de entrada, calcula el valor total de entrada, compara el total con el mecanismo de filtrado (valores de umbral), y en seguida determina su propio valor de salida. El comportamiento complejo se modela conectando un conjunto de neuronas. El aprendizaje o "capacitación" ocurre modificando la "fuerza de conexión" o los parámetros que conectan las capas. Las redes neuronales se acondicionan con muestras adecuadas de la base de datos.

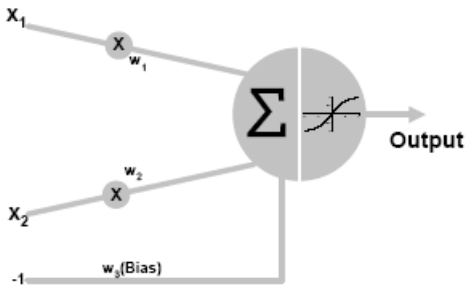

Figura 2.6: Esquema general de una red neuronal

Las redes neuronales aprenden en forma supervisada o no supervisada. En la modalidad supervisada, la red neuronal intenta predecir los resultados para ejemplos conocidos, compara sus predicciones con la respuesta objetivo y aprende de sus errores. Las redes neuronales supervisadas se emplean para predicción, clasificación y modelos de series históricas. El aprendizaje no supervisado es eficaz para la descripción de datos, pero no para la predicción de resultados. Las redes supervisadas crean sus propias descripciones y validaciones de clase y trabajan exclusivamente a partir de los patrones de datos. Las redes neuronales se ven afectadas por tiempos prolongados de aprendizaje. Debido a que actúan como una caja negra, algunos analistas empresariales no confían en ellas.

Entre los modelos más utilizados en redes neuronales se encuentran[HK06]:

a) *Perceptrón Multicapa(MLP) o Hacia Adelante (Feedforward).* Es el modelo más

estudiado y usado en la industria. Un MLP es una red conformada por una capa de entrada, una o varias capas ocultas, una salida y una función de transferencia en cada nivel. Se caracterizan por tener una conexión completa entre capas sucesivas, es decir, cada nodo en una capa está totalmente conectado sólo a todos los nodos en las capas adyacentes.

b) *Hopfield*. Son un tipo especial de redes, capaces de guardar recuerdos o patrones como el cerebro, no tienen una arquitectura de capas, sino por el contrario, es una sola capa de neuronas completamente interconectadas, en las cuales hay bucles de retroalimentación entre las neuronas.

c) *Mapas Auto-organizados* (Kohonen's Self-organizing Maps -SOM). Son modelos de redes neuronales para la reducción de dimensiones y agrupación de datos, con el fin de visualizar similitudes entre patrones

3. *Redes de Creencia Bayesiana(BBN):* La clasificación Bayesiana se basada en el teorema estadístico de Bayes, el cual provee un cálculo para la probabilidad a posteriori. De acuerdo al teorema de Bayes, si H es una hipótesis, tal que, el objeto X pertenece a la clase C, entonces la probabilidad que la hipótesis ocurra es: P(X| H) = (P (X|H) * P (H))/P(X). Una red de Creencia Bayesiana (BBN)[LSL96]provee una representación gráfica de las dependencias entre un conjunto de atributos. Una BBN se compone principalmente de dos elementos:

   a) Un grafo acíclico que codifica la dependencia de relaciones entre un conjunto de variables.

   b) Una tabla de probabilidad asociada a cada nodo para su nodo padre inmediato. En una BBN, para cada nodo X, existe una tabla de probabilidad condicional, en la cual se especifica la probabilidad condicional de cada valor de X, para cada posible combinación de los valores se sus padres (distribución condicional P (x|padre (x))). La estructura de la red puede ser definida o ser inferida desde los datos. Para propósitos de clasificación uno de los nodos puede definirse como nodo "clase". La

red puede calcular la probabilidad de cada alternativa de ”clase”.

4. *Máquinas de soporte Vectorial:* Las máquinas de soporte vectorial (SVM) son un conjunto de algoritmos para clasificación y regresión propuesta por Vapnik y su grupo AT&T Bell laboratorios [PKB01]. En simples términos, una SVM es un perceptrón (como una red neuronal) y es idealmente adecuado para la clasificación binaria de patrones que son linealmente separables [KPJ+03]. La idea principal de la SVM es obtener un único separador de hiperplanos que maximice el margen entre la separación de dos clases, como se observa en la Figura 2.7. La característica de los vectores que se encuentran en la frontera que separa la definición de este Margen, en la jerga del álgebra lineal, se denomina ”Support Vector”.

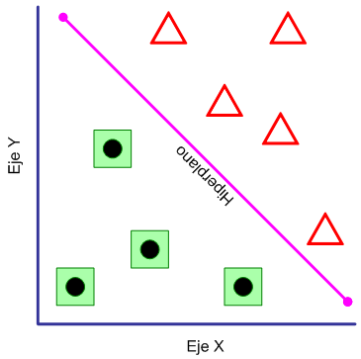

Figura 2.7: Ejemplo de un separador lineal de SVM

En el modelamiento de patrones de fraude, las SMV se pueden trabajar como un modelo de clasificación binaria, donde ”+1” representa a los clientes sospechosos de fraude y ”-1” representa a los clientes usuales, para ello se tiene un modelo [PKB01] en el que dado F = {a1; a2;... ak} un conjunto de características de un cierto tipo de comportamiento de un cliente, obtenidas por algún conocimiento previo.

En la tabla 2.1, se presenta un breve resumen de las tareas, metas y técnicas de Minería más utilizadas en la Detección de Fraude.

Tarea	Meta	Técnica de Minería
Encontrar datos Inusuales	Detectar registros con valores anormales. Detectar múltiples ocurrencias de valores. Detectar relaciones entre registros.	Análisis de Anomalías
Identificar Relaciones Inesplicables	Determinar perfiles. Determinar registros duplicados. Detección de registros con referencias de valores anormales. Detectar relaciones indirectas. entre registros Detectar registros con combinaciones de valores anormales.	Análisis de Cluster Análisis de Cluster y Anomalías Análisis de Relaciones Asociación
Características Generales de Fraude	Encontrar criterios, tales como reglas. Calificación de transacciones sospechosas.	Modelos Predictivos

Cuadro 2.1: Técnicas de minería de datos para la detección de fraude

2.3. Redes Neuronales

En las dos últimas décadas se ha visto un gran número de investigaciones y aplicaciones de las Redes Neuronales para una variedad de problemas de clasificación del mundo real. Varios estudios empíricos han señalado que hay algunos dominios de problema para el cual las redes neuronales ofrecen una precisión superior de clasificación en comparación a otros algoritmos de aprendizaje [Cra96].

De igual forma hay una gran variedad de arquitecturas de redes neuronales y métodos de aprendizaje tanto para aprendizajes supervisados como no supervisados. El trabajo en esta tesis se centra en redes neuronales "MultiPerceptron" aplicadas a tareas de clasificación, y por lo tanto la discusión que sigue es restringida a este tipo de enfoque de red neural.

2.3.1. Red Neuronal - Proceso de Clasificación

Como se muestra en la figura 2.8, una red neuronal "MultiPerceptron" se compone de varias capas y nodos de procesamiento. El estado de un nodo en un momento dado, está representado por su activación, que es un valor real, normalmente en el intervalo [0, 1] o en el intervalo [-1, 1]. La capa de entrada de una red contiene nodos cuyas activaciones representan los valores de las características del dominio del problema para el cual la red está siendo aplicada. Los nodos en la capa de salida representan las decisiones tomadas por la red. Entre los nodos de entrada y de salida, puede haber un número de capas ocultas.

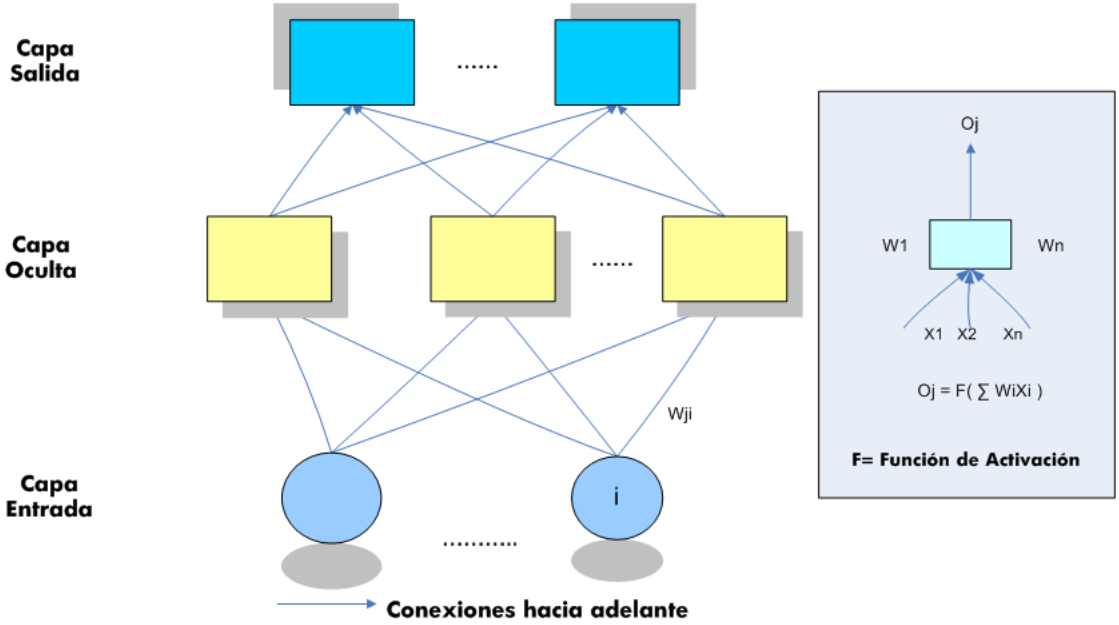

Figura 2.8: Modelo computacional de una Red Neuronal

Una red neuronal para clasificación que tiene únicamente entrada y salidas es capaz de representar únicamente decisiones lineales en el espacio (Minsky & Papert, 1969)[MP88]. Para representar espacios más complejos, es necesario adicionar capas ocultas. El rol de las capas ocultas es transformar el espacio de entrada en otro espacio más adecuado para hacer más discriminaciones lineales de los nodos de salida.

La Red Neuronal "MiltilayerPerceptron" comienza estableciendo valores de activación para

los nodos de entrada, los cuales representan un dominio de problema particular. El proceso continua a través de las conexiones a los nodos de las capas ocultas y de salida. La respuesta proporcionada por la red está determinada por la activación resultante de los nodos de salida.

Para un ejemplo concreto, la entrada de un nodo oculto o de salida está dado por:

$net_i = \sum_j w_{ij} a_j + \theta_i$

Donde

w_{ij} es el peso del nodo j al nodo i, a_j es el valor de activación de entrada del nodo y θ_i es el sesgo(bias) para el nodo i, que puede considerarse como un nodo de alto (o bajo) valor de activación antes de que recibe alguna señal de un nodo de la red. La activación de un nodo oculto o de salida está determinada por la función de transferencia (algunas veces llamada función de activación). Una función muy común es la Sigmoide:

$a_i = \frac{1}{1+e^{-net_i}}$

Esta función da como rango de activación el intervalo [0,1]. Una función similar de transferencia es la tangente hiperbólica, para la cual el rango de activación está entre [-1, 1]:

$a_i = \frac{e^{net_i} - e^{-net_i}}{e^{net_i} + e^{-net_i}}$

2.3.2. Red Neuronal - Proceso de Aprendizaje

El algoritmo de aprendizaje más usado en las redes neuronales es el "Backpropagation" (Rumelhart et al., 1986)[Fu94a]. El aprendizaje en la red neuronal involucra modificar los pesos y sesgos (bias) de la red para minimizar la función de costo. La función de costo siempre incluye un término de error, el cual, es una medida de que tan cerca está la predicción respecto a la clase dado unos ejemplos del conjunto de entrenamiento.

El aprendizaje e inferencia a modo general puede extraerse de la siguiente forma (según Limin Fu [Fu94a]):

1. *Algoritmo de Aprendizaje(A)*

   Dado n instancias a entrenar

a) Inicializar los pesos de la red. Fijar *i=1*

b) Recorre la *ith* instancia de la red en la capa de entrada

c) Obtener el nivel de activación del nodo de salida usando el algoritmo de inferencia (ver algoritmo de inferencia, numeral B). Si el desempeño de la red neuronal encontrado es el adecuado(o según el criterio de parada) terminar.

d) Actualizar los pesos dada las reglas de aprendizaje de la red

e) Si i=n , entonces fijar i=1, en otro caso , incrementar i en uno e ir al paso (b).

2. *Algoritmo de inferencia (B)*

a) Recorrer cada nodo de la red neuronal de la capa de entrada

b) Calcular el nivel de activación de los nodos a través de la red

c) Si el nivel de activación de todos los nodos de salida son calculados, entonces, terminar, caso contrario ir al paso (b). Sin embargo, si la red se encuentra inestable terminar y fallo inferencia.

2.3.3. Redes Neuronales - Interpretación

El aprendizaje de una red neuronal está definido por:

1. La topología de la red
2. La función de activación usada para los nodos de la capa oculta y de salida
3. Los parámetros asociados con las conexiones de la red (i.e pesos, bias)

Las hipótesis aprendidas por las redes neuronales son difíciles de comprender. Primero las redes neuronales típicamente tienen cientos o miles de parámetros con valores reales. Estos parámetros codifican las relaciones entre las características de entrada y las clases de salida. Aunque la codificación de un sólo parámetro de este tipo no suele ser difícil de entender, el gran número de parámetros en una red puede hacer que la tarea de comprensión sea muy

difícil. Por otra parte, en redes multi-capa, estos parámetros puede representar relaciones no lineales entre las características de entrada y los valores de salida. Así, normalmente no es posible determinar, el efecto de una determinada característica en el valor objetivo, ya que este efecto puede estar mediado por los valores de otras características. Estas relaciones no lineales representadas por las unidades ocultas en una red, combinan múltiples características creando un modelo de dependencias entre las características. Comprender las unidades ocultas es a menudo difícil debido a que tienen representaciones distribuidas de aprendizaje[Cra96]

2.4. Extracción de reglas desde una red neuronal

Una de las grandes desventajas de las Redes Neuronales es la falta de transparencia en el conocimiento generado, dada la estructura y los parámetros de las conexiones entre los elementos de la red. Es por eso, que a comienzos de los noventa empieza a aparecer el concepto de extracción de reglas(RE) para expresar el conocimiento embebido en una Red Neuronal. Varios conceptos y algoritmos se han desarrollado hasta hoy,[ADT95] da una amplia recopilación sobre las investigaciones en este campo.

Hay dos conceptos principales en la extracción de reglas. Ambos tienen en común que toman una red neuronal entrenada (ANN). La ANN(o sus pesos) son analizados para extraer un conjunto de reglas las cuales pueden representar el conocimiento embebido en la red neuronal[MHRT06]. En la figura 2.9, se presenta un simple contexto del proceso.

Figura 2.9: Principio de extracción de reglas desde una ANN entrenada

Los dos conceptos principales son:

1. *Caja Negra*. Podemos tratar la red neuronal entrenada como una caja negra, sin tener en cuenta su arquitectura y pesos. Los valores de entrada son mapeados por la Red Neural Entrenada con los valores de salida. En la literatura estos tipos de algoritmos son llamados "pedagógicos"

2. *Descomposicional*. Este tipo de algoritmos de extracción se basa en los pesos de la red. Para ello hay algoritmos que tratan de agrupar pesos similares para simplificar las reglas. Otros buscan caminos dominantes a través de la red.

2.4.1. Propiedades de los algoritmos de extracción de reglas

En la recopilación realizada por Andrews[ADT95]sobre algoritmos de extracción de reglas, propuso una taxonomía para clasificarlos según las siguientes propiedades:

1. *Poder expresivo de la extracción (formato de la regla)*

   Esta taxonomía de clasificación, se centra en la forma como se presentan las reglas extraídas al usuario final. A la fecha se utilizan los siguientes formatos:

   a) *Reglas simbólicas convencionales (booleanas, proposicionales)*. Este tipo de representación es la de la forma SI A=x ENTOCES C=y.

b) *Reglas basadas en conjuntos de lógica difusa.* Este tipo de representación puede incluir reglas probabilísticas o de lógica difusa de la forma SI... ENTONCES... SINO ESTRUCTURA. Por ejemplo se permite tener verdades parciales como SI X es bajo y Y es alto ENTONCES Z es medio, donde alto, bajo y medio son conjuntos difusos con miembros de funciones correspondientes.

c) *Reglas expresadas en primer orden lógico.* Con este tipo de formato se permite tener reglas con cuantificadores y variables.

2. *Calidad de las reglas extraídas*

Esta clasificación permite dar una medida de cómo la tarea de extracción necesita un alto grado de explicación. Para ello se han definido criterios de calidad para la evaluación de las reglas generadas desde la red neuronal, clasificados en:

a) *Exactitud de la regla.* Con esta característica se tiene una medida de como la regla puede clasificar un conjunto inédito de ejemplos correctamente.

b) *Fidelidad de la regla.* Corresponde a la medida en que la regla imita el comportamiento de la Red Neuronal a partir de la cual se extrajo.

c) *Consistencia de la regla.* Esta característica mide como bajo diferentes sesiones de entrenamiento de la red neuronal, las reglas generadas desde esta producen la misma clasificación para ejemplos inéditos.

d) *Comprensibilidad de la regla.* Esta característica permite medir el tamaño del conjunto de reglas en cuanto al número de reglas y el número de antecedentes por regla.

3. *Transparencia de la técnica de extracción*

Esta dimensión de clasificación está diseñada para ver que la relación entre la técnica de extracción de las reglas y la arquitectura interna de la red neuronal entrenada es independiente de la técnica usada. Entre las técnicas de extracción se tiene:

a) *Descomposicional (Decompositional*). Esta técnica se centra en extraer reglas de la

red neuronal en su mínimo nivel de granularidad, desde el punto de vista de nodos ocultos y de salida. Esta técnica requiere que el resultado producido de los nodos de la capa oculta (hidden) y de salida (output) en la red entrenada sean mapeados a una salida binaria (si, no) que corresponde a la noción de regla consecuente. Por lo tanto, cada nodo de la capa de salida u oculta puede ser interpretada como una regla booleana, lo cual reduce el problema de la extracción de la regla a determinar en que situaciones la regla es verdadera. El subconjunto de reglas extraídas del nivel individual (nodos de la capa de salida y oculta) son agregadas para formar la relación global de la regla.

b) *Pedagógica (Pedagogical)*. Es una técnica que extrae relaciones globales entre las entradas y salidas de la red neuronal directamente, sin analizar las características detalladas de la solución de la red neuronal, es decir, se ve la red como una caja negra.

c) *"Eclectic*". Este tipo de extracción fue concebido para el uso de técnicas híbridas que analizan la red neuronal desde un nivel individual pero que extrae reglas a nivel global.

4. *Complejidad algorítmica de la técnica de extracción*

   La inclusión de esta dimensión se refleja en la exigencia casi universal de los algoritmos que sustentan el proceso de extracción de reglas sean tan eficaces como sea posible. En particular, un elemento crucial en el desarrollo de un proceso de extracción de reglas de una Red Neuronal Entrenada(ANN) es limitar el tamaño de la solución del espacio de búsqueda. La dificultad en esta clasificación ha residido en que no todos los autores han informado sobre el tema.

5. *Portabilidad de la técnica de extracción para varias arquitecturas de redes neuronales*

   Esta dimensión trata de ver como el proceso de extracción de reglas es independiente de la arquitectura de red neuronal y el tipo de entrenamiento usado, para dar portabilidad a la solución.

2.4.2. Algoritmos de extracción de reglas

En esta sección se presenta los trabajos realizados en el área de la extracción de la reglas a partir de una red neuronal entrenada. El propósito de esta sección es proporcionar los antecedentes y el contexto adecuado para el trabajo desarrollado en los siguientes capítulos. Dentro de los principales algoritmos desarrollados en la extracción de reglas según Andrew[ADT95] tenemos:

1. *Algoritmo "M of N "*. Fue publicado por Towell & Shavlik(1993)[TSN90, MHRT06] y es basado en el algoritmo "SUBSET". Difiere de este en la forma de hacer la búsqueda de las reglas de la forma: SI(M de los siguiente N antecedentes son verdaderos) ENTONCES...

   Los pasos del algoritmo son:

   a) Para cada nodo, hacer un agrupamiento de los pesos de entrada en grupos similares

   b) Encontrar el promedio de cada grupo a partir de los pesos que forman el grupo.

   c) Remover grupos que tengan poco efecto en los nodos de salida

   d) Depurar pesos y actualizar el bias de la red

   e) Formar reglas de cada nodo

   f) Simplificar reglas a la forma "M-of-N"

2. *Algoritmo VIA*. Fue inventado por Trun in 1993[Thr94]. El principio básico de "VIA" es el manejo de intervalos de validación. Cada intervalo limita la activación de patrones en la red. Más específicamente, un intervalo de validez de una unidad específica en un rango máximo de su valor de activación. Inicialmente, el usuario puede asignar intervalos arbitrarios a todos (o un subconjunto de todos) los nodos y "VIA" refina estos intervalos. Esto se hace de forma iterativa, detectando y excluyendo los valores de activación que son lógicamente incoherentes con los pesos y los sesgos de la red. Este mecanismo, garantiza que cada patrón de activación es consistente con el conjunto inicial de intervalos.

   Hay dos posibles salidas del proceso de análisis "VIA":

   a) *La rutina converge*. Los intervalos resultantes constituyen un sub-espacio que incluye todos los patrones de activación en concordancia con la periodicidad inicial.

b) *Contradicción* (i.e un intervalo vacío). Si se genera un intervalo vacío, significa que el límite inferior de un intervalo excede a su límite superior, por ende no habrá patrón de activación alguno que pueda satisfacer las limitaciones impuestas por el intervalo inicial de validez. Por consiguiente, los intervalos iniciales son incompatibles con los pesos y los sesgos de la red.

3. *Algoritmo "SUBSET"*. Fue desarrollado por Towell (1991)[TSN90] , es una de las primeras aproximaciones a un algoritmo descomposicional. "SUBSET" es una búsqueda basada en la extracción de reglas de la forma SI - ENTONCES desde una red Multi-Layer Perceptron (MLPs) con unidades binarias de salida. Adicionalmente la red tiene únicamente valores binarios de entrada. El algoritmo encuentra las combinaciones de pesos positivos conectados a un nodo que se active. Estos conjuntos de conexiones se combinan con pesos negativos para formar reglas que activen el nodo. El algoritmo realiza una búsqueda exhaustiva sobre todas las conexiones entrantes de un nodo y, como tal, es necesario restringir el espacio de búsqueda.

4. *Algoritmo "RULENET".* Es una algoritmo que emplea una técnica descomposicional para generar reglas de la forma SI... ENTONCES ... SINO. Fue propuesto por McMillan, C., Mozer, M. C. and Smolensky[MMS92]. Se basa en los pasos del método científico, induce una hipótesis e iterativamente refina la hipótesis hasta explicar la observación. El algoritmo se basa en los siguientes pasos:

   a) Entrenamiento de la Red Neurona(conjunto de entradas y salidas)

   b) Extracción simbólica de reglas(usando las conexiones más fuertes de la red)

   c) Insertar reglas extraídas en la red(prueba de hipótesis)

   El proceso termina cuando, la base de las reglas extraídas caracteriza el dominio del problema. "RuleNet" tiene 3 capas definidas de la siguiente forma:

   a) Capa de entrada

   b) Capa nodo de condición

c) Capa de salida

El vector de pesos *ci* que conecta los nodos de entrada a los nodos de condición es usado para detectar la condición de la regla para ser aprendida. Después del entrenamiento, cada nodo de condición representa una regla. La matriz de pesos Ai que conecta los nodos de condición i a los nodos de salida es un conjunto que asegura que hay un único mapeo entre las entradas y las salidas. La extracción de reglas es alcanzada por la descomposición del vector de pesos ci para los componentes de condición de la regla y la descomposición de la matriz Ai para los componentes de acción de la regla.

5. *Algoritmo "REANN"*. Propuesto por S. M. Kamruzzaman[KI06], un estándar de tres capas de la red neuronal "MultilayerPerceptron" es la base del algoritmo. El algoritmo consta de las siguientes fases.

   a) En la primera fase, el número de nodos ocultos de la red se determina automáticamente de una manera constructiva por adición de nodos, uno tras otro basado en el desempeño de de la red sobre los datos de entrenamiento.

   b) En la segunda fase, la Red Neuronal es podada de tal manera que las conexiones pertinentes y nodos de entrada son eliminados, mientras que su exactitud de predicción se mantiene.

   c) En de la tercera fase, los valores de activación de los nodos ocultos son discretizados mediante una agrupación heurística eficiente.

   d) finalmente en la cuarta fase, se extraen las reglas mediante el examen de los valores de activación discretos de los nodos ocultos utilizando un algoritmo de extracción de reglas.

2.5. Evaluación y comparación de algoritmos de clasificación

Los algoritmos de aprendizaje de máquinas inducen clasificadores que dependen de un conjunto de entrenamiento, y de la necesidad de realizar pruebas estadísticas para:

- Evaluar el error esperado del algoritmo de clasificación y
- Comparar el error esperado de dos algoritmos de clasificación para evaluar cual es mejor.

Sin embargo, no podemos mirar el error del conjunto de entrenamiento y decidir sobre este. La tasa de error del conjunto de entrenamiento, por definición según [Alp04] es siempre menor que la tasa de error del conjunto de pruebas. Dado lo anterior se necesita de un conjunto de validación diferente al de entrenamiento. Incluso una sola corrida sobre el conjunto de validación y entrenamiento no puede ser suficiente, dado las siguientes razones según[Alp04] :

- Los conjuntos de entrenamiento y validación pueden ser pequeños y tener casos excepcionales, como ruido y valores atípicos, que pueden engañarnos
- El método de aprendizaje puede depender de otros factores aleatorios que afectan a la generalización

Para resolver lo anterior, se han empleado métodos de validación cruzada que permiten hacer una mejor medición del error (i.e matriz de confusión) y métodos para evaluar el desempeño de un algoritmo de clasificación (i.e prueba de ANOVA), los cuales se presentan a continuación en más detalle.

2.5.1. Métodos de Validación Cruzada

Este tipo de estrategia, obtiene un par de conjuntos de entrenamiento y validación, a partir del conjunto de datos original, para ello divide cada parte aleatoriamente en K partes, luego divide cada parte de forma aleatoria en dos partes, una entrenamiento y otra para validación. K es típicamente 10 o 30 dependiendo de la cantidad de registros del conjunto de datos. Este proceso, se repite dividiendo los mismos datos de manera diferente K veces. Dentro de los principales métodos según [Alp04],se tiene:

1. *Validación cruzada de K-Fold.* El conjunto de datos X es dividido aleatoriamente en K partes iguales, $X_j, j = 1, ..., K$. Para generar cada pareja, se mantiene una parte de

K para el conjunto de validación (V) y se combinan K-1 partes para el conjunto de entrenamiento (T). Este proceso se repite K veces, obteniendo K parejas de la siguiente forma:

$V_1 = X_1 T_1 = X_2 U X_3 U ... U X_k$

$V_2 = X_2 T_2 = X_1 U X_3 U ... U X_k$

.

.

.

$V_k = X_k T_k = X_1 U X_3 U ... U X_{k-1}$

2. *Validación cruzada de 5*2.* Propuesta por Dietterich(1998), la cual usa un conjunto de entrenamiento y validación de igual tamaño, para ello se divide el conjunto de datos X aleatoriamente en dos partes:$X_1^{(1)}$ y $X_2^{(2)}$, estableciendo el primer conjunto de entrenamiento y validación : $T_1 = X_1^{(1)}$ y $V_1 = X_1^{(2)}$. Luego hay una función de intercambio para obtener la segunda pareja: $T_2 = X_1^{(2)}$ y $V_2 = X_1^{(1)}$. Esto corresponde al primer grupo de validación (fold), este proceso se repite 5 veces, obteniendo diez conjuntos de entrenamiento y validación

3. *Remuestro "Bootstraping"*. Esta alternativa, permite generar nuevas muestras a partir de una muestra original con reemplazo. Si se efectúa una sola validación, el conjunto de datos original es usado como conjunto de validación, caso contrario, se puede generar muchos conjuntos de entrenamiento y validación. La probabilidad de tomar un ejemplo es 1/N y la probabilidad de no escoger una instancia es 1-1/N. La probabilidad de no tomar ninguna instancia después de N veces es:

   $(1 - \frac{1}{N})^N \approx e^{-1} = 0.368$

   Esto significa que el conjunto de datos de entrenamiento contiene aproximadamente el 63.2 % de las instancias, que es lo mismo, que el sistema no es entrenado con el 36.8 % de los datos

2.5.2. Métodos de Medición de Error

Para analizar los errores generados a partir de un modelo de clasificación se emplea según[Alp04] :

1. *Matriz de Confusión.* Es una herramienta de visualización que se emplea en aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real, como se muestra en la tabla 2.2. De igual forma, uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo dos clases.

	Clase a predecir	
Clase Real	SI	NO
SI	TP: Verdaderos Positivos	FN:Falsos Negativos
NO	FP: Falsos Positivos	TN: Verdaderos Negativos

Cuadro 2.2: Matriz de Confusión

De la tabla anterior se extraen las siguientes medidas:

a) Tasa de Error: $Error = \#errores/\#ejemplo = \frac{|FN|+|FP|}{N}$

Donde

N= |FN|+|FP|+|TN|+|FN| es el total de ejemplos del conjunto de validación

b) Exactitud (Accuracy). Es la proporción del número total de predicciones que son correctas.

$Accuracy = \frac{|TP|+|TN|}{|FP|+|FN|+|TP|+|TN|}$

c) Recall. Es la proporción de casos postivos que fueron identificados correctamente.

$Recall = \frac{|TP|}{|TP|+|FN|}$

d) Precisión (precisión). Es la predicción de casos positivos que fueron clasificados correctamente.

$precision = \frac{|TP|}{|TP|+|FP|}$

e) Tasa de Falsos Negativos (FN Rate). Es la proporción de casos positivos que son incorrectamente clasificados como negativos.

$$FN = \frac{|FN|}{|TP|+|FN|}$$

f) Tasa de Falsos Positivos (FP Rate). Es la proporción de casos negativos que son incorrectamente clasificados como positivos.

$$FP = \frac{|FP|}{|TN|+|FP|}$$

2. *Curvas ROC(Receiver Operating Characteristic)*. Es una representación gráfica de la sensibilidad de los verdaderos positivos (TP/TP+FN) vs. los falsos positivos (FP/FP+TN). El punto (0,1) se llama una clasificación perfecta. La línea diagonal que divide el espacio de la ROC en áreas de la clasificación buena o mala. Los puntos por encima de la línea diagonal indican buenos resultados de clasificación, mientras que los puntos por debajo de la línea indican resultados equivocados (aunque el método de predicción puede simplemente ser invertido para conseguir puntos por encima de la línea). En la figura 2.10 se presenta una típica curva ROC.

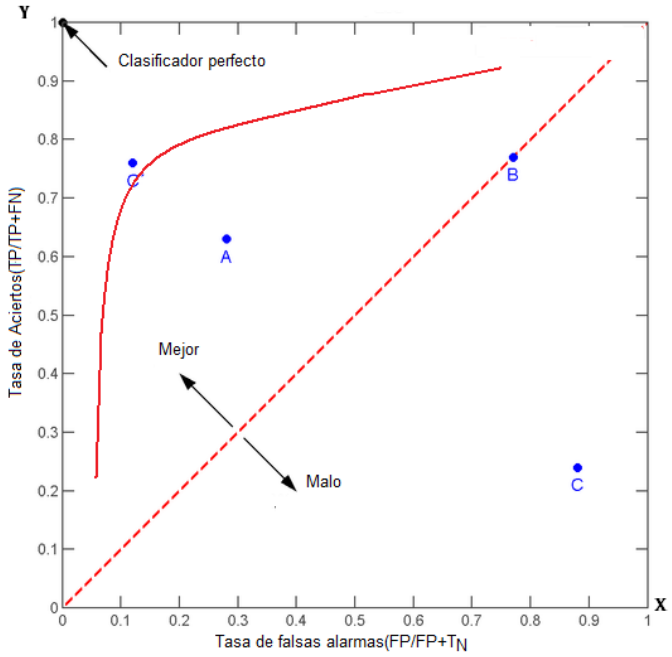

Figura 2.10: Típica curva ROC

2.5.3. Evaluación estadística

Una vez que un modelo se ha construido, es necesario evaluar su rendimiento respecto a otros modelos de clasificación a fin de determinar si el modelo propuesto es competitivo con los demás, para ello según[Alp04] las técnicas más usadas son:

1. *Prueba t por parejas (Pairwise t-Test).* Una prueba "t-test" es cualquier prueba de hipótesis estadística en la que el estadístico de prueba sigue una distribución t-Student. Para ello se parte de una hipótesis nula. En este escenario, se tiene un indicador de una variable aleatoria $x_i^t = 1$, si una instancia del conjunto de datos en el "fold" i es mal clasificada (i = 1, 2,... K). Entonces, la tasa de error para el "fold" i es:

   $p_i = \frac{\sum_{t=1}^{N} x_i^t}{N}$

   Donde N es el número de ejemplos en el "fold" i. Aceptamos la prueba si tenemos un error p_0 menor o igual a $t_{\alpha,K-1}$, esto es:

   $\frac{\sqrt{K}(m-p_0)}{S} \sim t_{K-1} \leqslant t_{\alpha,K-1}$

   Donde K es el número de "fold" (Típicamente K toma el valor de 10 o 30), m es la media de error y S es la desviación estándar de error.

   $m = \frac{\sum_{i=1}^{K} p_i}{K}$

   $S = \frac{\sum_{i=1}^{K}(p_i-m)^2}{K-1}$

2. *Análisis de varianza (ANOVA).* Proporciona una prueba estadística donde se tiene como hipótesis nula que las medias de varios grupos son iguales, y por lo tanto se generaliza una prueba t-Student para más de dos grupos. Para evaluar el modelo de clasificación es necesario evaluar sus errores de clasificación. En este caso tenemos una hipótesis nula igual a :

   $H_o = \mu_1 = \mu_2 = ...\mu_L$

   Donde μ_i es la media de error y L es el número de algoritmos (i.e modelos de clasificación). Luego tenemos las tasas de error para L algoritmos y K observaciones:

$X_{ij} \sim N(\mu_j, \sigma^2), j = 1, ..., L, i = 1, ..., K$

El enfoque en el análisis de varianza, requiere dos estimadores de σ^2. Un estimador es diseñado de tal forma que es verdad sólo cuando H_o es verdadero, y el segundo estimador es válido, sin importar si H_o es verdad o no. El primer estimador esta dado por :

$\tilde{\sigma}^2 = K \sum_{j=1}^{L} \frac{(m_j - m)^2}{L-1}$

y el segundo estimador, es el promedio del grupo de varianzas(S_j^2), definido como:

$\tilde{\sigma}^2 = \sum_{j=1}^{L} \frac{S_j^2}{L} = \sum_j \sum_i \frac{(X_{ij} - m_j)^2}{L(K-1)}$

Donde m y m_j son la media de error general y media de error para el algoritmo j respectivo.

Después de eso, es necesario construir la suma de cuadrados entre grupos (SSB) y la suma de cuadrados dentro del grupo (SSW)[Alp04], donde:

$\frac{SS_b}{\sigma^2} \sim X^2_{L-1}$ y $\frac{SS_w}{\sigma^2} \sim X^2_{L(K-1)}$

Por último, de acuerdo con [Alp04], la relación entre dos variables aleatorias independientes (que se distribuyen siguiendo una distribución un chi-cuadrado) dividido por sus respectivos grados de libertad es una variable aleatoria que tiene una distribución F:

$\left(\frac{SS_b/\sigma^2}{L-1}\right) / \left(\frac{SS_w/\sigma^2}{L(K-1)}\right) = \frac{SS_b/(L-1)}{SS_w/(L(K-1))} \sim F_{L-1,L(K-1)}$

Aceptamos la hipótesis nula H_o (es decir, todos las medias son iguales) si, dado un valor de significancia α, la tasa de error esperado es menor que $F_{\alpha,L-1,L(K-1)}$. Rechazamos H_o , si los dos estimadores, tienen diferencias significativas entre los L grupos

2.6. Resumen del Capítulo

Este capítulo presentó las bases teóricas sobre las cuales se desarrollo el trabajo de investigación. El capítulo inicio planteando el problema que se tiene en la detección de fraude y la forma como las entidades en el mundo, se han visto avocadas a emplear técnicas técnicas tradicionales (procesos estadísticos y conocimiento basado en reglas SQL) y técnicas de minería de datos para encontrar patrones de fraude en base a la información almacenada. Para ello

se hizo una recopilación de la literatura más sobresaliente en el tema. Finalmente se termina abordando con detalle el proceso de aprendizaje mediante redes neuronales como una técnica de solución al problema de detección de fraude, acompañada de las técnicas de extracción de reglas a partir de la red neuronal entrenada, que permiten una mejor comprensión de los resultados obtenidos por parte de los expertos.

Capítulo 3

Modelo de Detección de Fraude

En este capítulo se presenta el diseño del modelo de detección de fraude, basado en la clasificación binaria de transacciones normales e inusuales usando una red neuronal y en el descubrimiento simbólico de reglas de clasificación, provenientes de la red neuronal previamente entrenada, con el fin de caracterizar patrones para la detección de fraude que ayuden al experto del negocio a examinar y verificar más fácilmente los resultados obtenidos para apoyar la toma de decisiones.

El capítulo comienza planteando el problema que se tiene en la detección de fraude, luego se hace la descripción del modelo propuesto de detección, el cual no pretende cubrir todos los aspectos involucrados en un proceso de KDD, sino solo aquellos que permitan representar el conocimiento a extraer en términos de reglas de clasificación a partir de una red neuronal. De esta manera, el modelo presenta un conjunto de fases que permiten la preparación de datos, extracción y presentación del conocimiento, comunes en cualquier proceso de minería de datos. El capítulo hace énfasis en el proceso de aprendizaje mediante una red neuronal "MultiLayer Perceptron" y en el proceso de extracción de reglas a través de una descripción detallada del algoritmo "M of N".

De igual forma el detalle de la implementación del modelo se explica en los Anexos A y B, que presentan los diferentes modelos construidos en "RapidMiner" para el proceso de entre-

namiento y clasificación de la Red Neuronal, al igual que el modelo de extracción de reglas; para este último, se hizo la implementación del algoritmo "M of N" como un operador más de aprendizaje de la herramienta "RapidMiner".

3.1. Modelo de extracción de conocimiento para la detección de fraude: Visión General

El problema del fraude ha representado cuantiosas pérdidas a lo largo de la historia. La mayoría de las entidades financieras, a nivel mundial, han adoptado estrategias para reducir el riesgo de fraude, estrategias entre las cuales encontramos la utilización de modernos sistemas de detección para determinar, por medio de una calificación probabilística o la aplicación de reglas, cuándo una transacción es fraudulenta. Muchos de estos sistemas implementan modelos de clasificación y predicción a través de técnicas de minería de datos como redes neuronales, con las cuales se logra una alta precisión [KLSH04],[STD07, SZC02], sin embargo sus mayores críticas son: el tiempo de aprendizaje resulta costoso y su forma de clasificación, que es una caja negra, y en muchas aplicaciones es altamente deseable el poder extraer las reglas de clasificación, con el fin de que expertos interpreten y verifiquen fácilmente los resultados obtenidos.

Este trabajo plantea un modelo de detección de fraude, basado en el análisis de datos utilizando técnicas de minería de datos, específicamente para un conjunto de transacciones de una franquicia en particular. El modelo propuesto se presenta en la figura 3.1, la cual presenta a grandes rasgos la interacción de los diferentes módulos y la comunicación con elementos externos al sistema (Bases de Datos y Archivos). El modelo se compone de varios módulos que proporcionan una funcionalidad específica, con el fin de obtener el conocimiento oculto que hay en el conjunto de datos. Los módulos en mención son:

1. Preparación y selección de datos: Encargado de la conexión con la fuente de información, la selección y la construcción del conjunto de datos para la extracción del conocimiento.

2. Extracción de conocimiento. Es el Motor del modelo de detección propuesto, compuesto por:

   a) Algoritmo de aprendizaje. Para el proceso se construye y entrena una red neuronal "MultilayerPerceptron". El modelo de la red se construye en base al número de atributos y clases del conjunto de datos de entrenamiento, sobre el cual, se aplica el algoritmo "backpropagation" para obtener el modelo de Red Neuronal Entrenada que se valida contra todo el conjunto de datos.

   b) Adaptación del algoritmo de extracción de reglas. En esta fase se hace la extracción simbólica de las reglas de clasificación de la red neuronal previamente entrenada. Las reglas extraídas son la de forma: SI(M de los siguientes N antecedentes son verdaderos) ENTONCES...

3. Presentación de resultados. Encargado de la interacción con el usuario y la presentación de los resultados finales del proceso de aprendizaje.

4. Predicción: Encargado de inferir una predicción sobre un conjunto de datos nuevos. Para ello se parte del conocimiento encontrado para hacer la clasificación del conjunto de datos.

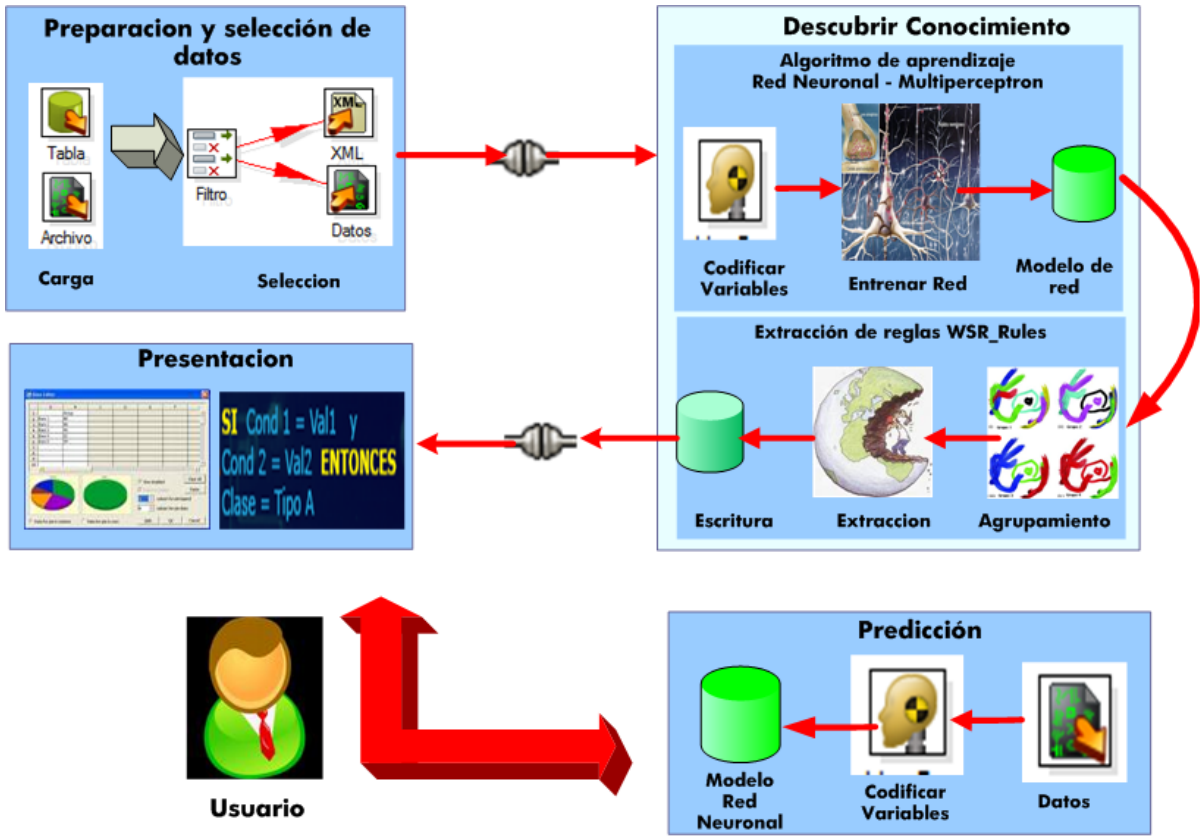

Figure 3.1: Modelo de extracción de conocimiento para la detección de fraude

3.2. Preparación y selección de datos

En esta fase se determinan, seleccionan y extraen los datos de la fuente de información para el proceso de extracción de conocimiento. En esta etapa se realizan las siguientes actividades:

1. *Extracción de datos relevantes de la fuente de información* haciendo énfasis en:

   a) Identificación de los datos relevantes en las fuentes de información (para el caso, un modelo relacional de tablas)

   b) Extracción e integración. A pesar de que los aspectos importantes a evaluar del negocio pueden estar distribuidos en diferentes tablas en una misma base de datos, en el modelo propuesto, se considera que estos datos conforman una sola tabla, a

partir de una consulta en la base de datos que retorna una tabla virtual (conjunto de registros), con los datos a utilizar en el proceso de extracción de conocimiento.

2. *Preparación de datos*, en esta parte se hace énfasis en:

   a) Eliminación de registros inconsistentes.

   b) Eliminación de atributos que no aportan información, por ejemplo redundancia de información, valores perdidos,etc.

   c) Ruido e inconsistencia de datos.

   d) Discretización de atributos.

3. *Selección de la información* a utilizar en el proceso de extracción de conocimiento, a partir de determinar cuales van hacer las variables independientes y cual la variable objetivo (conclusión). Para esto, el usuario es el encargado de determinar las variables a considerar en el proceso de aprendizaje y extracción de reglas.

Esta fase puede ser soportada por herramientas estadísticas, herramientas de limpieza y otras, de tal manera que el usuario pueda realizar estas operaciones de una forma eficiente.

3.3. Algoritmo de extracción de conocimiento

Como se mencionó en los capítulos anteriores, las redes neuronales producen resultados que son difíciles de entender para un dominio dado, entonces ¿por qué se aplican redes neuronales para estos problemas?. En su lugar, ¿por qué no usar métodos de aprendizaje que producen modelos más compresibles al ser humano?. La respuesta a estas preguntas, es que, para algunos problemas, las redes neuronales tienen un sesgo inductivo más adecuado que los demás algoritmos de aprendizaje.

En algunos casos, las redes neuronales tienen un sesgo más apropiado que restringe el espacio de hipótesis en un grado mayor que otros algoritmos de aprendizaje. Por ejemplo, el método "Q-Learning" por refuerzo de aprendizaje (Watkins, 1989)[Cra96]requiere que se muestren

hipótesis a las funciones definidas y que se actualicen después de cada ejemplo de entrenamiento. Las tareas de predicción secuencial, facilitan otro tipo de problema, para el cual las redes neuronales proporcionan un espacio de hipótesis más adecuado.

De igual forma, varios estudios empíricos han señalado que hay algunos problemas de aprendizaje, en el cual las redes neuronales ofrecen una precisión superior con respecto a otros algoritmos de aprendizaje simbólico (Fisher y McKusick[Cra96]). En los últimos años han sido aplicadas exitosamente a problemas de aprendizaje del mundo real, en especial al problema de detección de fraude (Efstathios Kirkos[KSM07]), el algoritmo de propagación inversa ("backpropagation") ha sido el más utilizado de los actuales sistemas de redes neuronales, dado su buen desempeño.

El trabajo en esta tesis se centra en redes neuronales "MultilayerPerceptron" aplicadas a tareas de clasificación (las demás arquitecturas y métodos de aprendizaje no serán tenidos en cuenta), y por lo tanto la discusión que sigue es restringida a este tipo de enfoque de red neuronal.

3.3.1. Algoritmo de Propagación Inversa (BackPropagation)

Es un algoritmo iterativo que permite entrenar redes multicapa, propuesto por Rumerlhart en 1986[Fu94a], su funcionamiento es el siguiente:

1. Inicialización de pesos. Se debe fijar unos pesos iniciales a todos los nodos Multilayer-Perceptrone de la capa de entrada aleatoriamente.

2. Calcular los valores de activación de los nodos.

   a) El nivel de activación del valor de entrada es determinado por una instancia presentada a la red.

   b) El nivel de activación O_j de los nodos de la capa oculta y de salida es determinado por $O_j = f(\sum W_{ij}O_i - \theta_j)$, donde:

   - W_{ij} son los pesos de entrada al nodo
   - O_j-θ_j es el umbral del nodo,

- f es la función sigmoide:

$$f(a) = \frac{1}{1 + e^{-a}}$$

3. Calcular los pesos de entrenamiento

   a) Comenzar con los nodos de salida hacia los nodos de la capa oculta recursivamente. Ajustar los pesos de acuerdo con: $W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji}$, donde

   - $W_{ji}(t)$ es el peso del nodo$i$ al nodo$j$ en el tiempo t (o i-esima iteración)
   - ΔW_{ji} es el peso ajustado

   b) El peso cambiado se computa por $\Delta W_{ji} = \eta \delta_j O_i$, donde:

   - η es la rata de aprendizaje $(0 < \eta < 1)$
   - δ_j es el error de gradiente del nodo j.
   - La convergencia es mas rápida por la adición del término de momentum:

$$W_{ji}(t+1) = W_{ji}(t) + \eta \delta_j O_j + \alpha [W_{ji}(t)] - W_{ji}(t-1)]$$

     Donde $0 < \alpha < 1$

   c) El error del gradiente está dado por:

   - Nodos de salida: $\delta_j = O_j(1 - O_j)(T_j - O_j)$, donde $T_j$es la etiqueta de la salida esperada y O_j es el actual valor de activación del nodo de salida j.
   - Nodos de la capa oculta: $\delta_j = O_j(1 - O_j) \sum_k \delta_k W_{kj}$, donde δ_k es el error del gradiente de la unidad k para la cual hay una conexión desde el nodo de la capa oculta j.

4. Repetir el ciclo hasta converger en términos del criterio de selección de error. Un ciclo incluye: la presentación de una instancia, calcular los valores de activación y modificar los pesos.

3.4. Algoritmo de extracción simbólica de reglas

Dada la forma de clasificación de una red neuronal, como una caja negra, entonces ¿Cómo hacer más entendible a un experto los resultados de esta clasificación?. La respuesta a esta pregunta es que para los seres humanos nos es más fácil tener un conjunto de reglas que permitan explicar el proceso de clasificación. Durante los últimos 15 años han aparecido algoritmos y metodologías que permiten hacer la extracción de reglas a partir de una red neuronal entrenada, dada una estructura particular de ésta.

De la gran variedad de algoritmos y métodos de extracción realizados a la fecha, el trabajo en esta tesis se centró en el algoritmo "M of N" propuesto por Towell (1991)[TSN90, MHRT06] y por lo tanto la discusión que sigue es restringida a este tipo de algoritmo, para ello se hizo como aporte a los trabajos existentes, una recopilación de diferentes fuentes a fin de tener un algoritmo detallado que permita comprender el funcionamiento del proceso de extracción de reglas a partir de una red neuronal.

3.4.1. Algoritmo M of N

Es un algoritmo descomposicional que permite extraer reglas a partir de una red multicapa, su funcionamiento es el siguiente:

1. *Agrupamiento.* En esta etapa se construyen grupos de pesos similares de entrada a cada neurona, empleando un algoritmo estándar de agrupación denominado "Join Algorithm" propuesto por Hartigan en 1975. Este método opera sucesivamente combinando dos valores cercanos y tomando como valor su promedio. El proceso inicia teniendo cada grupo con un solo peso. La agrupación se detiene cuando no encuentra un par de grupos menor a una distancia definida, en la literatura este parámetro es fijado entre 0.25 y 0.5 [MHRT06]. La medida para calcular la distancia entre los grupos se basa en la medida de la "distancia Euclidiana". Resultado del proceso de agrupamiento se va a tener un conjunto de grupos expresados de la siguiente forma:

$S = \{C_1, C_2, ..., C_m\}$

$C_k = \{W_{v1}, W_{v2}, ..., W_{vl}\}$

Donde $C_k$denota un grupo con pesos $w_{v1}, w_{v2}, ..., w_{vl}$

En la figura 3.2, se detalla el diagrama de flujo del proceso de agrupamiento.

2. *Extraer Reglas.* Esta etapa comienza identificando qué grupos de pesos no tienen efecto sobre la entrada total de activación a cada neurona de la red. Los grupos que no exceden el bias de la neurona son eliminados (este proceso es muy similar al algoritmo SUBSET). La entrada a una neurona en la capa oculta y de salida esta dada por:

   $net_i = \sum_{j=1}^{n} O_j * w_{ij} + \theta_i$

   Donde

   - w_{ij} es el peso de la neurona j a la neurona i,
   - O_j es la activación de la neurona j y
   - $\theta_i$es el bias de la neurona i.

   De igual forma, se asume que la neurona tiene una máxima activación (i.e valor cercano a uno) o inactivación (valor cercano a cero), a partir de este supuesto, cada entrada a la neurona puede interpretarse como una función de paso o regla booleana. Así, el problema de extracción de reglas se reduce a determinar la situación en la cual la regla es verdadera. Esto simplifica la ecuación anterior a:

   $net_i = \sum_{j=1}^{n} w_{ij} - \theta_i$

   Donde un signo menos indica una noción de semántica del umbral. Dado lo anterior, la extacción de reglas se limita a buscar para cada neurona un conjunto de enlaces cuya suma de pesos excedan el bias de la neurona.

   El algoritmo de generación de reglas divide los grupos hallados S en dos sub conjuntos S_py S_n definidos de la siguiente forma:

   $S_p = \{C_k | C_k \epsilon S \wedge wc_k \geqslant 0\}$

$S_n = \{C_k | C_k \epsilon S \wedge wc_k < 0\}$

Donde

- S_p es el subconjunto de pesos positivos
- $S_n$es el subconjunto de pesos negativos y
- wc_k es la suma del grupo C_k.

Durante la generación de reglas se buscan grupos de pesos positivos $P\epsilon\beta(S_p)$ y grupos de pesos negativos $N\epsilon\beta(S_N)$. Finalmente, dado α como la entrada de activación al conjunto de grupos que se puede enviar, el valor máximo y mínimo de activación está dado por:

$\alpha_{min} = \sum_{j=1}^{n} min\{w_j, 0\}$

$\alpha_{max} = \sum_{j=1}^{n} max\{w_j, 0\}$

Donde $w_1, w_2, ..., w_n$ son los pesos de entrada a la neurona.

Para cada neurona de la capa oculta y de salida se debe hacer:

a) Si $\alpha_{max} \leq \theta$,eliminar todos los grupos

b) Extraer un conjunto de grupos positivos $P\epsilon\beta(S_p)$ con la condición $w_p > \theta$, donde w_p es la suma de los pesos de P

c) Para cada conjunto de datos P encontrado:

1) Computar la entrada de activación $\alpha = \alpha_{min} + w_p$
2) Si $\alpha > \theta$entonces construir reglas para P y continuar con el siguiente conjunto
3) Extraer un conjunto de grupos negativos $N\epsilon\beta(S_n)$ con la condición $w_n < \alpha - \theta$, donde w_n es la suma de los pesos de N
4) Con cada grupo N crear una regla para P y N

En la figura 3.3, se detalla el diagrama de flujo para la fase de extracción de reglas, en términos de las diferentes neuronas de la capa oculta y de salida.

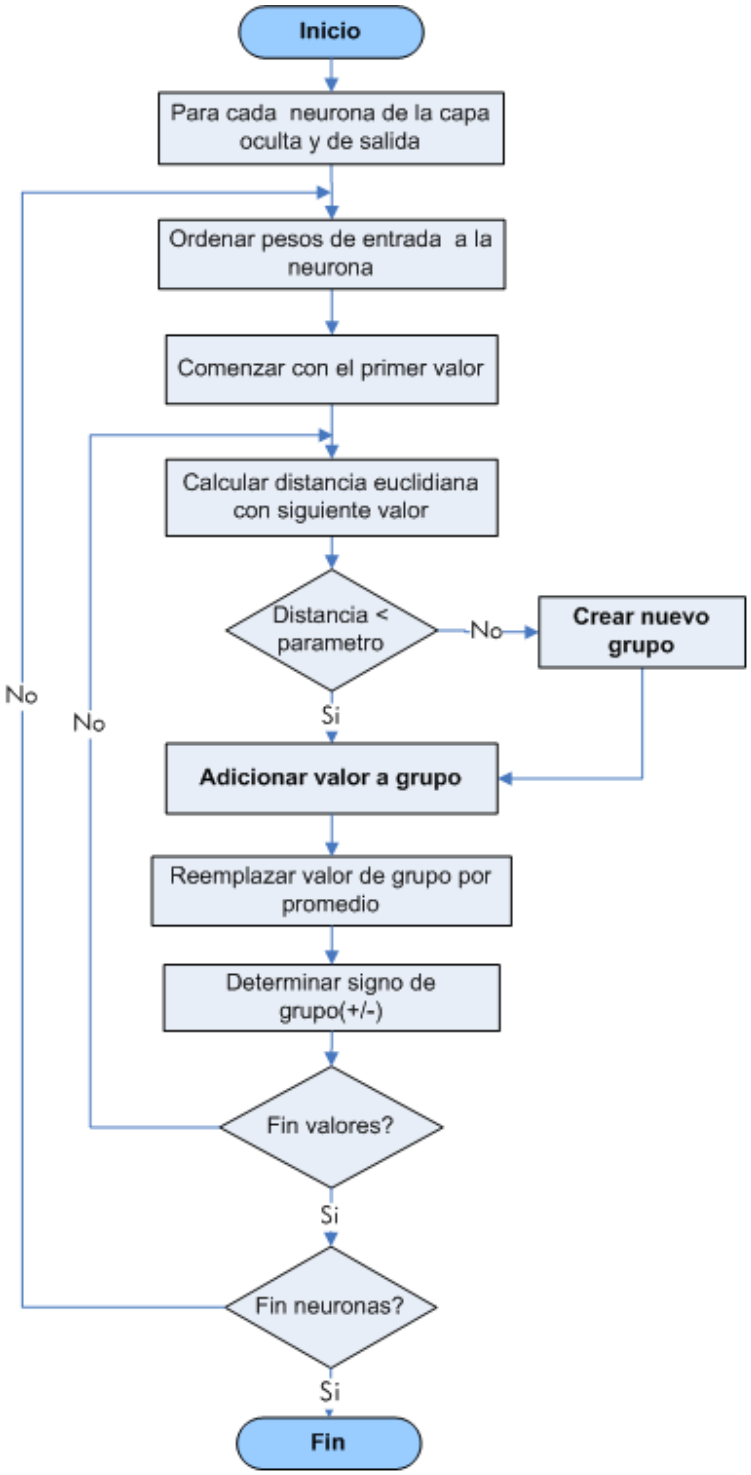

Figure 3.2: Etapa de agrupamiento - Algoritmo M of N

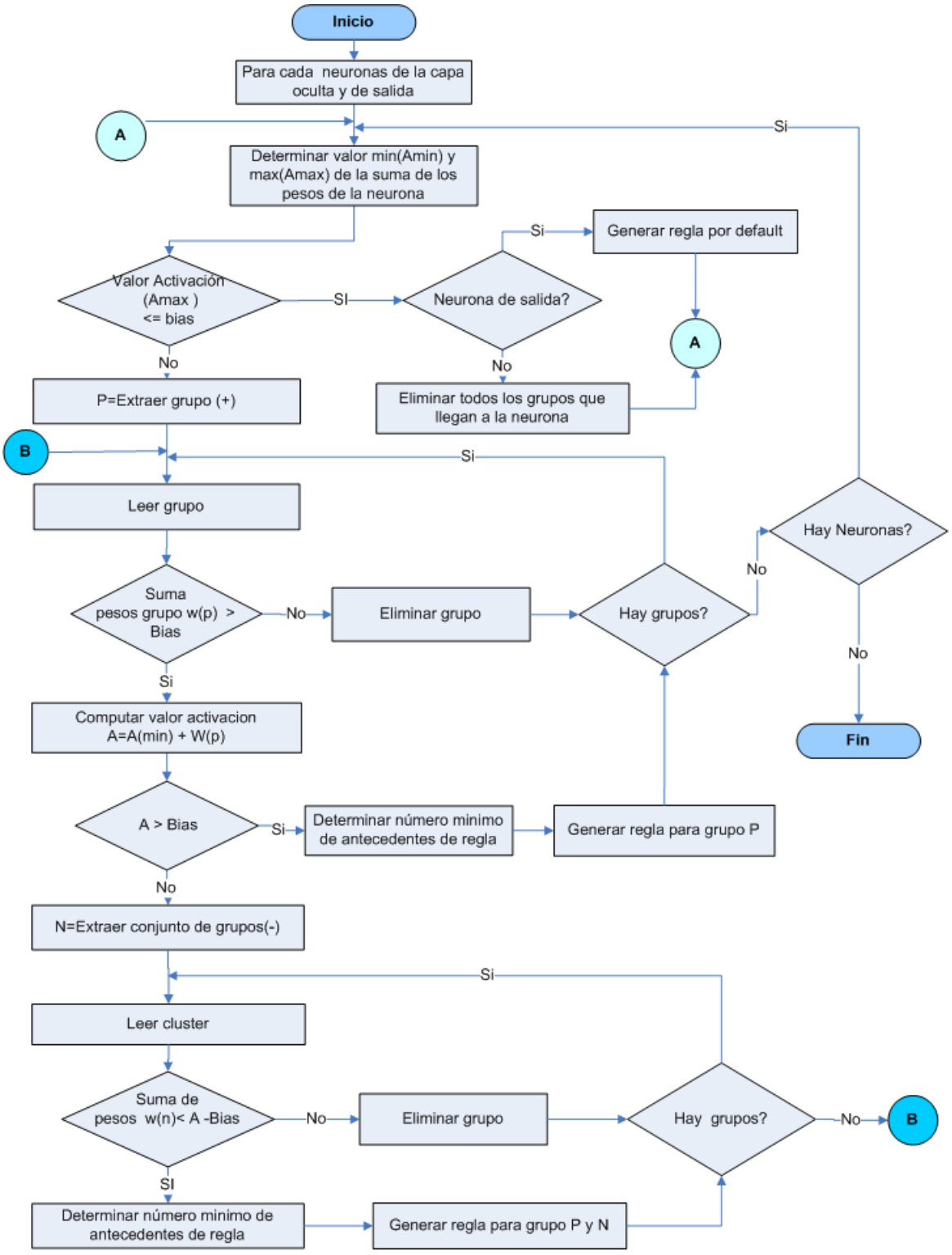

Figure 3.3: Etapa de extracción de reglas - Algoritmo M of N

3. *Agrupar Reglas.* En esta fase del algoritmo se forman reglas que re-expresan la red en términos de entradas y salidas, para ello se translada el bias y los pesos de entrada a cada neurona en reglas con pesos antecedentes que hacen verdadera la regla, si la suma de los pesos exceden el bias.

   Por ejemplo, considere una neurona de salida Z con seis entradas desde las neuronas A, B, C, D, E y F, y asuma que las entradas de la neuronas C y E han sido eliminadas durante la etapa de generación de reglas, así las reglas obtenidas son:

   a) SI *bias* < *promedio-grupo* * *número_aciertos*(B,F) ENTONCES Z

   b) SI *bias* < *promedio-grupo* * *número_aciertos*(B,F) +
   promedio-grupo2 * *número_aciertos*(D) ENTONCES Z

   En la figura 3.4, se detalla el diagrama de flujo para la fase de agrupacón de reglas en términos de entradas y salidas.

4. *Podar Reglas.* En esta fase se pretende eliminar reglas redundantes y generar reglas más generales que las existentes, para ello se re-escriben las premisas de las reglas actuales en la forma $(x_1, x_2, ..., x_n) \in \{H, L, -\}^n$, Donde $x_1, x_2, ..., x_n$ denota las entradas. El estado H significa que la entrada tiene un valor de activación, L la entrada tiene un valor de inactivación y$-$ la entrada es ignorada. Una vez hecha la poda, las reglas resultantes son escritas en téminos de las premisas de entrada original.

   Por ejemplo, dada las siguientes reglas expresada en téminos de H,L,-:

   a) *S*i (-,L,-,-,-,L) *Entonces* Z

   b) *S*i (H,L,-,-,-,L) *Entonces* Z

   c) *S*i (-,L,-,H,-,L) *Entonces* Z

   De las reglas anteriores se observa que la primera reglas es más general que las otras dos, lo cual ocasiona que las reglas con los patrones (H,L,-,-,-,L) y (-,L,-,H,-,L) sean podadas dado que el patrón (-,L,-,-,-,L) de la primera regla es más general.

   En la figura 3.5, se detalla el proceso de poda de las reglas.

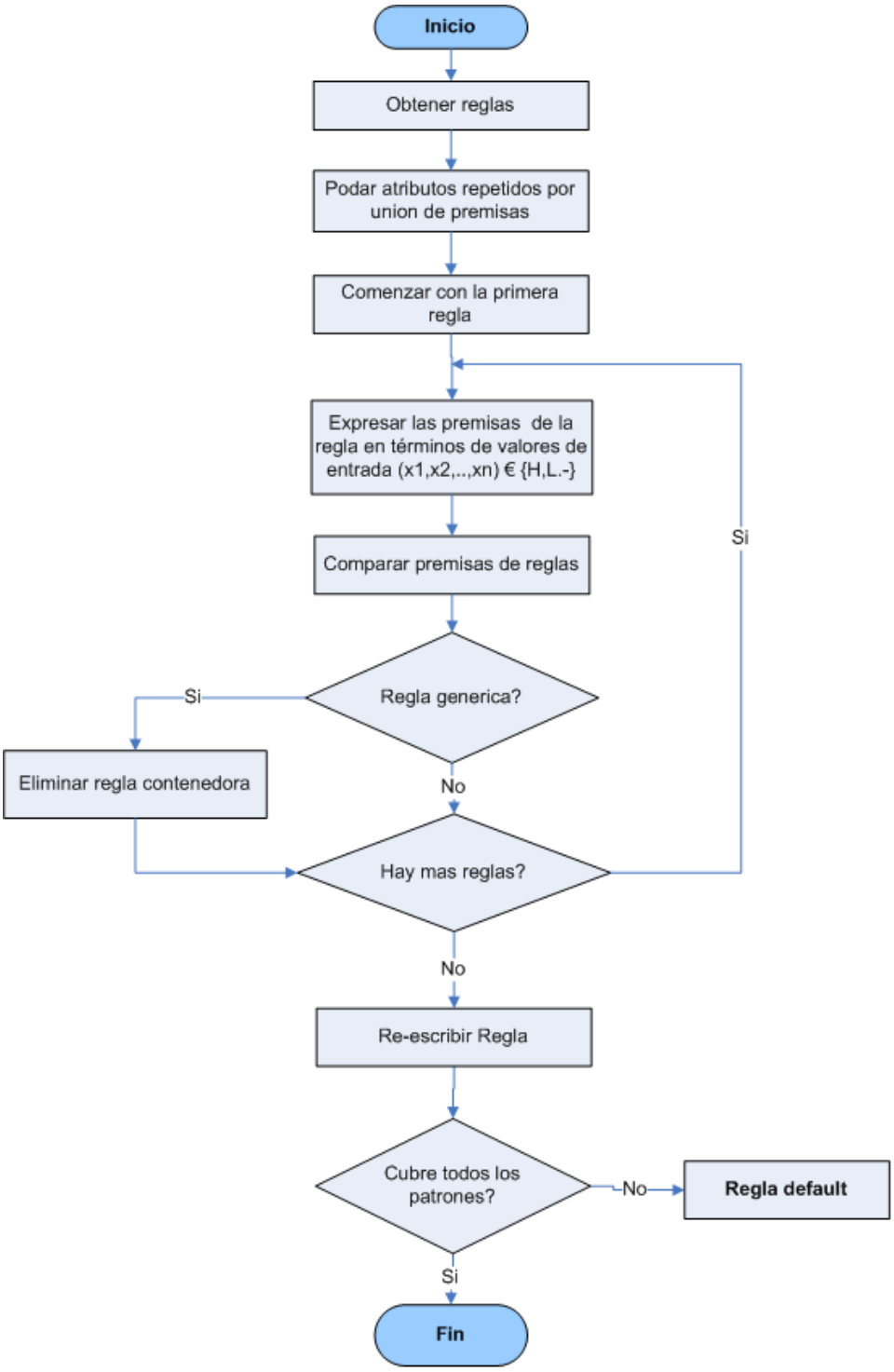

Figure 3.4: Etapa de poda de reglas - Algoritmo M of N

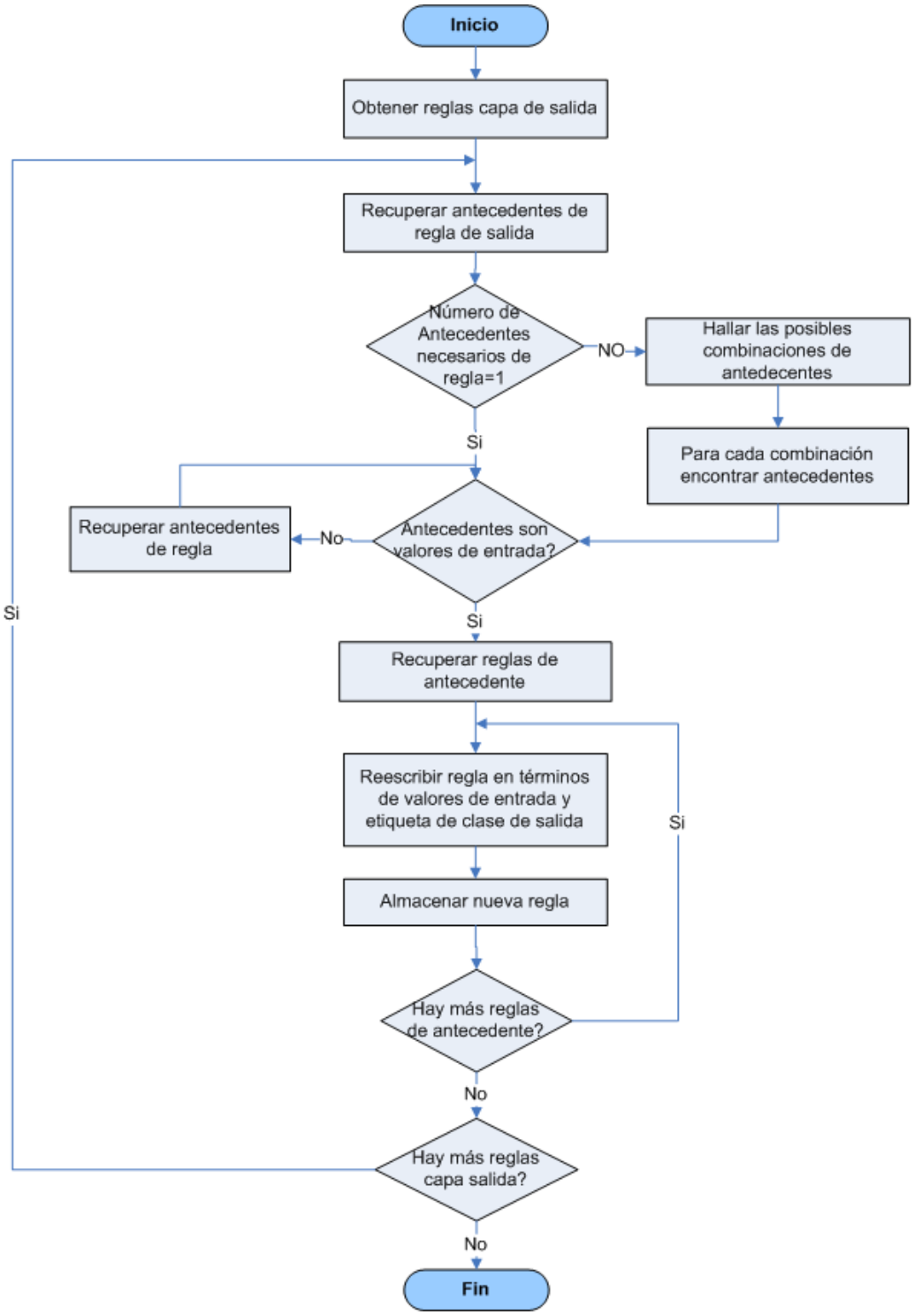

Figure 3.5: Etapa de agrupación de reglas - Algoritmo M of N

3.4.2. Lenguaje de especificación de conocimiento a descubrir

Aunque puede resultar evidente, el modelo usa un subconjunto del lenguaje de la lógica matemática para representar y expresar el conocimiento que se está descubriendo. Este lenguaje subconjunto está conformado por:

1. Un conjunto de variables independientes (atributos propios del conjunto de datos seleccionados para el proceso de extracción de conocimiento),
2. Una clase o variable especial (conclusión),
3. Una colección de conjuntos lógicos (símbolos lingüísticos proporcionados por la lógica matemática) y
4. Un conjunto de reglas de la forma : SI(M de los siguientes N *antecedentes* son verdaderos) ENTONCES <*conclusión*> Donde, *antecedentes* es cualquier expresión lógica construida a partir de expresiones atómicas sobre las variables dependientes y conclusión es un conjunto definido sobre la variable conclusión.

3.5. Presentación y Evaluación

La última fase y la más importante para el usuario es la visualización de los resultados. Todas las herramientas de minería cuentan con reportes escritos y con gráficas para indicar al usuario los puntos en donde se encontró algo interesante. Muchas veces los usuarios se enamoran de una herramienta por los gráficos que despliegan.

Las mejores gráficas que una herramienta puede mostrar son aquellas que el usuario entiende. Eso no quiere decir que las gráficas animadas y con mucho colorido no sean buenas, simplemente que los usuarios muchas veces no tienen los conocimientos necesarios sobre el tema al que realizaron minería por lo que no pueden interpretar los resultados; no pueden definir si los resultados arrojados son buenos o son malos para la organización. Por otro lado, los reportes deben de ser lo más sencillos posibles, de tal manera que el usuario los pueda interpretar sin

confundirse entre una gran cantidad de datos. El tipo de reporte depende en gran medida del tipo de técnica empleada para analizar los datos, por lo que también en este aspecto el usuario debe aprender a interpretar los reportes producidos por cada una de ellas, independientemente del conocimiento que tenga sobre el tema de análisis.

Por lo anterior el conocimiento obtenido será representado a través de dos estilos diferentes, un reporte escrito en el que el usuario podrá visualizar las respectivas reglas de inducción encontradas y un pastel de distribución de las reglas de inducción encontradas, de igual forma se presenta el desempeño que presentó la red neuronal.

Adicionalmente se debe proveer una herramienta que permita hacer predicción (clasificar) con un conjunto de datos nuevo. Para esto se implementa un modelo de inferencia que con base en el conocimiento encontrado permita hacer la validación del conjunto de datos.

3.6. Resumen del Capítulo

Este capítulo presentó el modelo de detección de fraude propuesto, sobre el cual se validará el conjunto de datos de estudio del presente trabajo. Para ello se describieron las etapas que incluye éste, como:

1. Selección y preparación de datos
2. Extracción de conocimiento
3. Presentación de resultados
4. Predicción

De igual forma, se resaltó la parte de extracción de conocimiento, el cual es el ”motor” de todo el modelo de detección propuesto. Para ello se definió:

1. Una etapa de clasificación, empleando una red neuronal ”MultilayerPerceptron”, con base en el número de atributos y clases del conjunto de datos de entrenamiento, sobre

el cual, se aplica el algoritmo ”backpropagation” para obtener el modelo de red neuronal entrenada que se valida contra todo el conjunto de datos y que es entrada para el proceso de extracción de reglas.

2. Una etapa de extracción simbólica de reglas a partir del modelo de red entrenado, para ello, se utiliza el algoritmo ”M of N” propuesto por Towell (1991)[TSN90]. De igual forma como aporte, se hace una descripción detallada de este algoritmo para cada una de las diferentes etapas, permitiendo comprender mejor el funcionamiento del proceso de extracción de reglas a partir de la red neuronal entrenada.

Capítulo 4

Validación Experimental

En este capítulo se describe el conjunto de datos sobre el cual se efectúa la validación del modelo. Para ello se hace un proceso de limpieza y pre-procesamiento en los datos de origen para generar una muestra con la cual se adelanta la experimentación del modelo. Dado el bajo número de casos inusuales en el conjunto de datos, se emplea un muestreo estratificado para los siguientes escenarios:

1. Muestra del 10 % del total de los datos.
2. Muestra que realza los casos inusuales en un proporción 50-50 y 56-44.

A partir de estos escenarios, se evalúa el modelo de detección de fraude propuesto en el contexto del proceso de clasificación y validación de la red neuronal y el proceso de extracción de reglas. El propósito de los experimentos, es evaluar la exactitud de clasificación para casos inusuales y rendimiento del modelo de red propuesto respecto a otras técnicas como árboles de decisión y Naive Bayes. Para ello se emplea el análisis de matrices de confusión y curvas ROC.

De igual forma, se evalúan las reglas extraídas a partir de la red neuronal en términos de su confianza y número, con otras técnicas como árboles de decisión y reglas de asociación.

4.1. Descripción: Conjunto de datos

En Colombia, "XYZ Limitada" es una organización dedicada a la prestación de servicios relacionados con el intercambio de divisas así como el envío y pago de remesas. Esta compañía cuenta con una solución que realiza el proceso de validación de datos que vienen en la transacción, con el objetivo de prevenir el lavado de activos y/o financiación del terrorismo. A la fecha se cuenta con controles que implementan las tipologías más comunes, entre las que se destacan, transferencias fraccionadas de dinero (pitufeo), tope de operaciones, remesas de un mismo ordenante para diferentes beneficiarios y viceversa, control de lista de negativos como la OFAC, BOE, Comunidad Europea y Naciones Unidas.

A continuación se describe, en forma resumida, el proceso que se ejecuta al momento de que una persona realice una de las cuatro grandes actividades involucradas en el envío de dinero (pago, envío, compra y venta de divisas):

1. Una persona puede acudir a una de las oficinas de "XYZ Limitada" con el objetivo de realizar una transacción bien sea para recibir dinero proveniente del exterior y/o hacer un giro internacional, al igual que la compra o venta de divisas.

2. En las instalaciones dicha persona es atendida por uno de los empleados quien se encarga de registrar la información de la transacción a realizar. Dicha información incluye también los datos personales en caso de que la persona no exista en el sistema.

3. "XYZ Limitada" hace la verificación de la información tanto del individuo que desea realizar el proceso como de la transacción que se desea ejecutar.

4. Si la información del individuo de acuerdo a un conjunto de validaciones en listas de control e información básica presenta alguna anomalía, la transacción es detenida para ser revisada por el área de cumplimiento de la entidad.

5. Dependiendo de la revisión efectuada, se autoriza o rechaza el pago.

A través de este proceso se ha consolidado un gran conjunto de información. Dicha información incluye datos de transacciones exitosas y negadas. Básicamente se almacena información de:

- Clientes. Quienes realizan operaciones de envío, recibo, compra y venta de divisas. La información es detallada en cuanto a datos básicos personales, información financiera (actividad económica, ingresos, patrimonio) y datos complementarios.
- Transacciones realizadas por los clientes. La información es detalla en cuanto al monto de la operación, tipo de operación, gravámenes fiscales, lugar de recolección, datos del beneficiario y/o ordenante. Con los controles actuales las transacciones se clasifican en normales o inusuales

4.1.1. Selección de datos

A continuación se describe brevemente el proceso realizado para la selección de datos.

4.1.1.1. Integración de Datos

La información almacenada en el modelo de datos relacional (i.e. Clientes y Transacciones) se encuentra distribuida en cinco tablas de las cuales dos contienen la información de las transacciones (usuales e inusuales) y tres tablas contienen los detalles de cada cliente.

A partir de esta información se realiza un proceso de integración el cual resulta ser sencillo debido a que las tablas de transacciones usuales e inusuales poseen los mismos atributos. Respecto a la información del cliente se integran los datos básicos y su detalle. Posteriormente se crea una tabla única que contiene la información de los clientes y de las transacciones realizadas.

Después de haber realizado este proceso, se cuenta con 890.285 transacciones usuales, 20.788 transacciones inusuales para un total 911.073 transacciones. También se cuenta con 276.619 clientes de los cuales 233.553 poseen información parcialmente completa.

4.1.1.2. Análisis Preliminar

Inicialmente se selecciona la información de clientes y transacciones involucradas a partir del año 2006. A partir de los registros obtenidos se realiza un análisis preliminar con la

finalidad de eliminar algunos atributos (i.e. dimensiones). Este análisis se justifica en que existen atributos calculados a partir de otros, atributos redundantes (i.e. Código de ciudad y Nombre de ciudad), atributos incompletos que superen un 50 % del total (i.e. dirección y teléfono de la oficina de trabajo del cliente). Este proceso permitió reducir las dimensiones de 54 atributos a 28 atributos. Después de este proceso, la muestra se redujo a 534.247 registros. Los resultados obtenidos en esta etapa fueron:

1. Eliminación de registros inconsistentes. Se encontraron los siguientes casos:

   a) Montos de transacción inferiores a USD $ 10 y que la naturaleza de la transacción sea diferente a compra o venta de divisas ya que la comisión de estas transacciones son de USD $ 25;

   b) Nombre de clientes inconsistentes (i.e. 'XXX YYY', 'AA').

   c) Pagos declinados por caída del sistema (i.e. transacciones no terminadas).

2. Eliminación de atributos que no aportan información. Se encontraron:

   a) Algunos de los atributos poseen el mismo valor para todos los registros (i.e. Código de la transacción).

   b) Redundancia de información en dos atributos (i.e. el producto y subproducto es el mismo).

   c) Valores perdidos. El mayor problema se encuentra en el atributo que guarda los datos de "Origen de los Recursos" con un total de 15 % perdido.

3. Ruido e inconsistencia de datos. Existen atributos que poseen ruido (i.e. Fecha de nacimiento en donde al realizar cálculos de edad se encuentra que existen clientes con más de 100 años). Para estos casos se hace un análisis estadístico con el objetivo de determinar una edad apta; el estimador escogido fue la mediana (i.e. Se actualizan los atributos al valor de la mediana). Para el caso de la fecha de nacimiento el porcentaje de datos inconsistentes es aproximadamente el 8 %. Por otra parte, algunos clientes poseen

tipo de documento inconsistente respecto a su edad; para ello se revisó la relación del tipo de documento con la fecha de nacimiento (i.e. si es menor de edad y posee cédula).

4. Discretización de atributos. Varios de los atributos ya se encuentran discretizados y no es necesario realizar un trabajo exigente para cambiar valores. Dentro de estos atributos se consideran: el patrimonio, el ingreso, el monto de la transacción, la edad y los días de vinculación del cliente.

4.1.1.3. Preparación de datos

1. Valores perdidos. Para el atributo "Origen de los Recursos", se hizo un proceso de equivalencia con un nuevo atributo, donde se discretizó y completaron valores vacíos con la etiqueta 'OTROS'. Mediante este proceso se pasó de tener 12.806 valores diferentes a sólo 24.

2. Selección final de atributos. De los 28 atributos iniciales del análisis preliminar se dejaron 23 atributos, se descartaron atributos que no aportan al proceso de minería debido a su carácter informativo, algunos de estos atributos eliminados son: identificación, nombre, dirección y teléfono del cliente. Igualmente se adicionó un campo con el valor acumulado de las transacciones del cliente y otro campo con el número total de transacciones realizadas por un cliente específico discriminando por el tipo de producto. Estos dos atributos se incluyen con el fin de ver posibles prácticas de pitufeo. De los 23 atributos seleccionados, 14 son realmente útiles para el proceso de minería como se citan continuación: producto, departamento destino, sexo, estado civil, actividad económica, país origen de la transacción, uso de recursos, monto acumulado, edad, ingreso mensual, número de transacciones, días de vinculación, estado de la transacción, estado cliente. Tomando operaciones completamente distintas con los atributos finales seleccionados, el conjunto de registros se reduce a 157.200.

4.1.2. Análisis exploratorio

A continuación se describen algunas de las técnicas utilizadas en el proceso de visualización de datos:

1. *Diagramas de Caja*

   Se tuvieron en cuenta aquellos datos numéricos previamente discretizados, estos son: monto de la transacción, ingresos, patrimonio, días de vinculación y edad; los cuatro últimos son datos asociados al cliente. Dentro de la información analizada se encontraron algunos detalles de alto valor:

   a) Los montos de transacciones superiores a USD 1.767 son escasos y representan el 4.58 % del total de registros existentes.

   b) En el ingreso mensual se tiene que el 94.2 % de las transacciones es realizada por personas con ingresos inferiores a 700 mil pesos. Sin embargo existen transacciones realizadas por personas que poseen un ingreso superior, pero este porcentaje es muy reducido.

   c) De manera similar que en el ingreso mensual, el patrimonio de los clientes no suele superar los 25 millones de pesos (i.e. el 97.5 % de las transacciones), mientras que el 2.5 % de estos datos son poco comunes.

   d) La edad de los clientes está concentrada alrededor de los 37 y 47 años.

   e) El tiempo de vinculación como cliente se concentra principalmente entre los 333 y los 806 días.

   Las Figuras 4.1 y 4.2 reflejan esta información de manera resumida:

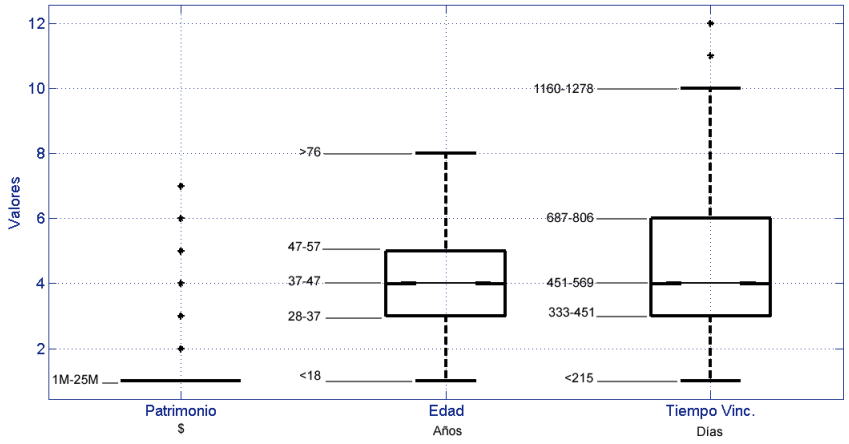

Figura 4.1: BoxPlots Patrimonio-Edad-Tiempo

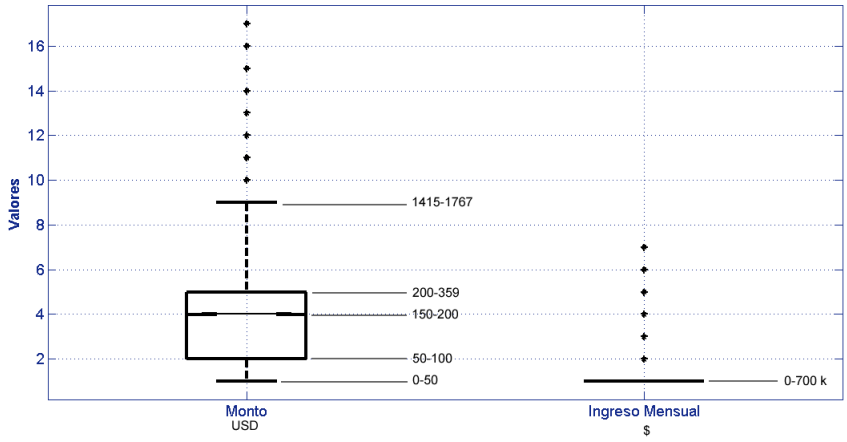

Figura 4.2: BoxPlots Monto-Ingreso mensual

2. Caras de Chernoff

   El conjunto de muestra tomado corresponde a los datos numéricos previamente discretizados y normalizados entre 0 y 1 . La relación de los atributos con las características de la cara es:

   a) Monto acumulado con el alto de la cara

   b) Ingreso mensual con el ancho de la cara

   c) Edad con la forma de la cara

d) Patrimonio con el alto de la boca

e) Días vinculación con el ancho de la boca

Al realizar el análisis se encontraron 7030 caras de las cuales se pueden distinguir dos patrones: atípicos (no tan frecuentes) y comunes (frecuentes), como se ilustra en la figura 4.3

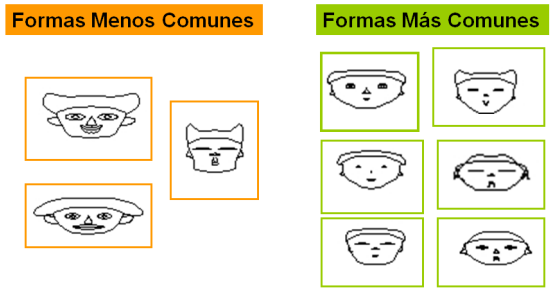

Figura 4.3: Caras de Chernoff

4.2. Muestreo

Debido a la gran cantidad de registros disponibles y a las altas exigencias de procesamiento que se necesita para aplicar las técnicas, fue necesario realizar un muestreo estratificado considerando la clase que permite identificar si una transacción es usual o inusual (i.e. el Estado de la Transacción). Inicialmente se cuentan con transacciones con cuatro estados posibles: usuales o normales (N), inusuales aprobadas (E), inusuales rechazadas (R) y transacciones canceladas (C).

El número de transacciones rechazadas (R) es cero, mientras que las transacciones canceladas (C) no aportan información al modelo ya que son operaciones cuya información se encuentra incompleta y no superan los 30 registros. Esto conduce a que solo se tengan dos clases útiles que se encuentran representadas en las transacciones usuales (N) e inusuales (E).

En primera instancia, se consideran los siguientes muestreos estratificados respecto al estado de la transacción:

1. *Caso1:* Se considera el conjunto de operaciones completamente distintas con los atributos finales seleccionados (véase sección Preparación de datos), lo que reduce el conjunto a 157.200 registros, 152.944 operaciones usuales (97.29 %) y 4.256 inusuales (2.71 %). Se considera una muestra del 10 % de los registros.

2. *Caso 2*: Dada la proporción de los datos en torno al tipo de la clase de transacción, 97.29 % corresponde a datos etiquetados como usuales y solo un 2.71 % a inusuales, debido al reducido número de ejemplos de la clase inusual (E) se realizó un muestreo estratificado de asignación proporcional con el fin de realzar la cantidad de ejemplos de transacciones inusuales y tener una muestra proporcional de ambas clases, respecto al número total de muestras, en función de sus respectivos tamaños, para ello se empleó la funcionalidad provista por Oracle denominado "ORA_HASH", la cual es una función que permite analizar un subconjunto de datos a través de la generación aleatoria de una muestra[Ora03].Lo anterior se hace para obtener una clasificación adecuada debido al poco número de ejemplos de la clase inusual.

4.3. Entrenamiento y Validación Red Neuronal

El modelo de red seleccionado para el proceso de aprendizaje fue una "MultiLayerPerceptron". Antes de iniciar el proceso de aprendizaje, se hizo una optimización de los parámetros de la red neuronal empleando la herramienta de RapidMiner, para ello se construyó un modelo de optimización de parámetros corriendo el algoritmo iterativamente y haciendo una validación cruzada para:

- Número de capas ocultas y de neuronas. Encontrado la topología con mejor porcentaje de clasificación a la compuesta por una capa oculta y 8 neuronas
- Momentun se definió en un rango de [0.1, 1.0] dando como parámetro óptimo 0.0029999999
- Rata de aprendizaje se definió en el rango de [0.1, 1.0], dando como parámetro óptimo 0.4

Se empleó la función sigmoide como función de activación para los nodos de la capa oculta y de salida. El número de iteraciones se definió en 500 épocas y se hizo una validación cruzada de 10 ”folding”, de igual forma en el proceso de entrenamiento de la red, se hizo la codificación de los atributos de entrada en binario empleando la técnica de codificación de variables, de 14 atributos de entrada se pasó a tener 256 atributos. Se utilizó la herramienta de ”RapidMiner” para efectuar las simulaciones. Los resultados obtenidos para los diferentes experimentos, son presentados a través de una matriz de confusión y de curvas ROC. Finalmente se hizo una prueba de hipótesis para medir el desempeño del algoritmo de clasificación empleando *Análisis de varianza (ANOVA)* y *Prueba t por parejas (Pairwise t-Test).*

4.3.1. Modelo de clasificación con muestreo estratificado

Empleando un muestreo estratificado sobre la clase estado de la transacción se tomó el 10 % del total de registros, teniendo una muestra de 15.264 registros usuales y de 426 inusuales. Esta muestra fue empleada como conjunto de entrenamiento y prueba para la red neuronal, al igual que otras técnicas como árboles de decisión y Naive Bayes. Finalmente los modelos de entrenamiento fueron validados con todo el conjunto de datos (157.200 registros).

En las tablas 4.1 y 4.2 se presentan los resultados obtenidos por la red neuronal en términos de la matriz de confusión para el conjunto de entrenamiento/pruebas y validación.

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	15.333	60
Clase Inusual	354	65

Cuadro 4.1: Matriz Confusión NN -Conjunto de entrenamiento y pruebas caso 1

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	152.944	0
Clase Inusual	4256	0

Cuadro 4.2: Matriz de Confusión NN- Conjunto de validación caso 1

De las anteriores tablas, se aprecia cómo para el conjunto de entrenamiento y pruebas el

sistema pudo clasificar algunos inusuales, mientras que para el conjunto de validación, no identificó ninguna clase inusual. Lo anterior se explica dado el reducido número de registros en el conjunto de entrenamiento que no permitió al modelo de red tener suficientes ejemplos, para generalizar la clasificación de casos etiquetados como inusuales. De igual forma se observa como la red neuronal se inclinó por la clase con mayor número de registros.

En las tablas 4.3 y 4.4 se presentan los resultados del árbol de decisión en términos de la matriz de confusión para el conjunto de entrenamiento/pruebas y validación

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	15.287	106
Clase Inusual	266	153

Cuadro 4.3: Matriz de Confusión J48- Conjunto de entrenamiento y pruebas caso 1

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	151.901	1042
Clase Inusual	2700	1556

Cuadro 4.4: Matriz de Confusión J48- Conjunto de validación caso 1

De las anteriores tablas, se aprecia como el árbol de decisión clasificó casos inusuales y usuales tanto en el conjunto de entrenamiento y pruebas como en el conjunto de validación.

En las tablas 4.5 y 4.6 se presentan los resultados del modelo probabilístico de Naive Bayes en términos de la matriz de confusión para el conjunto de entrenamiento/pruebas y validación

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	15.311	82
Clase Inusual	330	89

Cuadro 4.5: Matriz de Confusión Naive Bayes- Conjunto de entrenamiento y pruebas caso 1

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	152.089	817
Clase Inusual	3329	927

Cuadro 4.6: Matriz de Confusión Naive Bayes-Conjunto de validación caso 1

Analizando la información de las matrices en las tablas 4.2, 4.4 y 4.6 en términos de los resultados del conjunto de validación, se comparan las diferentes técnicas de clasificación en cuanto Exactitud (Accuracy), Recall, Error, Tasa de Falsos Positivos (FP Rate) y Tasa de Falsos Negativos (FN Rate). La tabla 4.7 muestra estos resultados teniendo en cuenta la clase etiquetada como inusual (E) .

	Red Neuronal	Árbol de Decisión	Bayes
Accuracy	97.29 %	97.62 %	97.36 %
Precisión	-	58.89 %	53.15 %
Recall	0 %	36.56 %	21.78 %
Error	2.71 %	2.38 %	2.64 %
FP Rate	0 %	0,68 %	0.53 %
FN Rate	100 %	63.43 %	78.21 %

Cuadro 4.7: Comparación Técnicas caso de estudio 1

Igualmente, se realizó un análisis de comportamiento a través de Curvas ROC sobre las clases etiquetadas como inusuales (E) las cuales son el interés principal de este trabajo. La Figura 4.4 muestra las curvas ROC que comparan los algoritmos de red neuronal, Naive Bayes y J48.

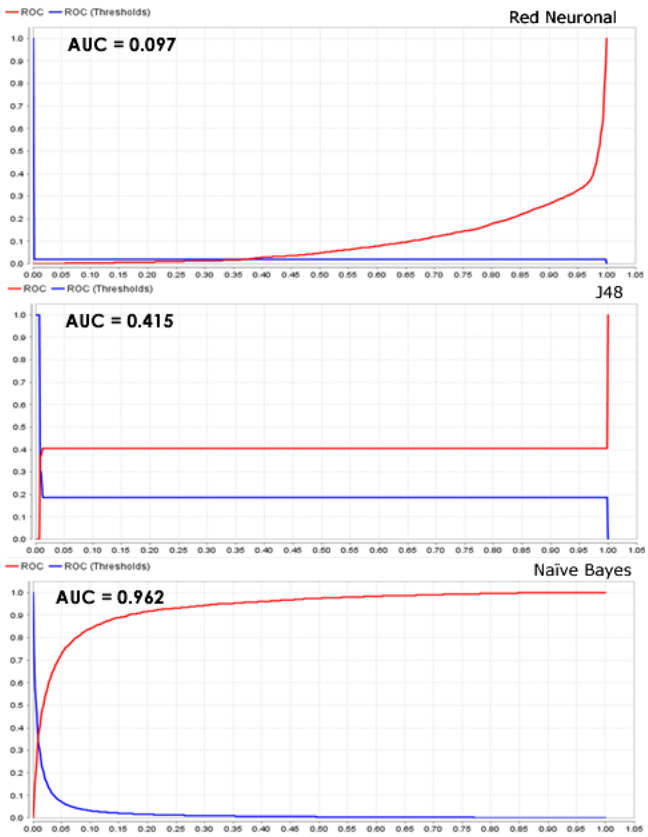

Figura 4.4: Curvas ROC - Caso de estudio 1

Analizando los resultados anteriores, dado el conjunto de datos de validación, se ve la mala exactitud de todas las técnicas en la detección de clases inusuales (error superior al 60 %), aunque la red neuronal es la de peor comportamiento respecto a los demás algoritmos (100 % de error). De igual forma, revisando las curvas ROC y el área bajo estas, el algoritmo con mejor desempeño es Naive Bayes con una área muy cercana a 1 (0,962) que contrasta con el mal resultado de la red neuronal y árbol de decisión con un porcentaje por debajo de 0.5. Los anteriores resultados se pueden justificar dado el reducido número de ejemplos de la clase inusual utilizados en el conjunto de entrenamiento, lo cual ocasionó que los algoritmos se inclinarán por la clase con mayor presencia.

Finalmente, los tiempos empleados para el conjunto de entrenamiento y validación se muestran en la tabla 4.8

Algoritmo	Entrenamiento(Seg)	Validación(Seg)
Red Neuronal	14400	150
J48	15	5
Bayes	5	5

Cuadro 4.8: Tiempo conjunto de entrenamiento y validación caso 1

De la tabla anterior, se aprecia cómo la red neuronal es la que emplea mayor tiempo en el proceso de entrenamiento y validación, que se une al bajo desempeño en comparación con Naive Bayes que fue la mejor técnica en cuanto a tiempo y desempeño de clasificación.

4.3.2. Modelo de clasificación con muestreo estratificado balanceando igual número de clases usuales e inusuales

Dado los malos resultados en el proceso de predicción para las clases etiquetadas como inusuales del muestreo anterior, se decidió hacer un muestreo estratificado proporcional sobre la clase estado de la transacción en una relación 50-50. Para ello, se tomó una muestra total de 6384 registros, correspondiente a un 4.06 % del porcentaje total, realzando la clase inusual con 3192 registros (75 % del total de registros inusuales) y 3192 registros de la clase usual (2 % del total de operaciones usuales). Esta muestra fue empleada como conjunto de entrenamiento y pruebas (validación cruzada de 10 "fold") para el modelo de red neuronal y otras técnicas como árboles de decisión y Bayes. Finalmente los modelos de entrenamiento se validaron sobre todo el conjunto de datos (157.200 registros).

En las tablas 4.9 y 4.10 se presentan los resultados obtenidos con la red neuronal para el conjunto de entrenamiento/pruebas y validación.

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	2925	260
Clase Inusual	248	2959

Cuadro 4.9: Matriz Confusión NN -Conjunto de entrenamiento y pruebas caso 2

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	145.023	7921
Clase Inusual	619	3637

Cuadro 4.10: Matriz de Confusión NN-Conjunto de validación caso 2

De las anteriores tablas, se aprecia cómo tanto para el conjunto de entrenamiento y validación el sistema clasificó casos inusuales y usuales en comparación con el anterior muestreo.

En la tablas 4.11 y 4. 12 se presentan los resultados del árbol de decisión sobre el conjunto de entrenamiento/pruebas y validación

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	2868	317
Clase Inusual	386	2821

Cuadro 4.11: Matriz de Confusión J48- Conjunto de entrenamiento y pruebas caso 2

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	137541	15337
Clase Inusual	265	3991

Cuadro 4.12: Matriz de Confusión J48- Conjunto de validación caso 2

En las tablas 4.13 y 4.14 se presentan los resultados del modelo probabilístico de Naive Bayes para el conjunto de entrenamiento y validación

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	2720	465
Clase Inusual	347	2860

Cuadro 4.13: Matriz de Confusión Naive Bayes- Conjunto de entrenamiento y pruebas caso 2

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	129556	23298
Clase Inusual	437	3819

Cuadro 4.14: Matriz de Confusión Naive Bayes- Conjunto de validación caso 2

Analizando la información de las matrices en las tablas 4.10, 4.12 y 4.14 en términos de los resultados del conjunto de validación, se comparan las diferentes técnicas de clasificación en cuanto Exactitud (Accuracy), Recall, Error, Tasa de Falsos Positivos (FP Rate) y Tasa de Falsos Negativos (FN Rate). La tabla 4.15 muestra estos resultados teniendo en cuenta la clase etiquetada como inusual (E) .

	Red Neuronal	Árbol de Decisión	Bayes
Accuracy	94.57 %	90.07 %	84.89 %
Precisión	31.47 %	20.65 %	14.08 %
Recall	85.46 %	93.77 %	89.73 %
Error	5,43 %	9.93 %	10,27 %
FP Rate	5.17 %	10.032 %	15.24 %
FN Rate	14.54 %	6.22 %	10.26 %

Cuadro 4.15: Comparación Técnicas - Caso de estudio 2

Igualmente, se realizó un análisis de comportamiento a través de Curvas ROC sobre las clases etiquetadas como inusuales (E) las cuales son el interés principal de este trabajo. La Figura 4.5 e.muestra las curvas ROC que comparan los algoritmos de red neuronal, Naive Bayes y J48.

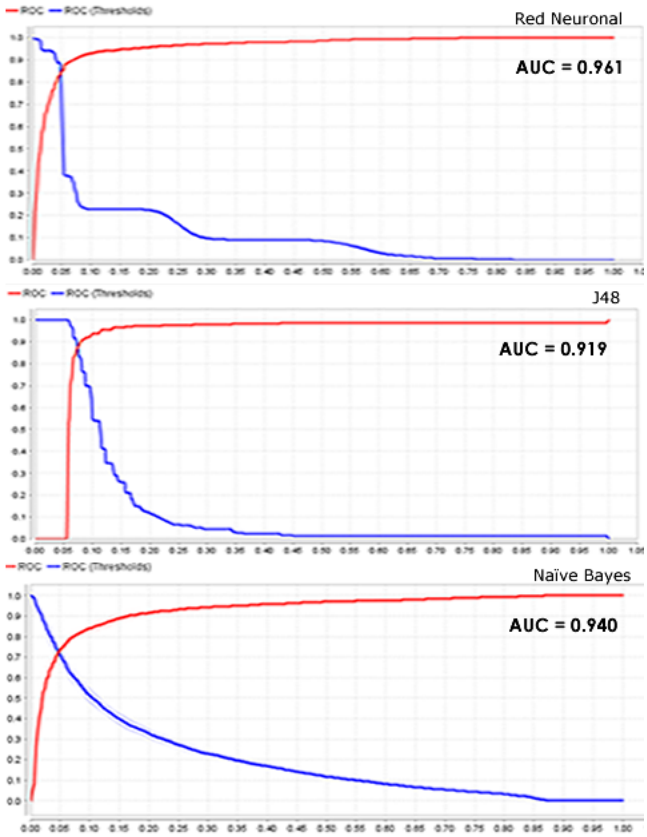

Figura 4.5: Curvas ROC - Caso de estudio 2

Analizando los resultados anteriores, dado el conjunto de datos de validación, se ve como la red neuronal incrementó su nivel de exactitud, casi en 5 % respecto a los demás algoritmos, aunque la medida de "recall" es un 6 % menor que las demás técnicas, la cual es de interés en este estudio. De igual forma, revisando las curvas ROC y el área bajo éstas, todos los algoritmos presentan un buen desempeño con un área mayor al 0.9 en comparación con el experimento de la sección anterior. Los anteriores resultados se pueden justificar dado el balanceo de ejemplos de la clase inusual efectuado en la muestra.

Finalmente, los tiempos empleados para el conjunto de entrenamiento y validación se muestran en la tabla 4.16

Algoritmo	Entrenamiento(Seg)	Validación(Seg)
Red Neuronal	6540	150
J48	15	5
Bayes	5	5

Cuadro 4.16: Tiempo entrenamiento y validación caso 2

De la tabla anterior, se aprecia como la red neuronal es la que emplea mayor tiempo en el proceso de entrenamiento y validación, lo cual se refleja con su buena exactitud de clasificación, aunque en el proceso de detección de clases inusuales es un poco menos competitiva que las demás técnicas.

4.3.3. Modelo de clasificación con muestreo estratificado proporcional realzando clases inusuales

Dado los resultados anteriores, se aprecia una mejoría en la predicción de clases inusuales como usuales, pero el error de predicción de clases inusuales sigue siendo bastante alto en comparación con las clases usuales, por ello se decidió hacer un muestreo estratificado proporcional sobre la clase estado de la transacción en una relación 57(usual) - 43(inusual), tomando una muestra total de registros 7397, correspondiente a un 3.59 % del porcentaje total. En este escenario, se realzó la clase inusual con 3207 registros (75.35 % del total de registros inusuales) y 4190 registros de la clase usual (2.67 % del total de operaciones usuales). Esta muestra fue empleada como conjunto de entrenamiento y pruebas para el algoritmo de red neuronal y otras técnicas como árboles de decisión y Bayes. Finalmente los modelos de entrenamiento se validaron con el total del conjunto de datos (157.200 registros).

En las tablas 4.17 y 4.18 se presentan los resultados obtenidos por la red neuronal para el conjunto de entrenamiento/pruebas y validación.

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	3877	313
Clase Inusual	308	2899

Cuadro 4.17: Matriz de Confusión NN- Conjunto de entrenamiento y pruebas caso 3

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	138.219	14.725
Clase Inusual	289	3967

Cuadro 4.18: Matriz de Confusión NN- Conjunto de validación caso 3

De tablas anteriores, se aprecia cómo para el conjunto de validación la red neuronal mejoró la clasificación de casos inusuales, respecto a los resultados obtenidos en el conjunto de entrenamiento y pruebas.

En las tablas 4.19 y 4.20 se presentan los resultados del árbol de decisión para el conjunto de entrenamiento/pruebas y validación

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	3820	370
Clase Inusual	294	2913

Cuadro 4.19: Matriz de Confusión J48- Conjunto de entrenamiento y pruebas caso 3

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	139.412	13.529
Clase Inusual	370	3886

Cuadro 4.20: Matriz de Confusión J48-Validación caso 3

En las tablas 4.21 y 4.22 se presentan los resultados del modelo probabilístico de Naive Bayes sobre el conjunto de entrenamiento/pruebas y validación

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	3704	486
Clase Inusual	437	2770

Cuadro 4.21: Matriz de Confusión Bayes- Conjunto de entrenamiento y pruebas caso 3

	Pred. Clase Normal	Pred. Clase Inusual
Clase Normal	135.336	17.533
Clase Inusual	548	3708

Cuadro 4.22: Matriz de Confusión Bayes- Conjunto de validación caso 3

Analizando la información de las matrices en las tablas 4.18, 4.20 y 4.22 en términos de los resultados obtenidos sobre el conjunto de validación, se pueden comparar las diferentes técnicas de clasificación en cuanto Exactitud (Accuracy), Recall, Error, Tasa de Falsos Positivos con un muestreo estratificado con igual número de clases usuales e inusuales (FP Rate) y Tasa de Falsos Negativos (FN Rate). La tabla 4.23 muestra estos resultados teniendo en cuenta la clase etiquetada como inusual (E) .

	Red Neuronal	Árbol de Decisión	Bayes
Accuracy	90.45 %	91.16 %	88.49 %
Precisión	21.22 %	23.31 %	17.46 %
Recall	93.21 %	91.31 %	87.12 %
Error	9.55 %	8.84 %	11.51 %
FP Rate	9.62 %	8.84 %	11.46 %
FN Rate	6.79 %	8.69 %	12.87 %

Cuadro 4.23: Comparación Técnicas - Caso de estudio 3

Igualmente, se realizó un análisis de comportamiento a través de Curvas ROC sobre las clases etiquetadas como inusuales (E) las cuales son el interés principal de este trabajo. La Figura 4.6 muestra las curvas ROC que comparan los algoritmos de red neuronal, Naive Bayes y J48.

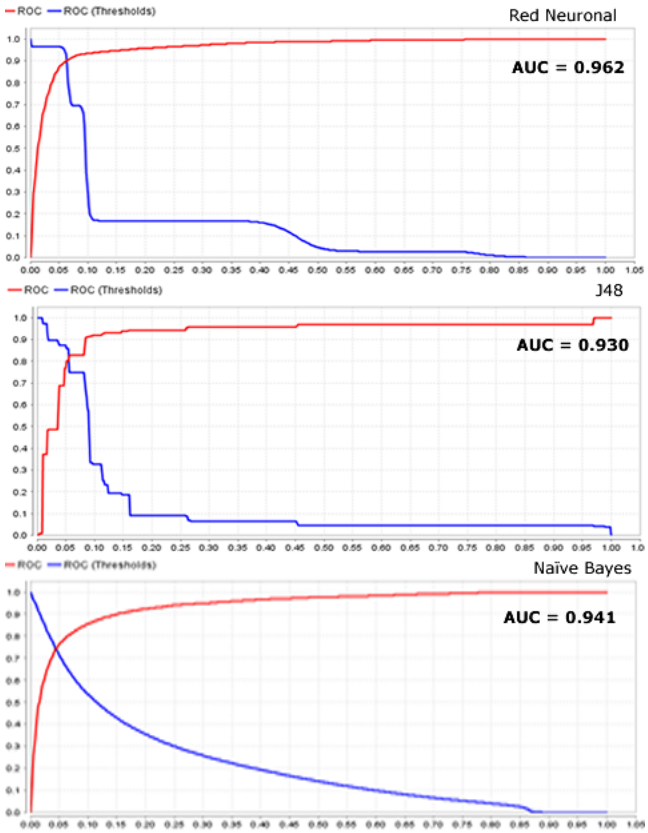

Figura 4.6: Curvas ROC - Caso de estudio 3

Analizando los resultados anteriores, dado el conjunto de datos de validación, se ve como la red neuronal incrementó su nivel de predicción de casos inusuales (recall y tasa de falsos negativos FN), en un 2 % respecto a los árboles de decisión y un 6 % respecto a Naive Bayes. De igual forma, revisando las curvas ROC y el área bajo estas, todos los algoritmos presentan un buen desempeño con un área mayor al 0.9, resaltando la red neuronal con un valor de 0.962.

Finalmente, los tiempos empleados para el conjunto de entrenamiento y validación se muestran en la tabla 4.24

Algoritmo	Entrenamiento(Seg)	Validación(Seg)
Red Neuronal	6600	150
J48	15	5
Bayes	5	5

Cuadro 4.24: Tiempo entrenamiento y validación caso 3

La tabla anterior muestra que el tiempo de entrenamiento y validación del modelo de la red neuronal respecto de los demás algoritmos es notoriamente alto, de la misma manera, para este experimento, la red obtuvo el mejor desempeño de clasificación de falsos negativos respecto a las demás técnicas, los cuales son el interés del presente trabajo.

4.3.4. Prueba de Hipótesis

Una vez que el modelo de clasificación se ha construido, es necesario evaluar su rendimiento respecto a modelos similares a fin de determinar, si el modelo propuesto es competitivo con los demás. En este caso, una prueba de significancia es realizada, en la cual, el modelo de Red Neuronal (NN) es comparado con un modelo de Árbol de decisión (J48) y un modelo probabilístico como Naive Bayes. La metodología para realizar las comparaciones se basó en seleccionar el mejor modelo encontrado, el cual corresponde al caso de la sección 4.3.3, teniendo como parámetro de desempeño la medida de "*recall*".

Las pruebas de significancia efectuadas son:

1. *Prueba t por parejas (Pairwise t-Test).* Una prueba "t-test" es una prueba de hipótesis estadística en la que el estadístico de prueba sigue una distribución t-Student. Para ello se parte de una hipótesis nula como verdadera, en este caso, se prueba si los tres clasificadores NN, J48 y Bayes, son iguales de acuerdo a su tasa de error en un intervalo de confianza. Para esta prueba, se definió una confianza de $1 - \alpha = 95\,\%$ con $\alpha = 0{,}05$ y una validación cruzada de 10 "fold".

   En la tabla 4.25 se presentan los resultados de la prueba "t-test" usando RapidMiner en función del valor $\alpha = 0{,}05$ y $K = 10$. Como resultado, se obtuvo el valor de *p-value=0,105* en la evaluación para el árbol de decisión y la red neuronal, lo cual no es menor que el valor de $\alpha = 0{,}05$. Así, la hipótesis nula no podemos rechazarla. Esto significa que la media de error de los clasificadores de la red neuronal y árbol de decisión son iguales. Caso contrario, el clasificador de Naive Bayes, tiene una diferencia significativa respecto a la red neuronal y árbol de decisión, dado que el valor obtenido del *p-value*

es cercano a cero($p - value = 4{,}484e - 07$), y este es menor que el valor de $\alpha = 0{,}05$. De igual forma, el valor de la distribución *t-Student experimental* $|t_{Exp} = -12{,}7837| \nleq t_9 \leqslant t_{0{,}05{,}9} = 1{,}883$, con lo cual se rechaza la hipótesis que el clasificador sea igual a los demás.

	0.878 +- 0.009	0.912 +- 0.009	0.918 +- 0.004
0.878 +- 0.009 (Bayes)	-	**0.000**	**0.000**
0.912 +- 0.009 (J48)	-	-	0.105
0.918 +- 0.004 (NN)	-	-	-

Cuadro 4.25: Comparación prueba t-test

2. *Análisis de varianza (ANOVA).* Es una generalización de dos muestras "t-Student" para más de dos grupos. Los resultados de la prueba ANOVA se presentan en una tabla con la siguiente información:

   - SS= Suma de cuadrados
   - MSS= Cuadrados de medias de error
   - DF= Grados de Libertad
   - F(Exp)= Distribución F experimental
   - F(Teo)= Distribución F teórica
   - Pro= Probabilidad
   - Fuente de variación. La cual hace referencia a :
     - SSB= Diferencia de error entre grupos y
     - SSW=Diferencia de error en el grupo

   En este caso, el interés se centra en probar si el modelo de Red Neuronal vs. el Árbol de decisión y Naive Bayes presentan diferencias significativas en su media de error. Para esta tarea, se emplearon los siguientes parámetros

   - Hipótesis nula: $H_0 = \mu_1 = \mu_2, = ... = \mu_L$, con $L = 3$ y $K = 10$
   - Confianza del 95 % ($\alpha = 0{,}05$)

En la tabla 4.26 se muestran los resultados obtenidos usando RapidMiner. La probabilidad obtenida dada una confianza del 95 % es considerablemente menor que el valor de alfa definido ($Pr = 0{,}0000001198 < \alpha = 0{,}05$), lo cual indica que la media de los errores entre los tres clasificadores (NN, J48 y Bayes) es diferente y tienen una variación significativa, así uno de estos algoritmos tiene mejor desempeño que los restantes. De manera similar, de acuerdo con el análisis anterior, podemos ver que: $60{,}810 \nsim F_{2,27} < F_{0,05,2,27} = 3{,}3541$, así la hipótesis nula es rechazada.

Variación	SS	DF	MSS	F(Exp)	F(Teo)	Prob.
Entre grupos	0.009	2	0.0045	60.810	3.3541	0.0000001198
En el grupo	0.002	27	0.000074			
Total	0.011	29				

Cuadro 4.26: Comparación prueba ANOVA

Dado los resultados anteriores, con el fin de determinar cual es el algoritmo de mejor desempeño, se hizo una nueva comparación entre el modelo de red neuronal y el de árboles de decisión, al igual, que el modelo de red neuronal y el modelo probabilístico de Naive Bayes con los siguientes parámetros:

- Hipótesis nula: $H_0 = \mu_1 = \mu_2, = ... = \mu_L$, con $L = 2$ y $K = 10$
- Confianza del 95 % ($\alpha = 0{,}05$)

En la tabla 4.27 se muestran los resultados de la comparación de la red neuronal vs. el árbol de decisión. Con una confianza del 95 %, la probabilidad obtenida fue mayor que el valor de alfa definido ($Pr = 0{,}10545 > \alpha = 0{,}05$), lo cual indica que la media de los errores entre los dos clasificadores (NN y J48) es similar y no hay una variación significativa para determinar que uno es mejor que otro. De acuerdo con el análisis anterior, podemos ver que: $3{,}245635 \sim F_{1,18} < F_{0,05,1,18} = 4{,}4138$, así la hipótesis nula es aceptada.

Variación	SS	DF	MSS	F(Exp)	F(Teo)	Prob.
Entre grupos	0.000157	1	0.000157	3.245635	**4.4138**	**0.10545**
En el grupo	0.0008711	18	0.00004839			
Total	0.0010281	19				

Cuadro 4.27: Comparación NN vs J48 - Prueba ANOVA

De igual forma, en la tabla 4.28 se muestran los resultados de la comparación de la red neuronal vs. Naive Bayes. Dada una confianza del 95 %, la probabilidad obtenida es menor que el valor de alfa definido ($Pr = 0{,}00000000008099 < \alpha = 0{,}05$), lo cual indica que la media de los errores entre los dos clasificadores (NN y Bayes) es diferente y tienen una variación significativa, así, el algoritmo de red neuronal tiene mejor desempeño. De manera similar, de acuerdo con el análisis anterior, podemos ver que: $180{,}139 \nsim F_{1,18} < F_{0,05,1,18} = 4{,}4138$, así la hipótesis nula es rechazada.

Variación	SS	DF	MSS	F(Exp)	F(Teo)	Prob.
Entre grupos	0.0078534	1	0.0078534	180.139	4.4138	0.00000000008099
En el grupo	0.0007847	18	0.00004359			
Total	0.0086377	19				

Cuadro 4.28: Comparación NN vs Naive Bayes - Prueba ANOVA

4.4. Extracción de reglas de clasificación a partir del modelo de Red Neuronal

El algoritmo con el cual se adelantó la parte experimental de extracción de reglas a partir de la red neuronal "MultiLayerPerceptron" previamente entrenada fue "M of N", el cual fue implementado y acoplado al software de RapidMiner. Para ello se validó con los diferentes experimentos planteados en la sección anterior, empleando como parámetro de distancia de agrupamiento el valor de 0.25 (según la literatura el más óptimo [MHRT06]), obteniendo los siguientes resultados:

4.4.1. Extracción de reglas a partir de un modelo de red con muestreo estratificado

Muestreo estratificado de la clase estado de la transacción tomando el 10 % del total de los datos, en una relación 98 % de clases usuales y 2 % de clases inusuales.

Durante la fase inicial de extracción se obtuvieron 17 reglas, en términos de la relación entre la capa de entrada y oculta, de igual forma entre la capa oculta y salida. Después del proceso de poda y re-escritura de las reglas en términos de las entradas y salidas se obtuvieron finalmente 3 reglas. Las reglas obtenidas fueron validadas respecto a todo el conjunto de datos.

1. IF (1 of (Attrib->estadoCivil=SOLTERO,Attrib->montoAcumulado=1514-2259,Attrib->montoAcumulado=769-1514,Attrib->paisOrigen=DESCONOCIDO)) THEN N Confianza: 97.04
2. IF (1 of (Attrib->montoAcumulado=<=769)) THEN N Confianza: 98.57 %
3. IF (1 of (Attrib->estadoCliente=A)) THEN N Confianza: 98.37 %

Con este modelo, el proceso de extracción sólo obtuvo reglas que caracterizan casos usuales, lo cual coincide con la medida de "recall" igual a cero que dio la red al hacer el proceso de validación sobre el conjunto de datos.

De igual forma, se comparan los resultados experimentales del algoritmo de extracción, con los resultados de otras técnicas tales como árboles de decisión (j48) y reglas de asociación (JRip) con el fin de ver la cantidad y exactitud de las reglas obtenidas. Los resultados obtenidos para cada técnica fueron:

1. *J48.* Con esta técnica se obtuvieron 3656 reglas, obteniendo un 92 % de reglas para casos usuales y un 8 % para casos inusuales. Algunos ejemplos de las reglas generadas son:
   - *a*) Si estadoCliente = I | diasVinculacion = 0-215: E Confianza: 70 %
   - *b*) Si estadoCliente = I | diasVinculacion = 0-215 | diasVinculacion = 215-333 |rangoEdad = 0-18: N Confianza:100 %
   - *c*) Si estadoCliente = I | diasVinculacion = 0-215 | diasVinculacion = 215-333|rangoEdad = 18-28: N Confianza:69 %
   - *d*) Si estadoCliente = A | montoAcumulado = 1514-2259 | estadoCivil = SOLTERO: N Confianza: 99 %
   - *e*) Si estadoCliente = I | diasVinculacion = 333-451 | departamentoDestino = AMAZONAS: N

f) SI estadoCliente = A | montoAcumulado = 0-769: N

g) Si estadoCliente = A | montoAcumulado = 0-769: N

h) Si estadoCliente = A | montoAcumulado >=15981: N

i) SI estadoCliente = A | montoAcumulado = 1514-2259 | estadoCivil = SEPARADO | paisOrigen = AFGANISTAN: N

j) Si estadoCliente = E | paisOrigen = KAZAHISTAN: E

k) Si estadoCliente = E | paisOrigen = ANTIGUA: E

Para los numerale del (e) al (k) la Confianza fue: 0 %

Analizando las reglas obtenidas sobre el conjunto de validación, el 73 % tienen una confianza igual a cero, lo cual explica el por qué de tantas reglas.

2. *JRIP.* Con esta técnica se obtuvieron 6 reglas para casos inusuales. Algunos ejemplos de las reglas generadas son:

   a) (estadoCliente = E) => txestado=E Confianza: 60 %

   b) (montoAcumulado = 5430-10661) and (numeroTransacciones = 1-10) and (producto = SEND) and (usoRecursos = ACTIVIDAD-COMERCIAL) => txestado=E Confianza 47 %

   c) (montoAcumulado = 5430-10661) and (numeroTransacciones = 1-10) and (producto = SEND) and (sexo = F) and (departamentoDestino = BOGOTA) => txestado=E Confianza: 45 %

   d) (montoAcumulado = >=15981) and (sexo = M) and (numeroTransacciones = 10-20) => txestado=E Confianza:32 %

   e) (montoAcumulado = 5430-10661) and (numeroTransacciones = 1-10) and (actividadEconomica = TURISTA) => txestado=E Confianza: 41 %

   f) (montoAcumulado = 10661-15981) and (numeroTransacciones = 1-10) and (diasVinculacion = 569-687) => txestado=E Confianza: 30 %

   g) => txestado=N

   Analizando las reglas obtenidas contra todo el conjunto de datos, esta técnica sólo generó reglas para casos inusuales, para los casos usuales la regla definida es la de omisión y no aporta información. De igual forma la confianza de las reglas está en su mayoría por debajo del 50 %.

Comparando las diferentes técnicas se aprecia cómo el algoritmo de extracción de reglas a partir de la red se inclinó sólo por reglas usuales dado el número de ejemplos de esta clase

en el proceso de entrenamiento, mientras que el algoritmo de reglas de asociación tuvo un comportamiento contrario, resaltando solo clases inusuales. De igual forma el árbol de decisión fue el único que presentó reglas para casos usuales e inusuales, pero muchas de ellas, no tienen una confianza que soporte dichas reglas (0 %), adicionalmente se vuelve una carga operativa el revisar demasiadas reglas.

4.4.2. Extracción de reglas a partir de un modelo de red con muestreo estratificado balanceando igual número de clases usuales e inusuales

Dada la no extracción de reglas para las etiquetas inusuales se decidió hacer un muestreo estratificado proporcional de la clase estado de transacción, en una relación 50-50.

Durante la fase inicial de extracción se obtuvieron 293 reglas, en términos de la relación entre la capa de entrada y oculta, de igual forma entre la capa oculta y salida. Después del proceso de poda y re-escritura de las reglas en términos de las entradas y salidas se obtuvieron finalmente 33 reglas. Las reglas obtenidas fueron validadas respecto a todo el conjunto de datos. A continuación se presentan las reglas más siginificativas:

1. IF (1 of (Attrib->estadoCliente=I)) THEN E Confianza:20 %
2. IF (1 of (Attrib->estadoCliente=E)) THEN E Confianza:60 %
3. IF (2 of (Attrib->estadoCliente=I,Attrib->paisOrigen=JAPON)) THEN E Confianza:100 %con un muestreo estratificado proporcional realzando clases inusuales
4. IF (1 of (Attrib->montoAcumulado=>=15981)) THEN E Confianza:27 %
5. IF (1 of (Attrib->montoAcumulado=10661-15981)) THEN E Confianza: 18 %
6. IF (3 of (Attrib->numeroTransacciones=>100,montoAcumulado>>=15981,actividadEconomica=ESTUDIANTE)) THEN E Confianza: 32 %
7. IF (4 of (Attrib-> diasVinculacion=215-333, Attrib->usoRecursos=VENTAS-ENSERES, Attrib->estadoCivil=SEPARADO, Attrib->departamentoDestino=ANTIOQUIA)) THEN E Confianza: 50 %
8. IF(3 of(Attrib->estadoCivil=SEPARADO,Attrib->departamentoDestino=ANTIOQUIA))) THEN E Confianza: 50 %
9. IF (1 of (Attrib->paisOrigenSERVIA-MONTENEGRO) THEN E Confianza: 33.63 %

Dado el modelo, sólo se obtuvieron reglas que caracterizan solo casos inusuales, con una confianza menor al 10 % del 70 % de las reglas obtenidas. De igual forma la confianza de la reglas esta por debajo de la exactitud de clasificación de la red.

Igualmente, se compara los resultados experimentales del algoritmo de extracción con los resultados de otras técnicas tales como árboles de decisión (j48) y reglas de asociación (JRip) con el fin de ver la cantidad y exactitud de las reglas obtenidas. Los resultados obtenidos fueron:

1. *J48.* Con esta técnica se obtuvieron 101 reglas, obteniendo un 77 % de reglas para casos usuales y un 33 % para casos inusuales. Algunos ejemplos de las reglas generadas son:

   a) Si estadoCliente = I: E Confianza: 20 %

   b) Si estadoCliente = R: E Confianza: 21 %

   c) Si estadoCliente = E: E Confianza: 60 %

   d) Si estadoCliente = A | montoAcumulado = >=15981: E Confianza: 26.70 %

   e) Si estadoCliente = A | montoAcumulado = <=769: N Confianza: 99 %

   f) Si estadoCliente = A | montoAcumulado = 5430-10661 |numeroTransacciones = 30-40 |actividadEconomica = HOGAR:N Confianza: 99 %

   g) Si estadoCliente = A | montoAcumulado = 769-1514: N Confianza: 99 %

   h) Si estadoCliente = A | montoAcumulado = 3749-5430 |numeroTransacciones = 20-30:N Confianza: 99 %

   i) Si estadoCliente = A | montoAcumulado = 0-769: N

   j) Si estadoCliente = A | montoAcumulado = 15981+: N

   k) Si estadoCliente = A|montoAcumulado = 3749-5430|numeroTransacciones = >100: E
   Para los numerales i,j y j la Confianza fue: 0 %

   Validando las reglas obtenidas sobre el conjunto de datos de validación (todo el conjunto de datos), el 56 % tiene una confianza igual a cero, lo cual no tiene valor alguno.

2. *JRIP.* Con esta técnica se obtuvieron 10 reglas para casos usuales. Algunos ejemplos de las reglas generadas son:

   a) (estadoCliente = A) and (montoAcumulado = <=769) => txestado=N Confianza:99.5 %

   b) (estadoCliente = A) and (montoAcumulado = 769-1514) => txestado=N Confianza:99.7 %

$c)$ (montoAcumulado = 1514-2259) and (estadoCliente = A) => txestado=N Confianza:99.6 %

$d)$ (montoAcumulado = 2259-3749) and (actividadEconomica = HOGAR) and (estadoCliente = A) and (producto = PAID) => txestado=N Confianza:99.6 %

$e)$ (paisOrigen = DESCONOCIDO) => txestado=N Confianza:100 %

$f)$ (estadoCliente = A) and (montoAcumulado = 2259-3749) and (estadoCivil = CASADO) and (diasVinculacion = 215-333) => txestado=N Confianza:99.5 %

$g)$ (estadoCliente = A) and (montoAcumulado = 2259-3749) and (actividadEconomica = INDEPENDIENTE) and (producto = PAID) => txestado=N Confianza:99.27 %

$h)$ => txestado=E

Analizando las reglas obtenidas sobre todo el conjunto de datos, esta técnica sólo generó reglas para casos usuales, para los casos inusuales la regla definida es muy vaga y no aporta información alguna. Se resalta que la confianza de las reglas para casos usuales, está en su mayoría en el 99 %. Se resalta con el experimento de la sección anterior, el cambio en el algoritmo, de predecir solo inusuales paso a predecir usuales.

Revisando las diferentes técnicas, se aprecia como el algoritmo de extracción de reglas a partir de la red, se inclinó sólo por reportar reglas inusuales, a pesar que en la muestra del conjunto de entrenamiento se equilibró en las dos clases. De otro lado, el algoritmo de reglas de asociación tuvo un comportamiento contrario, resaltando sólo clases usuales. Finalmente las reglas generadas por el árbol de decisión fue el único que presentó reglas para clases usuales e inusuales, pero muchas de ellas, tienen una confianza igual a cero.

4.4.3. Extracción de reglas a partir de un modelo de red con muestreo estratificado proporcional realzando clases inusuales

Dada los resultados anteriores de sólo extraer reglas para un grupo de clases en particular, se hizo nuevamente un muestreo estratificado proporcional al de la clase estado de transacción, en una relación 57(usual) - 43(inusual).

Durante la fase inicial de extracción se obtuvieron 140 reglas, en términos de la relación entre la capa de entrada y oculta, de igual forma entre la capa oculta y salida. Después del

proceso de poda y re-escritura de las reglas en términos de las entradas y salidas se obtuvieron finalmente 51 reglas. Las reglas obtenidas fueron validadas respecto a todo el conjunto de datos. A continuación se presentan las reglas más significativas:

1. IF (1 of (Attrib->estadoCliente=I)) THEN E Confianza:20 %
2. IF (1 of (Attrib->estadoCliente=E)) THEN E Confianza:60 %
3. IF (1 of (Attrib->montoAcumulado=>=15981)) THEN E Confianza:27 %
4. IF (2 of (Attrib->estadoCliente=I,Attrib->paisOrigen=JAPON)) THEN E Confianza:100 %
5. IF (1 of (Attrib->estadoCliente=R)) THEN E Confianza:21 %
6. IF (1 of (Attrib->estadoCliente=A)) THEN N Confianza:98.35 %
7. IF (1 of (Attrib->paisOrigen=DESCONOCIDO)) THEN N Confianza:100 %
8. IF (1 of (Attrib->estadoCivil=SOLTERO) THEN N Confianza: 97.04 %
9. IF (1 of (Attrib->montoAcumulado=<=769)) THE N Confianza: 98.57 %
10. IF (1 of (Attrib->montoAcumulado=769-1514)) THEN N Confianza: 98.7 %
11. IF (1 of (Attrib->montoAcumulado=1514-2259)) THEN N Confianza: 98.91 %
12. IF (1 of (Attrib->diasVinculacion=1160-1278)) THEN N Confianza: 97.99 %
13. IF (3 of (Attrib->actividadEconomica=HOGAR,Attrib->paisOrigen=ECUADOR, Attrib->numeroTransacciones=40-50,Attrib->numeroTransacciones=30-40)) THEN N Confianza: 95.23 %
14. IF (1 of (Attrib->actividadEconomica=EMPLEADO)) THEN N Confianza: 97.27 %

Dado el modelo, se obtuvieron reglas que caracterizan casos inusuales (27 reglas) y casos usuales (24 reglas). Para los casos usuales las reglas tienen una confianza mayor al 97 %, mientras que para los casos inusuales, el promedio es menor a un 50 %.

De igual forma, los resultados experimentales del algoritmo de extracción son comparados con los resultados de otras técnicas tales como árboles de decisión (j48) y reglas de asociación (JRip) con el fin de ver la cantidad y exactitud de las reglas obtenidas. Los resultados obtenidos fueron:

1. *J48.* Con esta técnica se obtuvieron 377 reglas, obteniendo un 84.88 % de reglas para casos usuales y un 15.12 % para casos inusuales. Algunos ejemplos de las reglas generadas son:

   $a)$ Si estadoCliente = I: E Confianza: 20 %

   $b)$ Si estadoCliente = R: E Confianza: 21 %

   $c)$ Si estadoCliente = E: E Confianza: 60 %

   $d)$ Si estadoCliente = A| montoAcumulado = 10661-15981: E Confianza: 17.52 %

   $e)$ Si estadoCliente = A | montoAcumulado = 3749-5430 numeroTransacciones = 1-10: E Confianza: 7.32 %

   $f)$ Si estadoCliente = A | montoAcumulado = 5430-10661 numeroTransacciones = 1-10: E Confianza: 23.60 %

   $g)$ Si estadoCliente = A | montoAcumulado = 2259-3749 | numeroTransacciones = 1-10 actividadEconomica = ESTUDIANTE: E Confianza: 14.39 %

   $h)$ Si estadoCliente = A | montoAcumulado <=769 :N Confianza: 99.84 %

   $i)$ Si estadoCliente = A | montoAcumulado =1514-2259 :N Confianza: 99.66 %

   $j)$ Si estadoCliente = A | montoAcumulado =5430-10661|numeroTransacciones = 20-30:N Confianza: 97.87 %

   $k)$ Si estadoCliente = A | montoAcumulado =5430-10661| numeroTransacciones = 10-20|paisOrigen = AFGANISTAN:N

   $l)$ Si estadoCliente = A | montoAcumulado =5430-10661| numeroTransacciones = 10-20|paisOrigen =ESPAÑA|usoRecursos = ARREGLO-VEHICULO: E

   $m)$ Si estadoCliente = A | montoAcumulado =5430-10661| numeroTransacciones = 10-20|paisOrigen =ESPAÑA|usoRecursos = AHORROS: E

   Para los numerales k,l y m la Confianza fue:0 %

   Analizando las reglas obtenidas sobre todo el conjunto de datos, el 81.16 % tienen una confianza igual a cero, lo cual no aporta valor alguno.

2. *JRIP.* Con esta técnica se obtuvieron 16 reglas para casos inusuales. Algunos ejemplos de las reglas generadas son:

   $a)$ (estadoCliente = E) => txestado=E Confianza:60 %

   $b)$ (montoAcumulado = 5430-10661) and (numeroTransacciones = 1-10) => txestado=E Confianza:24 %

   $c)$ (montoAcumulado = >=15981) => txestado=E Confianza:27 %

   $d)$ (montoAcumulado = 10661-15981) => txestado=E Confianza:18.21 %

   $e)$ (montoAcumulado = 3749-5430) and (numeroTransacciones = 1-10) and (producto = SEND) => txestado=E Confianza:14.98 %

$f)$ (montoAcumulado = 3749-5430) and (actividadEconomica = ESTUDIANTE) => txestado=E Confianza:15.70 %

$g)$ (montoAcumulado = 5430-10661) and (departamentoDestino = VALLE-DEL-CAUCA) and numeroTransacciones = 10-20) => txestado=E Confianza:4.86 %

$h)$ => txestado=N

Validando las reglas obtenidas sobre el conjunto de validación, este técnica sólo generó reglas para casos inusuales, para los casos usuales, la regla definida es muy vaga y no ayuda en mucho. De igual forma la confianza de las reglas está por debajo del 50 %.

Comparando las diferentes técnicas, se aprecia como el algoritmo de extracción de reglas a partir de la red reportó reglas para clases inusuales como usuales, mientras que el algoritmo de reglas de asociación tuvo un comportamiento contrario, resaltando solo clases inusuales. De igual forma el árbol de decisión fue el único que presento reglas con una confianza igual a cero (81 % del total de reglas generadas). Se resalta que la confianza para las reglas usuales supera más del 97 %, sin embargo, las reglas de clasificación inusual está por debajo del 50 %.

En la tabla 4.29 se presenta un resumen comparativo de los resultados de experimentación del algoritmo de extracción implementado vs otras técnicas.

Exp	Algoritmo	# Reglas	#Reg Usuales	#Reg Inusuales	Confianza=0	Tiempo(seg)
1	M of N	3	3	0	-	2
1	J48	3656	3363	293	2690	5
1	JRIP	6	0	6	-	16
2	M of N	33	0	33	-	8
2	J48	101	73	28	57	5
2	JRIP	10	10	0	-	11
3	M of N	51	24	27	-	12
3	J48	377	320	57	306	4
3	JRIP	16	0	16	-	18

Cuadro 4.29: Resultados experimentación técnicas de extracción de reglas

Todos los experimentos anteriores, se corrieron en un equipo con las siguientes especificaciones:

- Memoria RAM: 3 GB
- Procesador: Intel Core 2 Duo T5550 / 1.83 GHz

4.5. Resumen del Capítulo

Este capítulo presentó una serie de experimentos efectuados sobre el conjunto de datos de estudio. Inicialmente se describe el proceso de selección y preparación efectuado al conjunto de datos. Posteriormente, dado el bajo número de casos inusuales en el conjunto de datos, se empleó un muestro estratificado para los siguientes escenarios:

1. Tomando una muestra del 10 % del total de los datos.
2. Tomando una muestra en la que se realza los casos inusuales en un proporción 50-50 y 56-44.

A partir de estos escenarios, se evaluó el modelo de detección de fraude propuesto en el contexto del proceso de clasificación y validación de la red neuronal y el proceso de extracción de reglas. El propósito de los experimentos, fue orientado a evaluar la exactitud de clasificación para casos inusuales y rendimiento del modelo de red propuesto respecto a otras técnicas como árboles de decisión y Naive Bayes. Para ello se empleó el análisis de matrices de confusión y curvas ROC. Igualmente se presentó un análisis de hipótesis utilizando ”t-test” y ”ANOVA”, en el cual se tomo como medida de desempeño ”recall”, encontrando que las redes neuronales y los árboles de decisión no presentan una diferencia estadística significativa para decir que un algoritmo es mejor que el otro; caso contrario con el modelo de clasificación de Naive Bayes, la diferencia si es significativa y por ende resulta mejor el uso de la red neuronal.

De igual manera, se evaluaron las reglas extraídas a partir de la red neuronal en términos de su confianza y número con otras técnicas como árboles de decisión y reglas de asociación. Finalmente, de los diferentes experimentos efectuados, para los casos de estudio 4.3.3 (red neuronal) y 4.4.3 (algoritmo de extracción de reglas) se obtuvieron los mejores resultados para el modelo propuesto de detección de fraude.

Capítulo 5

Conclusiones y Trabajos Futuros

Conclusiones

1. Las redes neuronales son uno de los enfoques más utilizados para el aprendizaje inductivo y han demostrado un comportamiento predictivo bueno en una gran variedad de problemas interesantes (i.e detección de fraude).Sin embargo tienen una gran limitación, sus hipótesis aprendidas suelen ser incomprensible. Para hacer frente a esta limitación, una serie de grupos de investigación han desarrollado técnicas para la extracción de reglas, las cuales consiste en representar las funciones de una red neuronal en un idioma, como reglas de inferencia simbólica, que facilita una mejor comprensión. El enfoque de esta tesis ha sido acoplar un modelo de red neuronal y un método de extracción de reglas, denominado M of N como parte de un modelo de detección de fraude, con el fin de ayudar al experto del negocio a examinar y verificar más fácilmente los resultados obtenidos para apoyar la toma de decisiones.

2. La adaptación del algoritmo M of N efectuado en esta tesis, tiene una ventaja significativa sobre otros algoritmos de extracción de reglas, ya que no requiere hacer poda de la red neuronal para extraer las reglas, lo cual evita un re-entrenamiento de esta, reduciendo los tiempos de procesamiento. Este enfoque es escalable para grandes redes

y dominios de problemas con espacios grandes características.

3. Como resultado de los diferentes experimentos efectuados sobre el modelo de red neuronal propuesto para la clasificación del conjunto de datos de estudio, se consiguió durante el proceso de validación(todo el conjunto de datos) un excelente desempeño de clasificación, con una exactitud del 90.45 % para casos usuales y 93.75 % en casos inusuales, para ello se empleo un muestreo estratificado proporcional que realza los casos inusuales en comparación con el tamaño total de la muestra.

4. El algoritmo de extracción de reglas implementado en esta tesis(M of N) aplicado sobre el modelo de red neuronal MultiLayerPerceptron previamente entrenado, obtuvo reglas de clasificación, que permite conocer los atributos tenidos en cuenta por la red neuronal para clasificar las clases analizadas. Para los efectos de experimentación sobre el conjunto de datos de estudio, la confianza de las reglas generadas para casos usuales es mayor al 97 %, mientras que para los casos inusuales la confianza en su mayoría es menor al 50 %. Este se debe al bajo número de registros de casos inusuales, sin embargo, el resultado obtenido es similar a otras técnicas como árboles de decisión(J48) y reglas de asociación(JRIP).

5. Comparando las reglas obtenidas del algoritmo implementado en el presente trabajo vs. las reglas generadas del modelo de árbol de decisión (J48) para el conjunto de datos estudiado, las reglas del árbol de decisión son numerosas lo que para un experto representa una alta carga operativa, mientras que las reglas obtenidas con el algoritmo propuesto, son menos del 20 % de las reglas del árbol. De igual forma se encontró en los diferentes experimentos que el 80 % de las reglas del árbol de decisión tienen una confianza igual a cero.

6. Resultado del análisis de las reglas obtenidas por el algoritmo implementado vs. un modelo de reglas de asociación (JRIP), las reglas del modelo de asociación sólo reportaron casos inusuales del conjunto de datos analizado, mientras que las reglas obtenidas con algoritmo propuesto produjeron reglas para ambos casos.

7. Los procesos de extracción de reglas siguen comportamientos lineales, lo cual contrasta con el proceso de aprendizaje de las redes neuronales, ocasionado que al momento de extraer las reglas de clasificación no se pueda representar todo el modelo embebido en la red.

8. Del análisis de hipótesis empleando "t-test" y "ANOVA", en el cual se tomo como medida de desempeño "recall", se encontró que las redes neuronales y los árboles de decisión no presentan una diferencia estadística significativa para decir que un algoritmo es mejor que el otro; caso contrario con el modelo de clasificación de Naive Bayes, la diferencia si es significativa y por ende resulta mejor el uso de la red neuronal.

9. Cualquiera que sea la técnica utilizada para la detección de fraude, es necesario hacer una labor de la mano de los expertos del negocio. En la construcción de los modelos, es necesario determinar el conjunto de entrenamiento, validación y pruebas con variables significativas que puedan arrojar un resultado confiable y no obvio.

Perspectivas de Trabajos Futuros

Los resultados presentados en el capitulo 4, indican que el método de extracción de reglas "M of N" es capaz de extraer reglas a partir de un modelo de red neuronal entrenado sin tener que hacer poda de la red, lo cual es una característica a resaltar en este método, en comparación con otras técnicas utilizadas para la extracción de reglas. Sin embargo, este método tiene algunas limitaciones que se proponen como estudios futuros:

1. *Proceso de agrupamiento.* Se sugiere utilizar diferentes técnicas de agrupamiento para la construcción de los grupos de pesos similares de entrada a cada neurona de la red, con el fin de optimizar las reglas generadas en el proceso.

2. *Distancia óptima de agrupamiento.* Se sugiere determinar en el actual proceso de agrupamiento, el valor óptimo de distancia requerida para la formación de los grupos.

3. *Independencia de la arquitectura de red neuronal.* Se recomienda adelantar estudios para la combinación de técnicas que permitan la extracción de reglas, independiente de la arquitectura de red neuronal empleada.

4. *Presentación de las reglas extraídas por "M of N".* Aunque las reglas extraídas desde la red neuronal por "M of N" son bastante comprensibles, se propone explorar alternativas de representación.

Se deben estudiar estrategias que permitan optimizar la aplicación de técnicas de Minería de Datos, analizando las siguientes premisas:

1. El gran volumen de información que presenta el sector Financiero demanda la utilización de un sistema altamente eficiente y escalable.

2. Datos altamente sesgados, sólo un porcentaje muy bajo de las transacciones resultan sospechosas, alrededor de 1 %; por lo que aplicar modelos de minería de datos a la información para la detección de Fraude, se podrían obtener respuestas muy obvias como que "todas las transacciones son normales o tienen una baja probabilidad de ser sospechosas".

3. Manejo de información incompleta y en algunos casos incoherentes, producto de la migración en los sistemas de información de las organizaciones.

Bibliografía

[ADT95] R. Andrews, J. Diederich, and A. B. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowledge-Based Systems*, vol. 8, no. 6, pp. 373–389, 1995.

[Alp04] E. Alpaydin, *Introduction to Machine Learning.* The MIT press Cambridge, MA, 2004.

[BF01] J. P. Bradford and J. A. Fortes, "Characterization and parallelization of decision-tree induction," *Journal of Parallel and Distributed Computing*, vol. 61, no. 3, pp. 322–349, 2001.

[BH02] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical Science*, vol. 17, no. 3, pp. 235–249, Aug. 2002, ArticleType: primary_article / Full publication date: Aug., 2002 / Copyright 2002 Institute of Mathematical Statistics. [Online]. Available: http://www.jstor.org/stable/3182781

[CHY96] M. S. Chen, J. Han, and P. S. Yu, "Data mining: An overview from a database perspective," *IEEE Transactions on Knowledge and data Engineering*, vol. 8, no. 6, pp. 866–883, 1996.

[CP95] C. Caroni and P. Prescott, "On rohlf's method for the detection of outliers in multivariate data," *Journal of Multivariate Analysis*, vol. 52, no. 2, pp. 295–307, 1995.

[Cra96] M. W. Craven, "extracting comprehensible models from trained neural networks," Ph.D. dissertation, University of Wisconsin -Madison, 1996.

[DG93] L. Davies and U. Gather, "The identification of multiple outliers," *Journal of the American Statistical Association*, vol. 88, no. 423, pp. 782–792, 1993.

[DMB04] D. Dancey, D. McLean, and Z. Bandar, "Decision tree extraction from trained neural networks," in *FLAIRS Conference*, 2004.

[Fu94a] L. M. Fu, *Neural networks in computer intelligence.* McGraw-Hill, Inc. New York, NY, USA, 1994.

[Fu94b] L. Fu, "Rule generation from neural networks," *IEEE transactions on systems, man and cybernetics*, vol. 24, no. 8, pp. 1114–1124, 1994.

[GR94] S. Ghosh and D. L. Reilly, "Credit card fraud detection with a neural-network," in *proceedings of the hawaii international conference on system sciences*, vol. 27, 1994, pp. 621–621.

[HDXH06] Z. He, S. Deng, X. Xu, and J. Z. Huang, "A fast greedy algorithm for outlier mining," in *Proceedings of 10th Pacific-Asia Conference on Knowledge and Data Discovery*, 2006, pp. 567–576.

[HK06] J. Han and M. Kamber, *Data mining: concepts and techniques.* Morgan Kaufmann, 2006.

[HS03] T. Hu and S. Y. Sung, "Detecting pattern-based outliers," *Pattern Recognition Letters*, vol. 24, no. 16, pp. 3059–3068, 2003.

[KI06] S. M. Kamruzzaman and M. M. Islam, "An algorithm to extract rules from artificial neural networks for medical diagnosis problems," *International Journal of Information Technology*, vol. Vol. 12 No. 8, pp. 41–59, 2006.

[KLSH04] Y. Kou, C. T. Lu, S. Sirwongwattana, and Y. P. Huang, "Survey of fraud detection techniques," in *Proceedings of IEEE Intl Conference on Networking, Sensing and Control*, 2004.

[KPJ+03] H. C. Kim, S. Pang, H. M. Je, D. Kim, and S. Y. Bang, "Constructing support vector machine ensemble," *Pattern Recognition*, vol. 36, no. 12, pp. 2757–2767, 2003.

[KSM07] E. Kirkos, C. Spathis, and Y. Manolopoulos, "Data mining techniques for the detection of fraudulent financial statements," *Expert Syst. Appl.*, vol. 32, no. 4, pp. 995–1003, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1222770

[LSL96] H. Lu, R. Setiono, and H. Liu, "Effective data mining using neural networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 957–961, 1996.

[MHRT06] H. MAYER, Huber, Rohde, and Tamme, "Rule extraction from artificial neural networks," *Universitšat Salzburg*, 12th October 2006 2006.

[MMS92] C. Mcmillan, M. C. Mozer, and P. Smolensky, "Rule induction through integrated symbolic and subsymbolic processing," in *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 1992, pp. 969–976.

[MP88] M. Minsky and S. Papert, "Perceptrons: An introduction to computational geometry." MIT press Cambridge, MA, 1988.

[MR05] O. Z. Maimon and L. Rokach, *Data mining and knowledge discovery handbook*. Springer-Verlag New York Inc, 2005.

[Ora03] Oracle, "Database sql reference-ora hash," 2003. [Online]. Available: http://download.oracle.com/docs

[PFF00] J. X. Pan, W. K. Fung, and K. T. Fang, "Multiple outlier detection in multivariate data using projection pursuit techniques," *Journal of Statistical Planning and Inference*, vol. 83, no. 1, pp. 153–167, 2000.

[PKB01] S. N. Pang, D. Kim, and S. Y. Bang, "Fraud detection using support vector machine ensemble," *Proc. ICORNIP2001*, pp. 1344–1349, 2001.

[PWM02] J. A. F. Pierna, F. Wahl, and D. Massart., "Methods for outlier detection in prediction," *Chemometrics and Intelligent Laboratory Systems*, vol. 63, no. 1, pp. 27–39, 2002.

[Qui86] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[Rap02] Rapidminer, "Rapidminer- paquete libre para trabajar minería de datos." 2002. [Online]. Available: http://rapid-i.com

[SL00] R. Setiono and W. K. Leow, "Fernn: An algorithm for fast extraction of rules from neural networks," *Applied Intelligence*, vol. 12, no. 1, pp. 15–25, 2000.

[STD07] A. Shen, R. Tong, and Y. Deng, "Application of classification models on credit card fraud detection," in *Service Systems and Service Management, 2007 International Conference on*, 2007, pp. 1–4.

[SZC02] H. Shao, H. Zhao, and G. Chang, "Applying data mining to detect fraud behavior in customs declaration," vol. 3, 2002.

[Thr94] S. B. Thrun, "Extracting symbolic knowledge from artificial neural networks," *Revised Version of Technical Research Report I-University Bonn*, 1994.

[TSK05] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining.* Pearson Addison Wesley Boston, 2005.

[TSN90] G. Towell, J. Shavlik, and M. Noordewier, "Refinement of approximate domain theories by knowledge-based neural networks," pp. 861–866, 1990.

[Wek] Weka, "Paquete libre para trabajar minería de datos," The University Waikato. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/

[ZB06] Q. Zhao and S. S. Bhowmick, "Association rule mining: A survey," *Nanyang Technological University, Singapore*, 2006.

Anexo A

Modelo de detección de Fraude implementado en RapidMiner

La implementación del modelo de detección, empleo como APIS de desarrollo :

- API de RapidMiner[Rap02], el cual facilito la construcción de los modelos XML sobre los cuales se entreno y valido la Red Neuronal, al igual que el proceso de extracción de reglas.
- API de Weka[Wek], con el cual se re-utilizo el algoritmo de red Neuronal "Multilayer Perceptron" y codificación de variables(Dummies).

A.1. Introducción a RapidMiner

RapidMiner (antes de Yale) es un entorno para el aprendizaje de máquinas y procesos de minería de datos. A través de un concepto modular, permite el diseño de modelos de aprendizaje empleando operadores de cadena para diversos problemas. El manejo de los datos es transparente para los diferentes operadores implementados, ya que no tienen que manejar los datos reales directamente, sino hace uso de transformaciones para su manejo. Hoy en día, RapidMiner es el líder en el mundo de código abierto para procesos de minería de datos y

aprendizaje de máquinas, y es ampliamente utilizado por los investigadores y empresas. Entre las características más sobresalientes tenemos:

1. El proceso de KDD es modelado como simples operadores de árbol.
2. Tiene una representación interna basada en archivos XML.
3. Tiene una interfaz gráfica de usuario (GUI) para el diseño de prototipos Interactivos.
4. Tiene una línea de comandos (modo batch) para ser automatizado a gran escala aplicaciones.
5. Tienen un API de Java que facilita el uso de RapidMiner para propios programas
6. Provee más de 400 operadores para algoritmos de aprendizaje de máquinas, operadores de WEKA, operadores de pre-procesamiento de datos, meta operadores, visualización y evaluación de desempeño. Para más información puede referirse a [Rap02]

A.1.1. Formato de Archivos

RapidMiner puede leer diversos formatos de entrada, como también leer y escribir modelos, conjuntos de parámetros y atributos. Los formatos más importantes que maneja son:

1. Formatos con extensión *.arff* , para ello usan el operador "ArffExampleSource"
2. Leer desde base de datos usando el operador "DatabaseExampleSource"
3. Formatos de archivos separados por un caracter de línea, para ello emplea el operador "ExampleSource". En este tipo de formato, la descripción de los atributos debe guardarse en un archivo XML con extensión .aml y los datos en un archivo de texto con el separador respectivo.

A.1.2. Operadores

Un operador es un proceso o algoritmo que ejecuta una tarea en especial. Cada operador se subdivide en varias partes:

1. Grupo e icono del operador
2. Una enumeración de las entradas y salidas de los objetos. Los objetos de entrada son generalmente consumidos por el operador y no son parte de la salida. En algunos casos, este comportamiento se puede modificar mediante un parámetro de mantener los datos en e l modelo diseñado.
3. Los parámetros de configuración del operador
4. Una lista de valores que se pueden registrar usando el operador "ProcessLog"
5. Si el operador representa un esquema de aprendizaje, las capacidades del aprendizaje son descritas.
6. Si el operador representa un operador de cadena (chain) una breve descripción de los operadores internos es dada.
7. Una descripción (corta y larga) del operador.

De acuerdo a los grupos de operadores se tienen los siguientes:

1. *Operadores Básicos*
    a) *Aplicar Modelo(ModelApplier).* Este operador se aplica a un modelo de datos (ExampleSet) para predicción o transformación de datos a otro modelo.
    b) *Agrupar Modelo(ModelGrouper).* Este operador Agrupa varios modelos de entrada en un solo modelo (modelos de reprocesamiento)
    c) *Desagrupar Modelo(ModelUngrouper).* Desagrupa un modelo en Modelos únicos.

d) *Actualizar Modelo(ModelUpdater).* Actualiza un modelo de acuerdo a un Ejemplo. Este operador solo puede usarse para actualizar modelos, caso contrario genera error.

e) *Operador de cadena(OperatorChain).* Una cadena de operadores es subsecuentemente aplicada. Este operador puede tener un número arbitrario de operadores internos .

2. *Operadores de Core*

a) *Operador de línea de comando(CommandLineOperator).* Este operador simplemente ejecuta un comando en un s sistema operativo o programa externo.

b) *Definición de macros(DataMacroDefinition).* Este operador puede ser usado para definir macros.

c) *Experimento(Experiment).* Operador raíz de cadena. El propósito de este operador es proveer algunos parámetros globales al moldeo.

d) *Salida de archivo(FileEcho).* Este operador simplemente dado un texto escribe este en un archivo

e) *Multiple entrada/salida(IOMultiplier).* Este operador simplemente multiplica los objetos seleccionados en una entrada.

f) *Recuperación de entrada/salida(IORetriever).* Este operador recupera un objeto que fue almacenado previamente en el proceso.

g) *Limpieza de memoría(MemoryCleanUp).* Liberar memoria dejada por operadores anteriores

h) *Proceso(Process).* operador raíz de la cadena necesario para ejecutar demás procesos.

3. *Operadores de Entada/Salida*

a) *Fuente de datos en Acces(AccessExampleSource).* Este operador lee registros desde una base de datos Access.

b) *Salida de datos en archivo .ARFF(ArExampleSetWriter).* Escribe los valores de un conjunto en un archivo .ARFF.

c) *Fuente de datos en .ARFF(ArExampleSource).* Este operador lee archivos .ARFF.

d) *Fuente de datos BibText(BibtexExampleSource).* Este operador lee archivos BibTEX.

e) *Salida de datos en archivo .cvs(CSVExampleSetWriter).* Este operador escribe archivos. Csv.

f) *Fuente de datos desde BD(DatabaseExampleSource).* Este operador lee un ejemplo desde una base de datos SQL.

g) *Fuente de entrada de datos genérica(ExampleSource).* Este operador lee archivo separado por un caracter de línea.

4. *Operadores de aprendizaje.* Este tipo de operadores agrupa los diferentes algoritmos de aprendizaje implementados en Rapidminer.

5. *Operadores de pre-procesamiento de datos.*

6. *Opradores de post-procesamiento de datos.*

7. *Operadores de visualización.*

8. *Operadores de validación de desempeño para algoritmo de aprendizaje.*

A.1.3. Interfaz Gráfica

La interfaz gráfica para el diseño de modelos, se basa en las siguientes partes fundamentales:

1. *Árbol de procesos.* En esta parte se define el modelo y los operadores a usar en este.

2. *Área de resultados.* En esta parte se visualizan los resultados producto de las ejecución de los modelos diseñados, al igual que los parámetros de configuración de los diferentes operadores.

3. *Área de Compilación y Ejecución.* En esta área se muestra el log de los procesos ejecutados en el árbol de procesos.

En la figura A.1, se muestra los componentes de la interfaz gráfica más importantes en RapidMiner

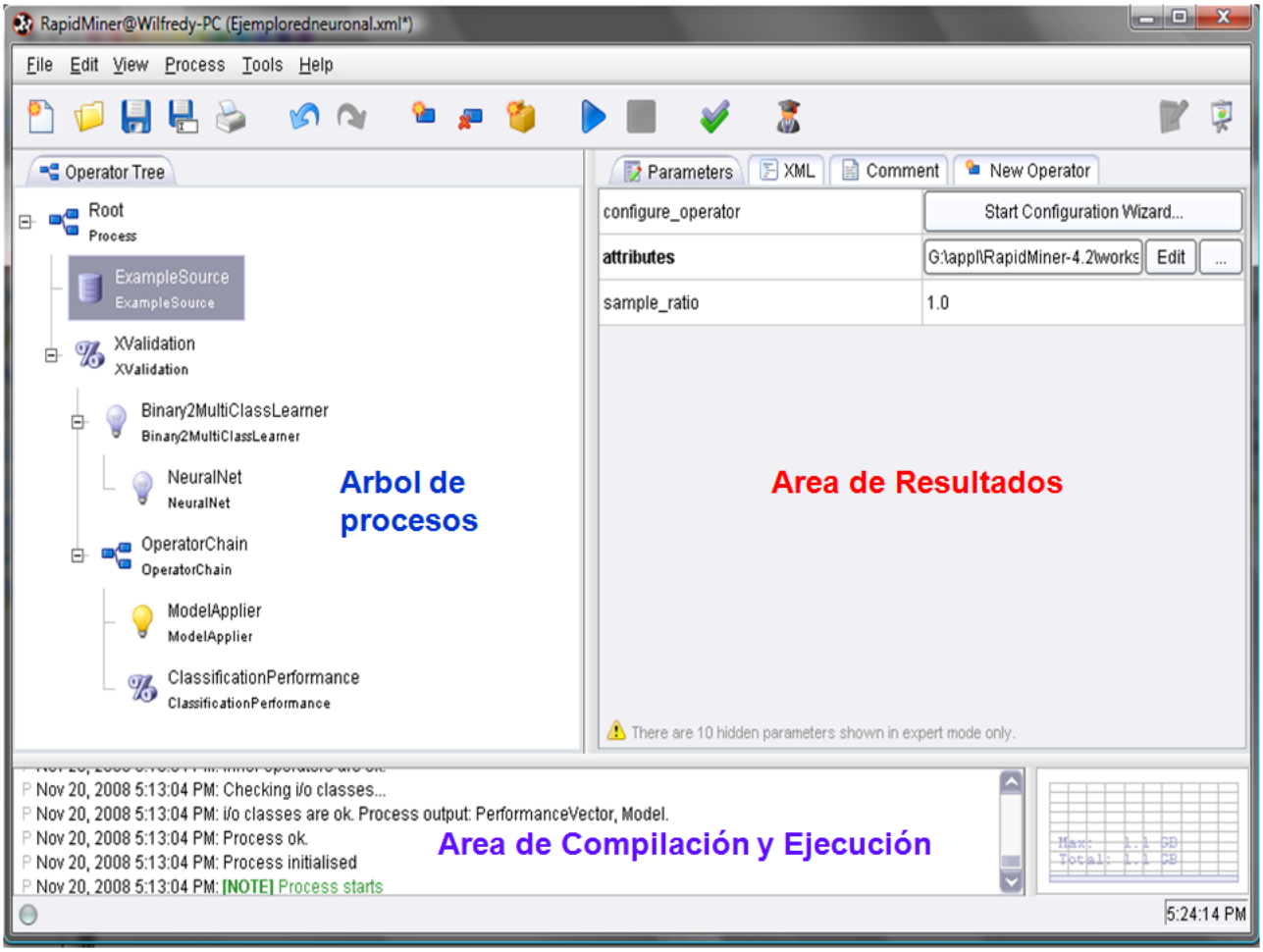

Figura A.1: Componentes de la Interfaz gráfica de RapidMiner

A.2. Modelos construidos en RapidMiner

El prototipo de modelo de detección de fraude, empleo los siguientes diagramas de componentes montados sobre RapidMiner:

A.2.1. Modelo de Optimización

El modelo de optimización se construyó con el fin de encontrar los parámetros óptimos de la red neuronal en cuanto a la topología de esta (número de capas ocultas y de neuronas), tasa de aprendizaje y momentum para ello se empleo los siguientes operadores:

1. *"ExampleSource"*: Este operador permite definir el conjunto de datos de entrada sobre el cual se ejecuta el proceso de optimización

2. *"GridParameterOptimization"*: Este operador permite definir los parámetros a optimizar del algoritmo de aprendizaje (para el caso, una red neuronal). Para ello en una grilla de búsqueda se listan los diferentes parámetros del algoritmo a optimizar y a su vez se define el rango de valores sobre los cuales se van a evaluar estos.

3. *"XValidation"*: Este operador permite definir el proceso de validación cruzada que se aplica sobre el conjunto de datos de entrada para evaluar el algoritmo de aprendizaje. Embebido en este operador se define los siguientes subprocesos:

   a) *"Learner"*: Con este operador se especifica el algoritmo de aprendizaje a usar.

   b) *"OperatorChain"*: Este operador ayuda a construir el proceso de evaluación de desempeño del operador de aprendizaje. Embebido en este operador se incluye los siguientes subprocesos:

      1) *"ModelAplier"*. Este operador ayuda a determinar el desempeño de la salida del algoritmo de aprendizaje

      2) *"RegresionPerformance"*: Este operador guarda las salidas del algoritmo de aprendizaje con los parámetros y valores utilizados en cada iteración hasta dejar los valores óptimos a través de un proceso de regresión.

4. *"ParametersetWrite"*: Este operador permite exportar los parámetros óptimos encontrados en el proceso a un archivo plano.

En la figura A.2, se presenta el modelo construido sobre la interfaz gráfica de RapidMiner.

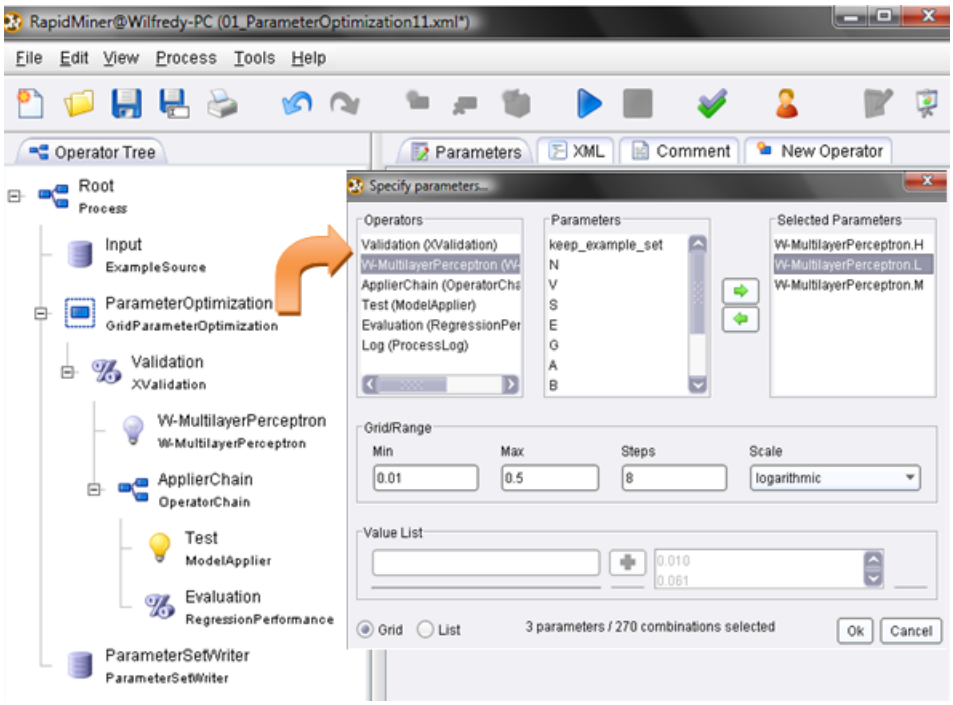

Figura A.2: Modelo de optimización de parámetros -Red Neuronal

A.2.2. Modelo de Entrenamiento Red Neuronal y Extracción de Reglas

El modelo implementado en RapidMiner para el entrenamiento y extracción de reglas empleo los siguientes operadores:

1. *"ExampleSource"*. Este operador permite definir el conjunto de datos sobre el cual se va entrenar el modelo, para ello se debe especificar los datos a cargar y la metadata de los mismos. A través de un asistente se permite hacer la configuración de :

   a) Selección del archivo de datos

   b) Definición de los separadores de cada atributo

   c) Definición nombre de atributos

   d) Definición del tipo de dato del atributo

   e) Definición atributo objetivo o clase

   f) Guardar la configuración, la cual es almacenada en una archivo XML con extensión .aml

Como parámetro adicional del proceso, se puede especificar el porcentaje de registros a tener en cuenta para el proceso de entrenamiento, para ello se define el porcentaje deseado en el parámetro *"sample_ratio"*. Las figuras A.3 y A.4 muestran los parámetros de configuración, al igual que la metadata de los atributos definidos.

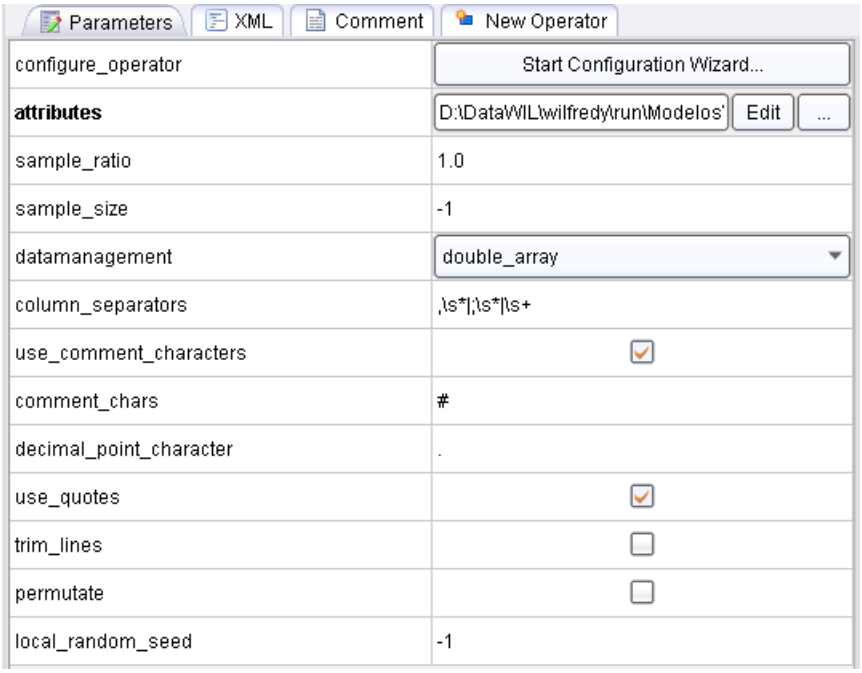

Figura A.3: Parámetros de configuración conjunto de datos

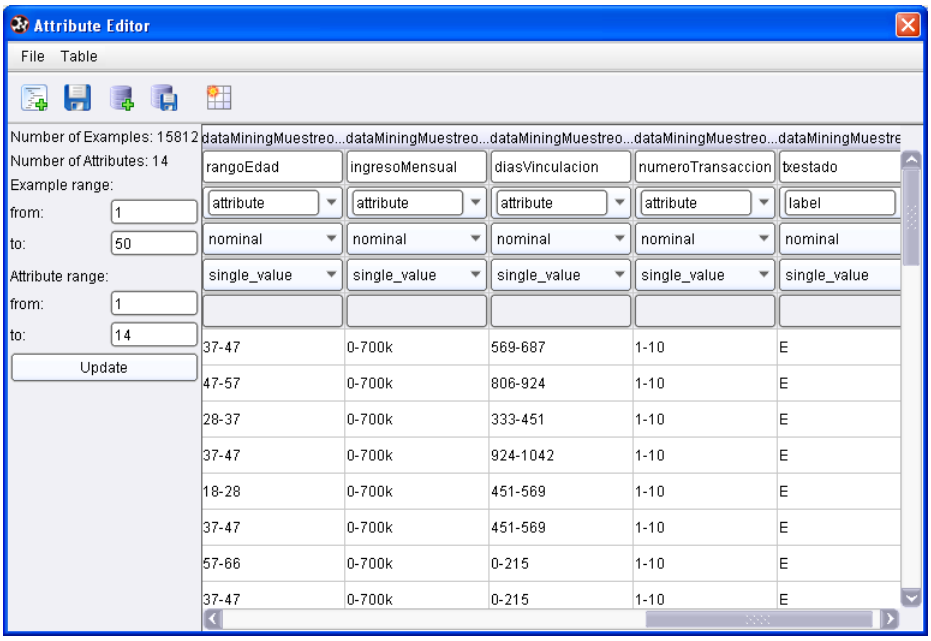

Figura A.4: Definición metadata de atributos y clase objetivo

2. *"XValidation"*. Con este operador se define el número de validaciones cruzadas a aplicar sobre el proceso, al igual que el tipo de muestreo aplicado a cada validación cruzada como se muestra en la figura A.5.

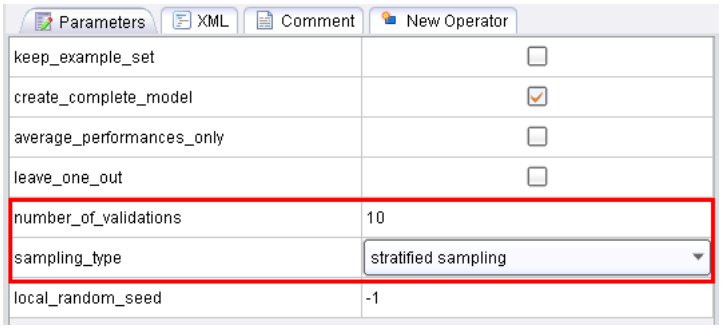

Figura A.5: Parámetros de configuración operador XValidation

Embebido en este operador se define los siguientes subprocesos:

a) *Learner*.Especifica el algoritmo de aprendizaje a utilizar, para efectos del caso de estudio se empleo *"W-MultilayerPerceptron"* correspondiente a una red neuronal "MultilayerPerceptron". Los parámetros definidos para este algoritmo previamente

se encontraron a través de la construcción de un modelo de optimización. Entre ello se destacan:

- Tasa de aprendizaje
- Momentum
- Número de capas oculta y neuronas. Para ello se utiliza una lista separada por comas para especificar las capas y el número para determinar la cantidad de neuronas a emplear.
- Repeticiones

La figura A.6 presenta los parámetros que se pueden configurar en este operador, para el caso, se resaltan los mencionados anteriormente.

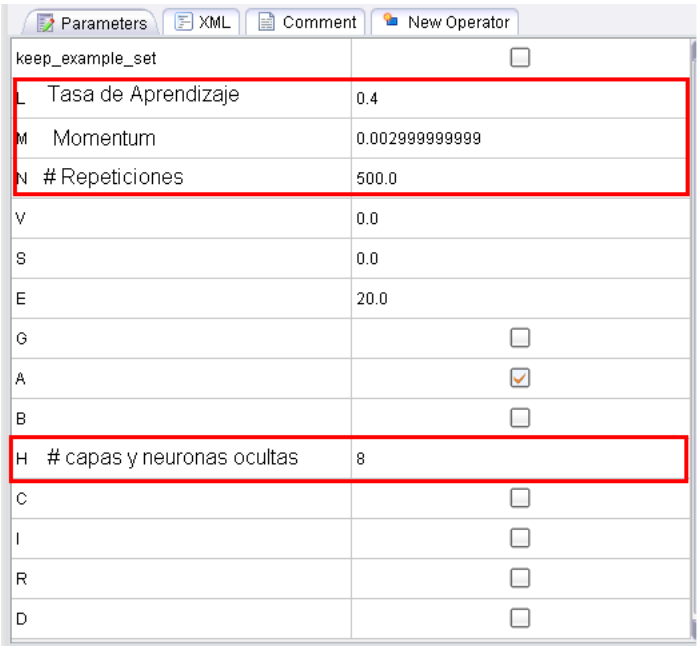

Figura A.6: Parámetros de configuración Red Neuronal

b) *"OperatorChain"*. Este operador ayuda a construir el proceso de evaluación de desempeño del operador de aprendizaje(MultilayerPerceptron). Embebido en este operador se incluye los siguientes subprocesos:

- *"ModelAplier"*. Este operador ayuda a determinar el desempeño de la salida del algoritmo de aprendizaje

- *"Performance"*: Este operador presenta los resultados de desempeño del algoritmo en términos de exactitud, precisión, recall, error y curva ROC.

3. *"ModelWriter"*. Este operador guarda en un archivo(xml, xmlzipped o binario) todo el modelo resultado del proceso de entrenamiento del algoritmo de aprendizaje al igual que los parámetros definidos en el árbol de procesos.

4. *"ResulWriter"*. Este operador guarda en un archivo plano el resultado del algoritmo de aprendizaje, para el caso, los pesos y bias óptimos de la red neuronal encontrados, el cual será la entrada al proceso de extracción de reglas implementado.

5. *"WSR-Rules"*. Este operador fue implementado y adicionado como un operador de aprendizaje supervisado a la herramienta de RapidMiner, el cual puede ser creado desde *"New Operator -> Leraner -> Supervised-> Functions ->WSR-Rules"* como se muestra en la figura A.7.

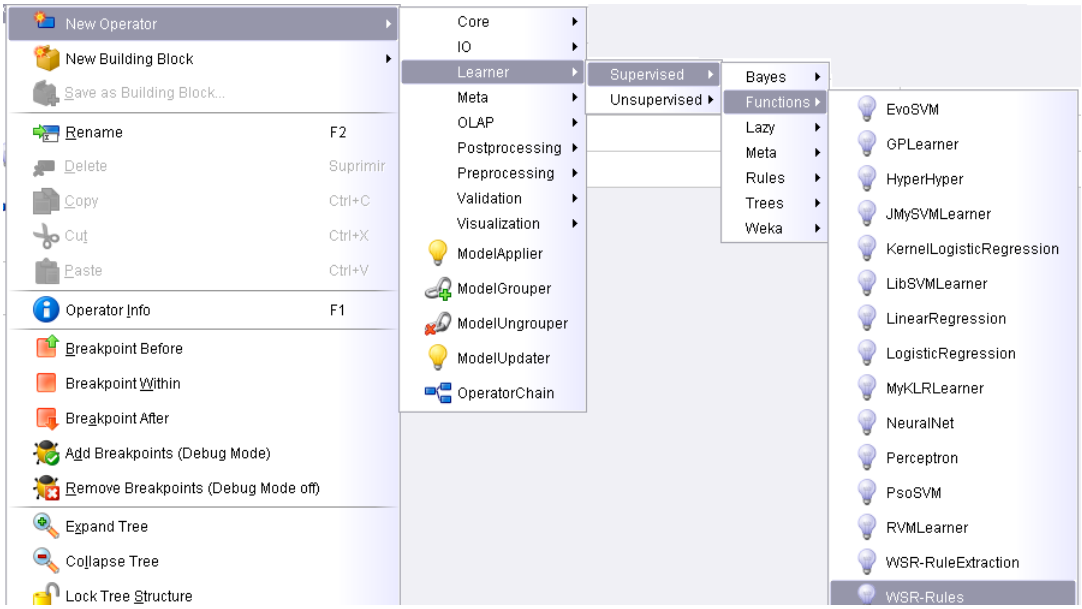

Figura A.7: Operador WSR-Rules

El operador *"WSR-Rules"* recibe como parámetros:

a) Modelo de Red Neuronal previamente entrenado "MultilayerPerceptron"

b) Distancia de agrupamiento para las neuronas de la capa ocultas

c) Distancia de agrupamiento para las neuronas de la capa de salida

d) Ruta de salida donde se guardan las reglas obtenidas

Resultado de la ejecución del operador, se presentan las reglas obtenidas a partir de la red neuronal entrenada como se aprecia en la figura A.8

Figura A.8: Reglas Obtenidas a partir de la Red Neuronal

En la figura A.9, se presenta todo el modelo construido sobre la interfaz gráfica de RapidMiner, el cual consta de:

1. Un proceso de selección de datos
2. Un proceso de entrenamiento de la Red Neuronal
3. Un proceso de almacenamiento del modelo de Red Neuronal entrenado
4. Un proceso de extracción de reglas.

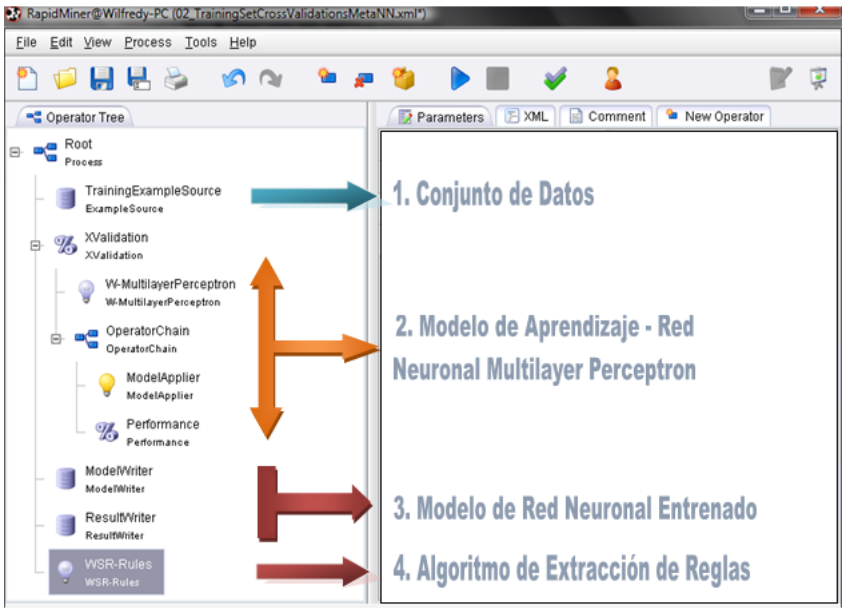

Figura A.9: Modelo de Detección - Fase de Entrenamiento y Test

A.2.3. Modelo de Validación

El modelo implementado en RapidMiner para la validación del modelo previamente entrenado, empleo los siguientes operadores:

1. *"ExampleSource"*. Este operador permite definir el conjunto de datos sobre el cual se va validar el modelo, para ello se debe especificar los datos a cargar y la metadata de los mismos.

2. *"ModelLoader"*. Este operador permite cargar el modelo de aprendizaje(Red Neuronal) previamente entrenado al igual que sus parámetros de configuración.

3. *"ModelApplier"*. Este operador ayuda a determinar el desempeño del modelo de aprendizaje almacenado(MultilayerPerceptron) contra el conjunto de datos de entrada.

4. *"Performance"*. Este operador presenta los resultados de desempeño del algoritmo en términos de exactitud, precisión,recall, error y curva ROC, como se aprecia en la figura A.10.

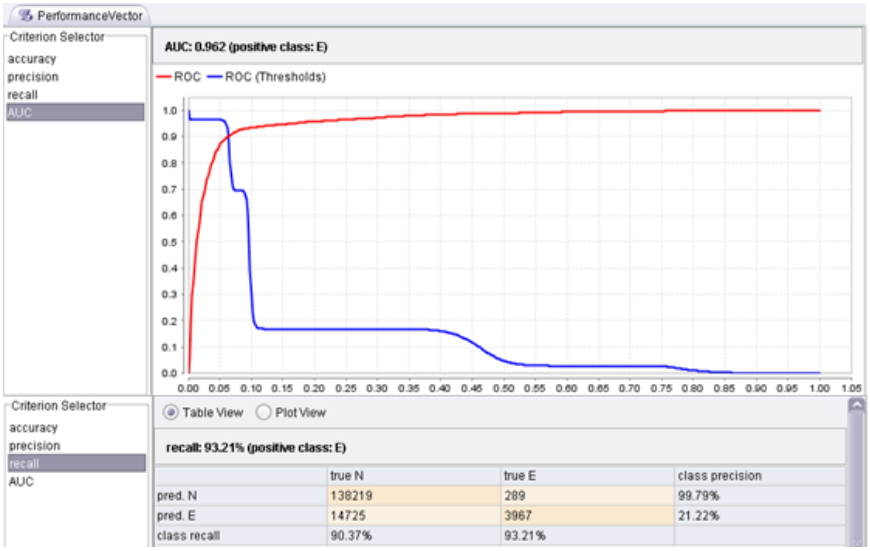

Figura A.10: Visualización desempeño del modelo de aprendizaje

En la figura A.11, se presenta el modelo completo construido sobre la interfaz gráfica de RapidMiner para hacer la validación del modelo de la sección anterior sobre todo el conjunto de datos.

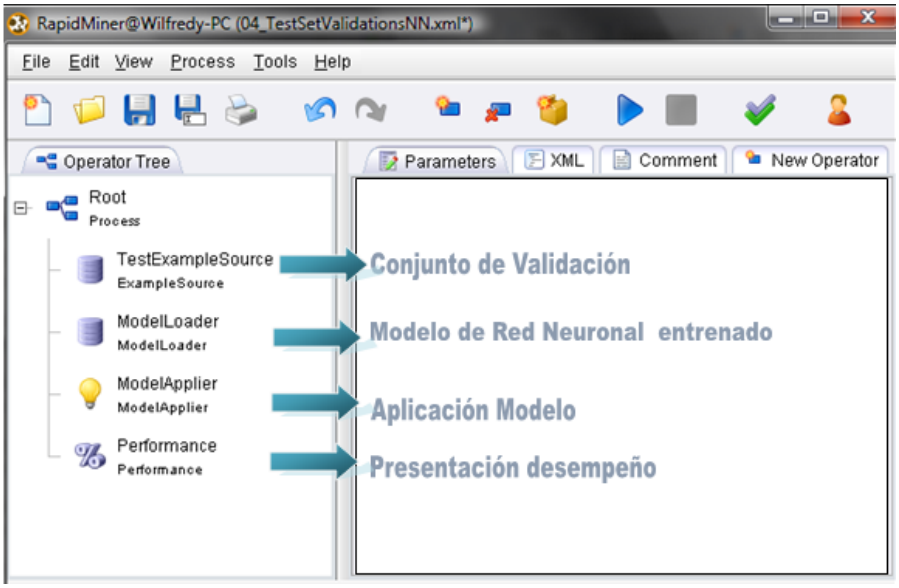

Figura A.11: Modelo de Detección - Fase de Validación

A.2.4. Modelo de Prueba de Hipótesis

El modelo implementado en RapidMiner para la prueba de hipótesis , empleo los siguientes operadores:

1. *"ExampleSource*". Este operador permite definir el conjunto de datos sobre el cual se va validar el modelo, para ello se debe especificar los datos a cargar y la metadata de los mismos.

2. *"XValidation*". Con este operador se define el número de validaciones cruzadas a aplicar sobre el proceso, al igual que el tipo de muestreo aplicado a cada validación cruzada.

   Embebido en este operador se define los siguientes subprocesos:

   a) *"ModelLoader"*. Este operador permite cargar el modelo de aprendizaje previamente entrenado.

   b) *"OperatorChain"*. Este operador ayuda a construir el proceso de evaluación de desempeño del operador de aprendizaje. Embebido en este operador se incluye los siguientes subprocesos:

   - *"ModelAplier*". Este operador ayuda a determinar el desempeño de la salida del algoritmo de aprendizaje
   - *"ClassificationPerformance*": Este operador presenta los resultados de desempeño del algoritmo en términos de exactitud, precisión,recall, error y curva ROC
   - "*ProcessLog*". En el contexto del modelo diseñado, este operador brinda la facilidad de almacenar los resultados de cada iteración del error, recall y exactitud como se muestra en la figura A.12

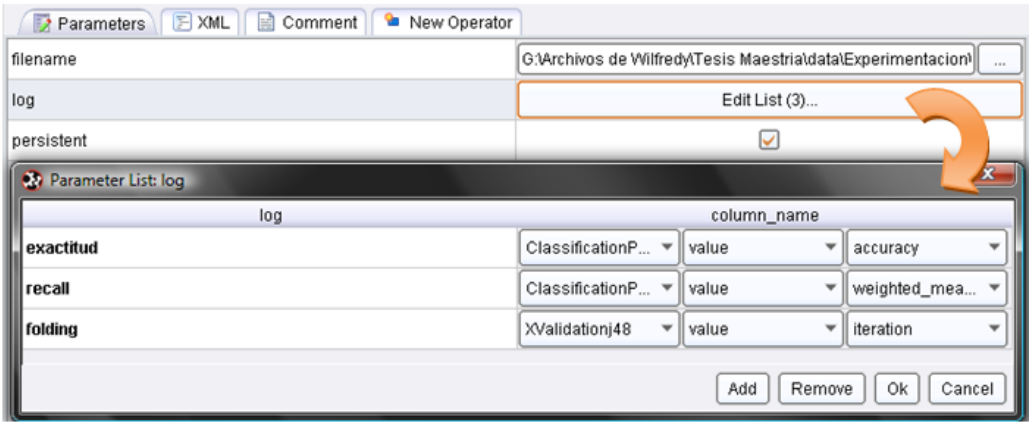

Figura A.12: Parámetros operador ProcessLog

3. *"T-Test"*. Este operador me permite determinar la probabilidad de una hipótesis nula, en la que el estadístico de prueba sigue una distribución t-Student. Se fija como parámetro el valor de alfa.

4. *"Anova"*. Este operador permite hacer una análisis de varianza, que determina si la probabilidad de la hipótesis nula es cierta o no. Se fija como parámetro el valor de alfa.

En el modelo de prueba de hipótesis, el operador del numeral 2 (*XValidation)*, se debe repetir por cada modelo de clasificación a comparar, para el caso, se tiene uno para la red neuronal, el árbol de decisión y el modelo probabilístico de Naive Bayes como se muestra en la figura A.13.

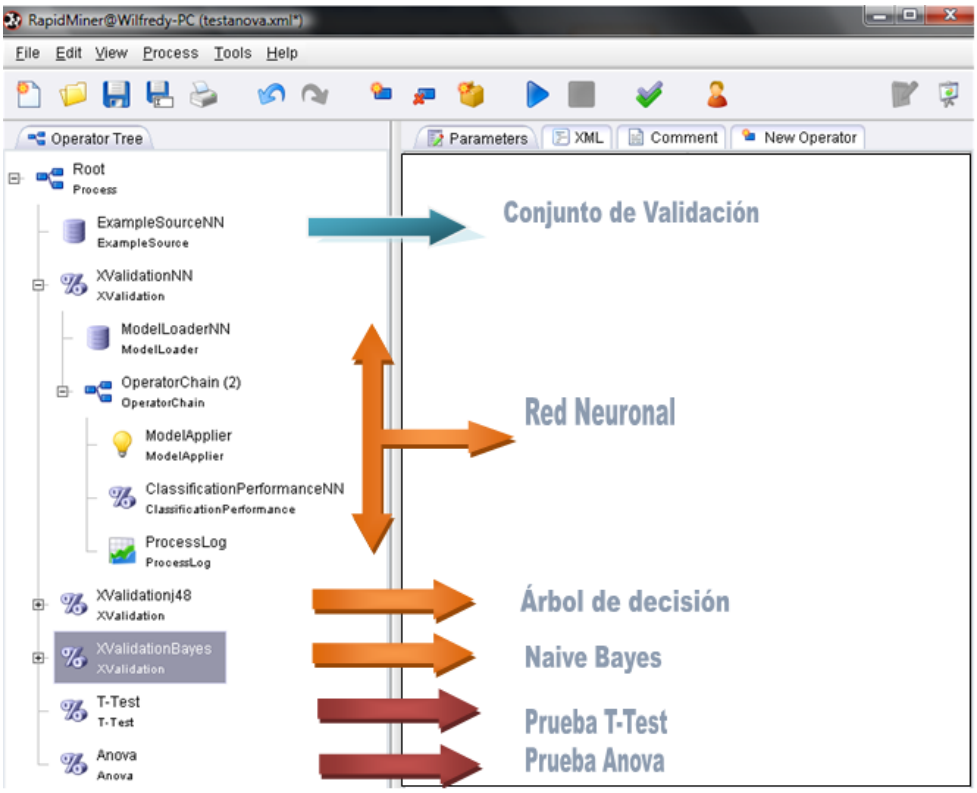

Figura A.13: Modelo de prueba de hipótesis

A.3. Instalación y Adición del operador de aprendizaje WSR-Rules

En este anexo se presentan los pasos que se deben seguir para instalar el aplicativo RapidMiner v 4.2 y la forma de adicionar el nuevo operador de extracción de reglas ”WSR-Rules” basado en el algoritmo ”M of N”.

A.3.1. Pre-requisitos

En este punto se definen los recursos mínimos requeridos tanto en hardware como en software para una correcta instalación y puesta en marcha de la aplicación.

Hardware

- Procesador Pentium III de 756 Mhz en adelante
- Disco Duro 200 Megas de capacidad de almacenamiento.
- Memoria Física 756 MB como mínimo Se recomienda utilizar más memoria en caso de utilizar bases de datos muy grandes.
- Tarjeta de vídeo SVGA
- Monitor SVGA (Resolución de 800 X 600)
- Accesorios tales como mouse, teclado.

Software

- Sistema operativo Windows XP, Vista o Windows 7.
- JRE(Java Runtime Environment) versión 5.0

A.3.2. Instalación RapidMiner

1. En la carpeta *Ejecutable* del CD, ejecute el archivo *rapidminer-4.2-community-windows.exe*, lo cual despliega la siguiente figura:

Figura A.14: RapidMiner - Instalación paso 1

2. De clic en el botón "Next >", para iniciar la instalación

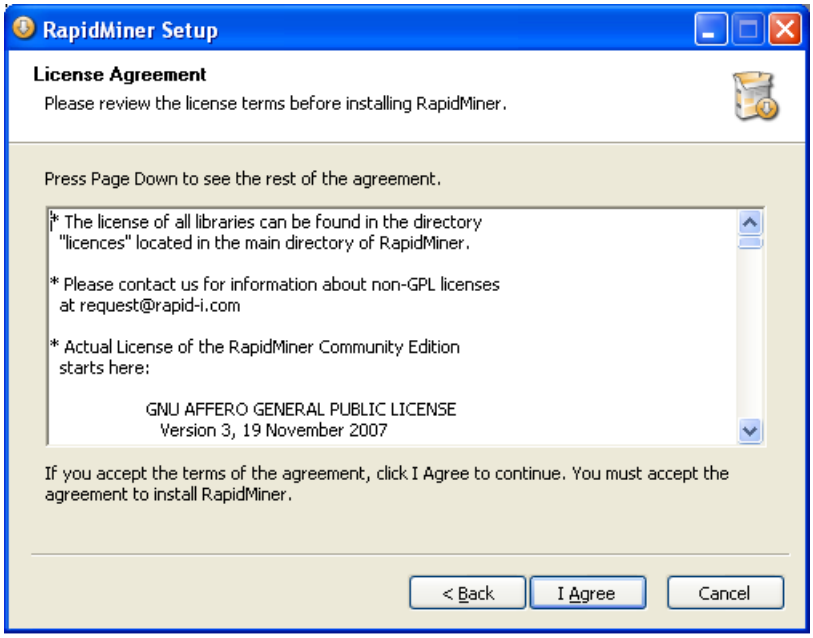

Figura A.15: RapidMiner - Instalación paso 2

3. De clic en el botón "I Agree", para aceptar los términos de la licencia.

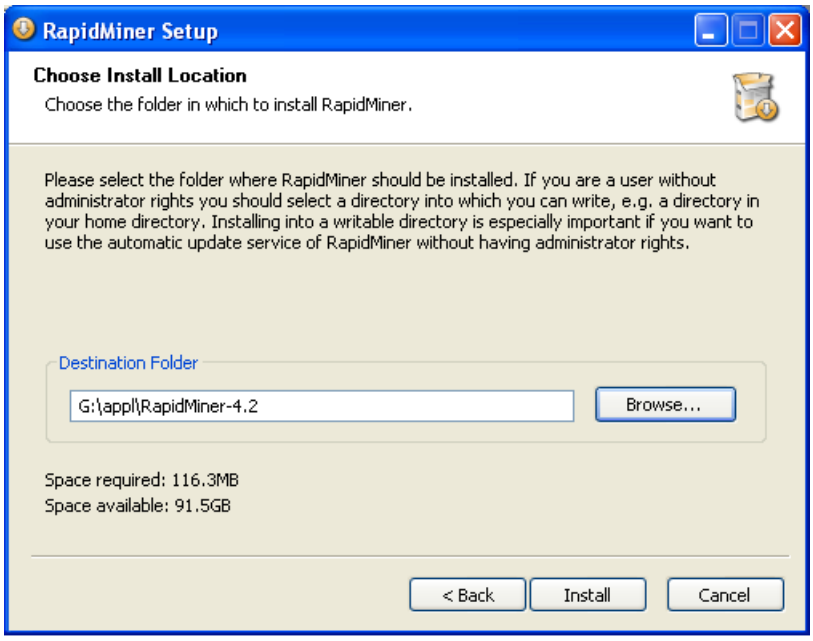

Figura A.16: RapidMiner - Instalación paso 3

4. Seleccione la carpeta destino donde se instala el aplicativo y clic en el botón "Install" para iniciar la instalación de los componentes de RapidMiner

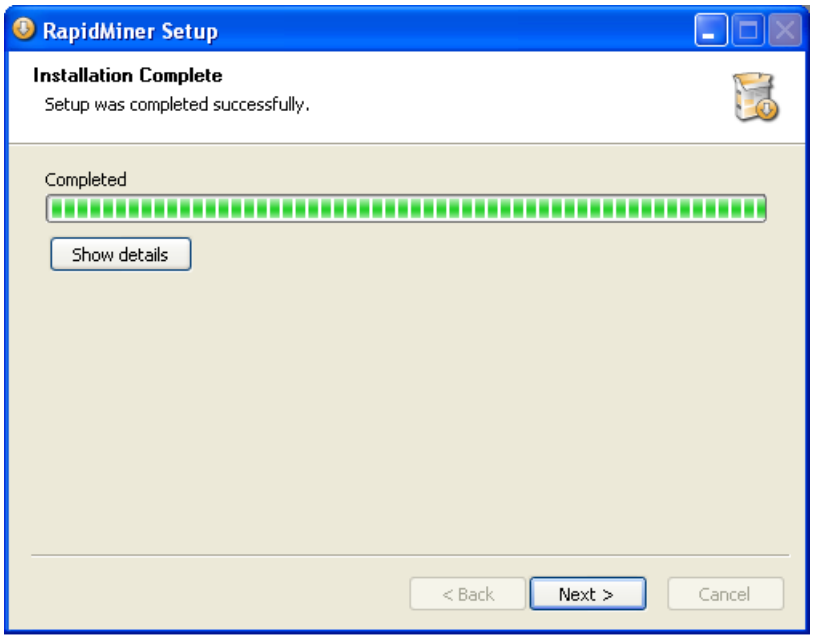

Figura A.17: RapidMiner - Instalación paso 4

5. De clic en el botón ”Next” para continuar la instalación

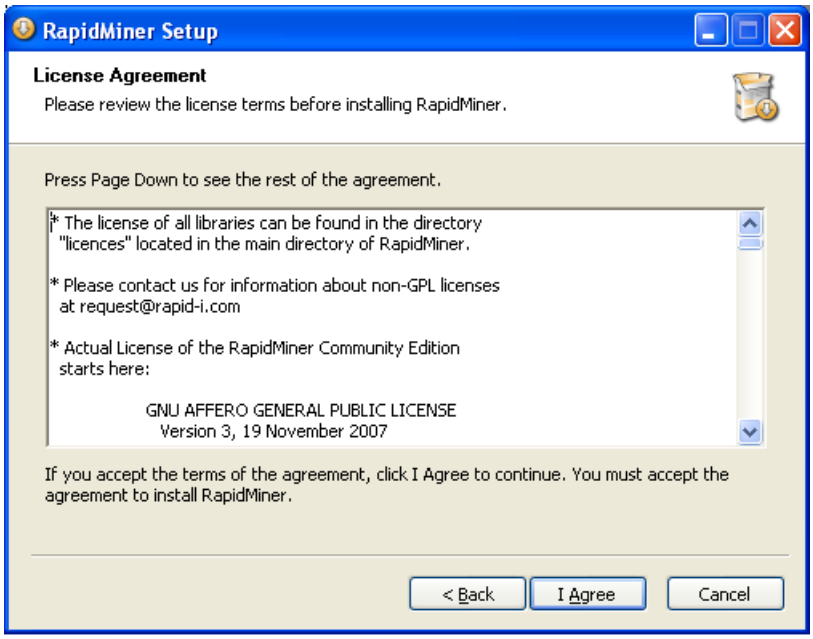

Figura A.18: RapidMiner - Instalación paso 5

6. De clic en el botón ”Finish” para finalizar la instalación

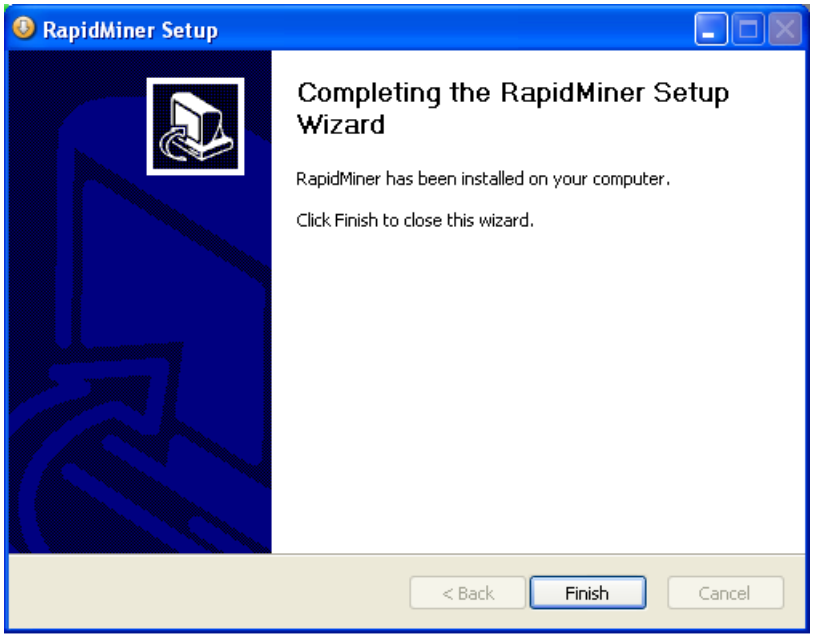

Figura A.19: RapidMiner - Instalación paso 6

Uva vez finalizado el proceso de instalación, la estructura de directorios de RapidMiner deber ser como se muestra en la siguiente figura A.20.

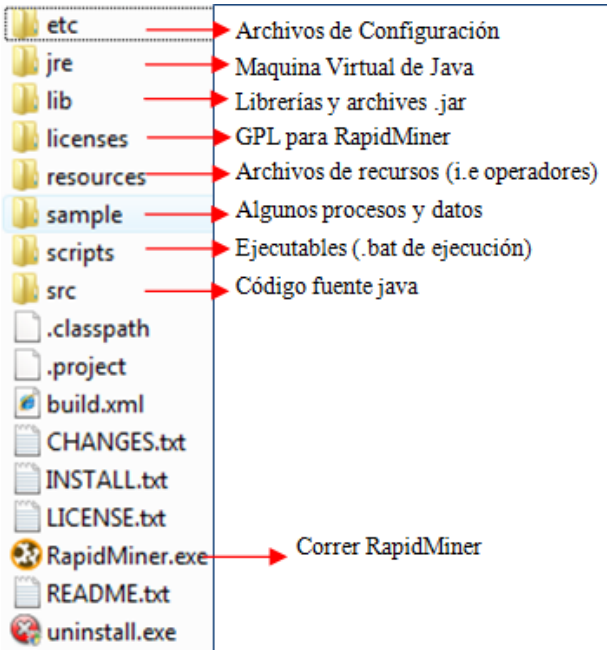

Figura A.20: Estructura de carpetas RapidMiner

A.3.3. Adición Operador WSR-Rules

Una vez completada la instalación de RapidMiner, siga los siguientes pasos para adicionar el operador de aprendizaje "*wsr-rules*"

1. *Adición de librerías.* En la carpeta *Ejecutable* del CD, copie los archivos de la carpeta .*lib* a la carpeta .*lib* de instalación de RapidMiner

2. *Actualización archivo de operadores(operadores.xml).* En la carpeta *Ejecutable* del CD, copie los archivos de la carpeta .*resource* a la carpeta .*resource* de instalación de RapidMiner

3. *Archivo classpath.* En la carpeta *Ejecutable* del CD, copie el archivo *.classpath* a la ruta de instalación de RapidMiner

4. *Fuentes java wsr-rules*. En la carpeta *Ejecutable* del CD, copie los archivos de la carpeta *.\scr* a la carpeta *.\scr* de instalación de RapidMiner

Uva vez finalizado el proceso anterior, para verificar la correcta adición del operador haga:

1. Ingrese a la interfaz gráfica de RapidMiner, ejecutando el archivo "RapidMiner.exe" y cree un nuevo proyecto:

2. Sitúese en la raíz del árbol de procesos y haga clic derecho para crear un nuevo operador desde la opción *"New Operator -> Leraner -> Supervised-> Functions ->WSR-Rules"* como se muestra en la figura A.21.

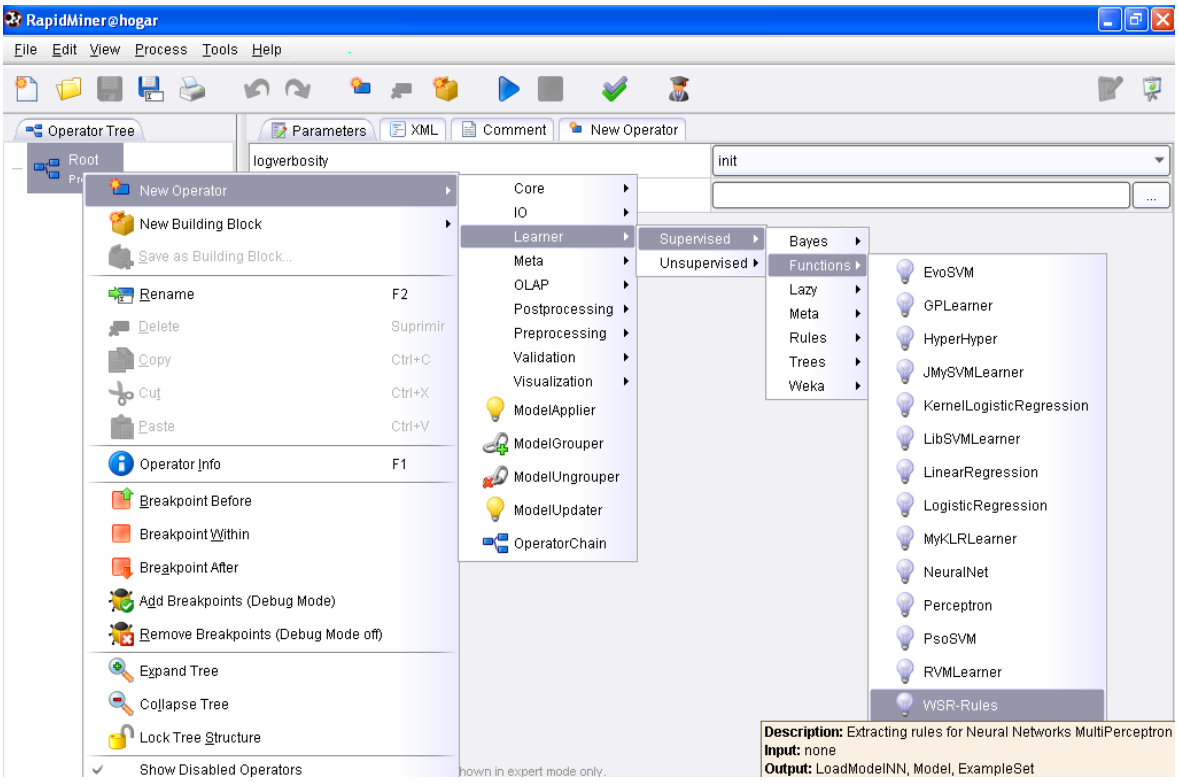

Figura A.21: Operador de aprendizaje WSR-Rules

Anexo B

Implementación del Algoritmo "M of N"

Diagramas de clase y estados UML

El código para el algoritmo "M of N" se desarrollo sobre el paquete java wsr.extractingrules y se adiciono como un operador de aprendizaje a la herramienta de RapidMiner. Este algoritmo se implemento en los siguientes 4 sub-paquetes:

- *wsr.extractingrules.mofn.clustering.* Este paquete implementa todo la etapa de agrupamiento.
- *wsr.extractingrules.mofn.ruleextractor.* Este paquete implementa toda la fase de extracción inicial de reglas en términos de nodos ocultos y nodos de salida, al igual que la extracción y poda de las reglas finales en términos de entradas y salidas.
- *wsr.extractingrules.model.* Este paquete implementa todo el modelo de datos sobre el cual se almacena la información de la red neuronal entrenada.
- *wsr.extractingrules.util.* Este paquete es un utilitario de funciones requeridas en el algoritmo como: cálculo de la distancia euclidiana y procesos de permutación de de nodos

para armar reglas.

La clase principal del algoritmo ”M of N” es ”LoadModelNN”, la cual permite enlazar las clases de la herramienta RapidMiner y Weka con las clases propias de la implementación del algoritmo. El diseño garantiza una fácil implementación de otros métodos de agrupamiento o reformulación del algoritmo de extracción de reglas. Todas las clases utilizadas en la implementación se muestran en los diagramas de clases UML de las figura C.1, C.2, C.3, C.4 y C.5.

De igual forma, en la figura C.6, se muestra el diagrama UML de estados para el proceso de extracción de reglas dado un modelo de Red entrenada ”Multilayer Perceptron”.

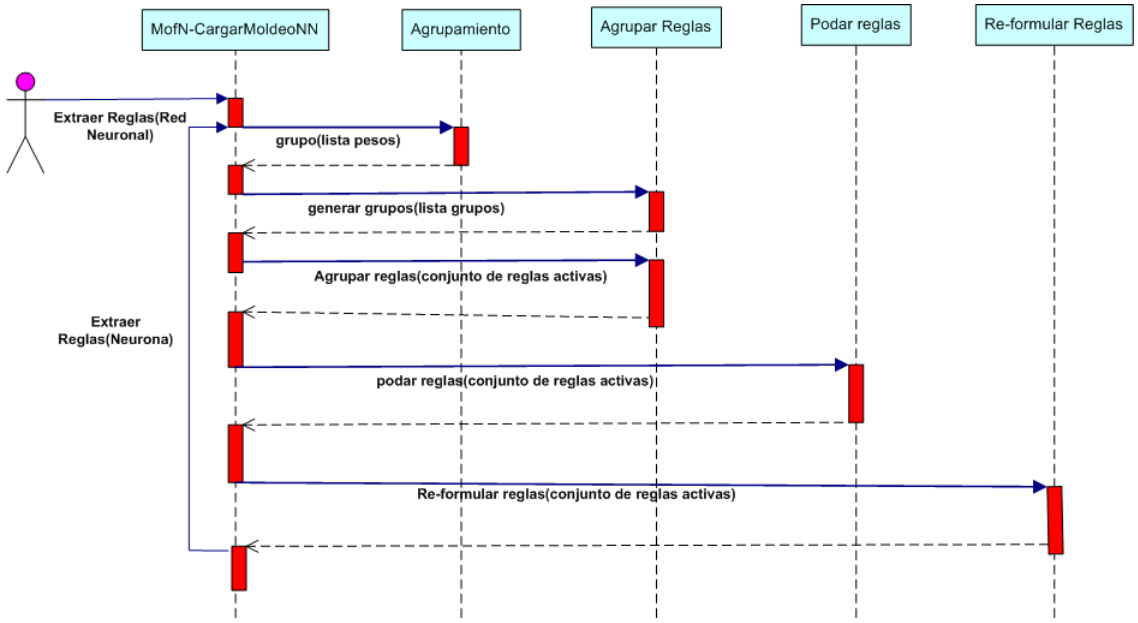

Figura B.6: Diagrama de estados UML -Algoritmo M of N

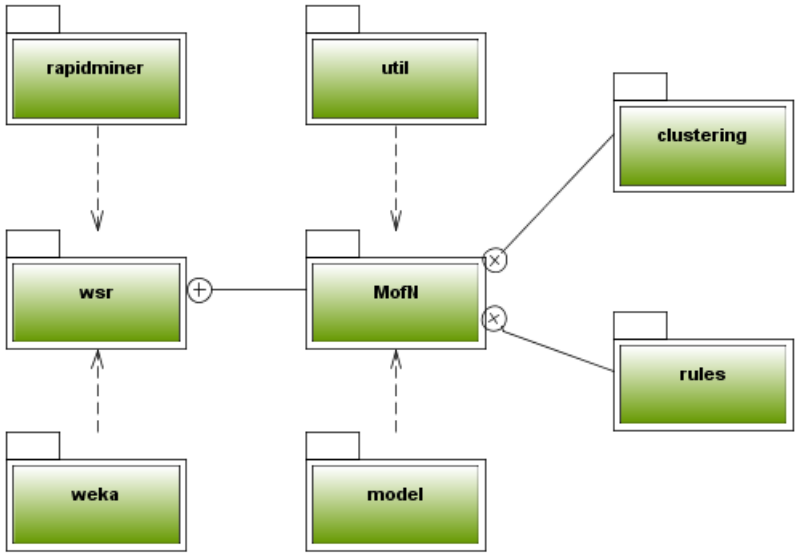

Figura B.1: Diagrama de paquetes UML- Implementación algoritmo M of N

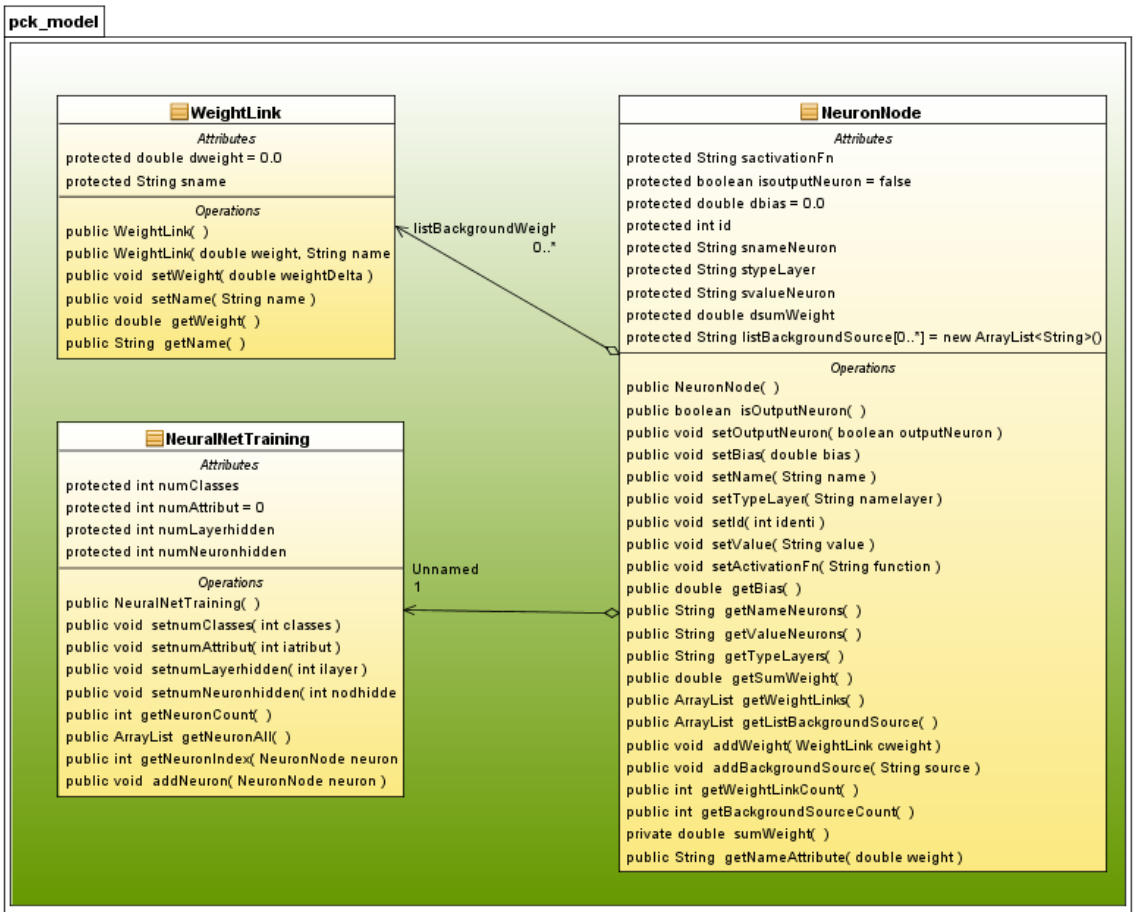

Figura B.2: Diagrama de clases UML - Paquete Modelo de datos Red Neuronal entrenada

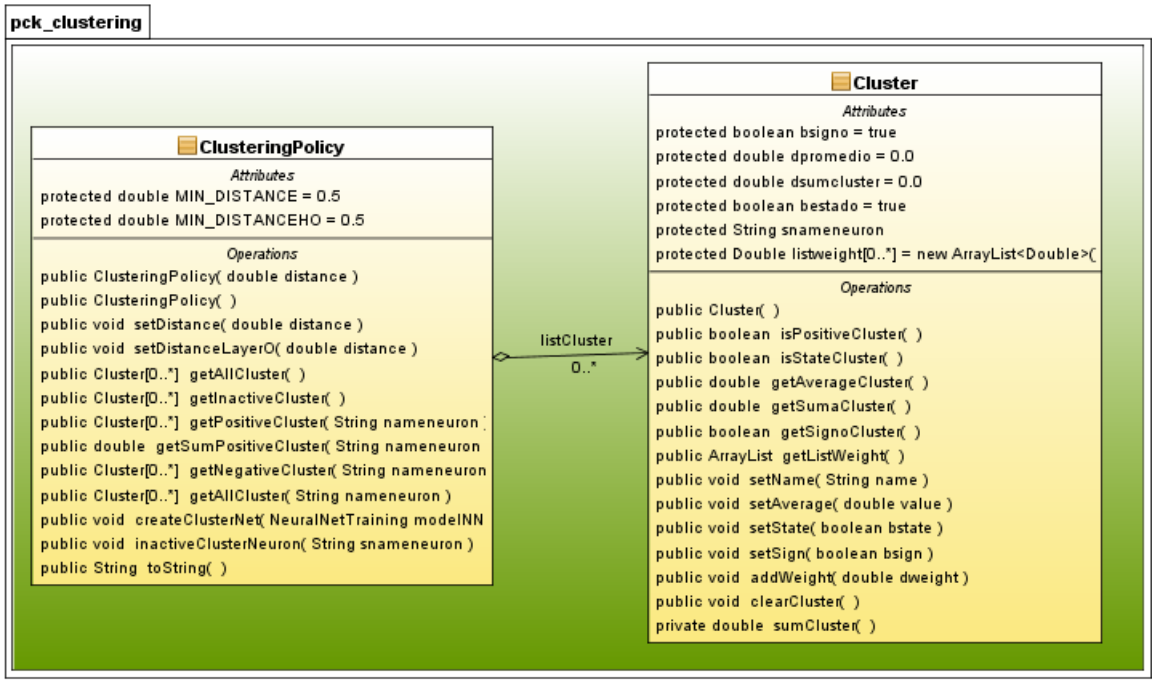

Figura B.3: Diagrama de clases UML - Paquete de Agrupamiento algoritmo M of N

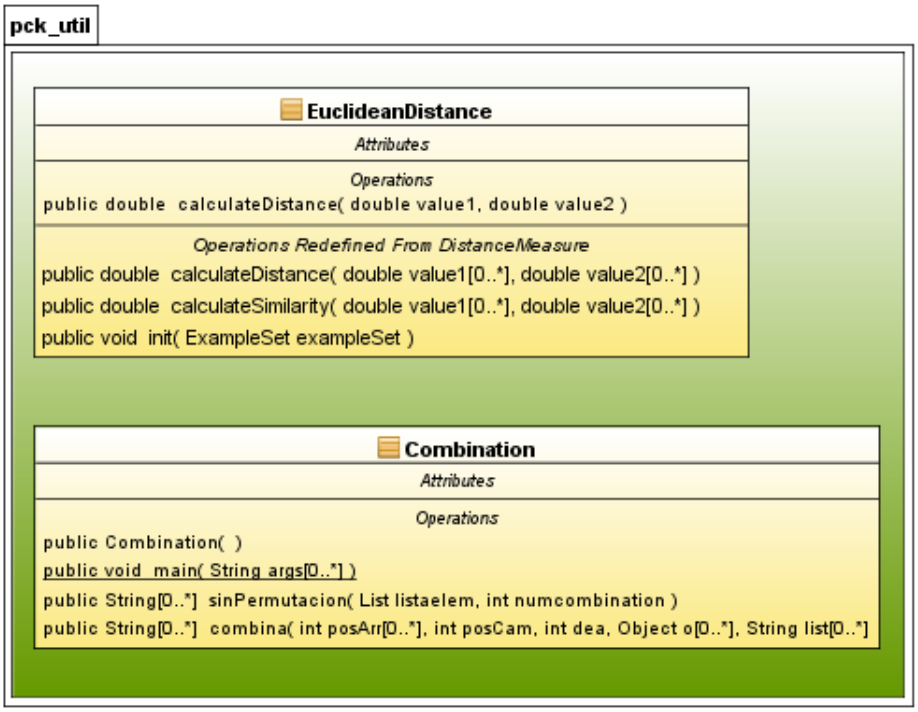

Figura B.4: Diagrama de clases UML - Paquete de utilitarios algoritmo M of N

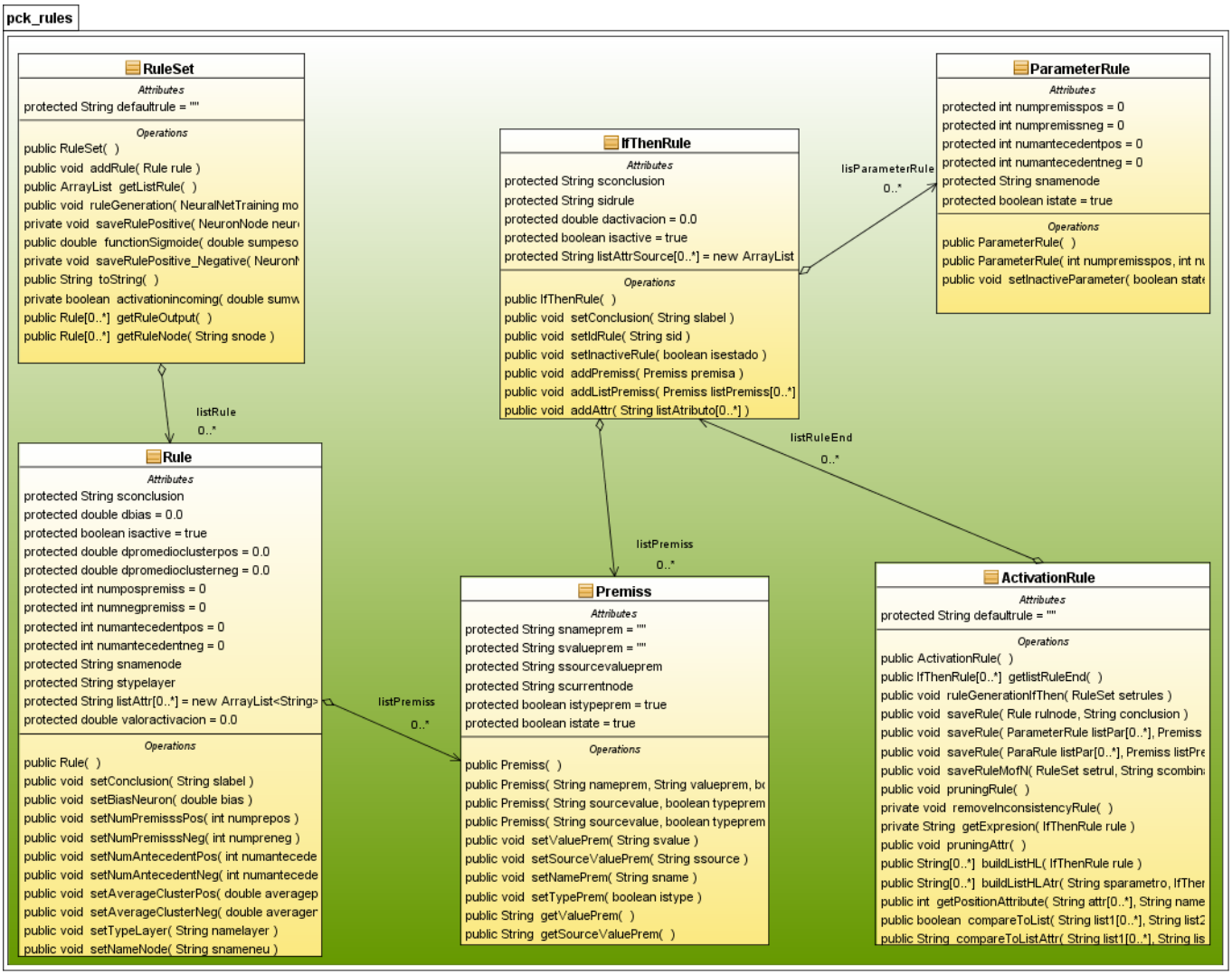

Figura B.5: Diagrama de clases UML - Paquete de Extracción de Reglas algoritmo M of N

Anexo C

Glosario

Almacenamiento de datos (Data Warehousing): El almacenamiento de datos se define como un proceso de organización de grandes cantidades de datos de diversos tipos "guardados" en la organización con el objetivo de facilitar la recuperación de la misma con fines analíticos. Por supuesto, el almacenamiento de datos tiene un gran importancia en el proceso de minería de datos pues en cierta medida, permite la recuperación o al menos la referencia a determinados conjuntos de datos de importancia para un proceso de toma de decisión dado. En la actualidad existe gran variedad de sistemas comerciales para el almacenamiento de datos entre los que se destacan Oracle, Sybase, MS SQL Server, entre otros.

Análisis exploratorio de datos (Exploratory Data Analysis): Las técnicas de análisis exploratorio de datos juega un papel muy importante en la minería de datos. Las mismas tienen como objetivo determinar las relaciones entre las variables cuando no hay o no está totalmente definida la naturaleza de estas relaciones. Las técnicas exploratorias tienen un fuerte componente computacional abarcando desde los métodos estadísticos simples a los más avanzados como las técnicas de exploración de multivariables diseñadas para identificar patrones en conjunto de datos multivariables. Entre las técnicas estadísticas sencillas se incluyen el estudio de distribuciones de las variables, estudio de correlaciones entre matrices, tablas de contingencias, entre otros. Por su parte, entre las técnicas más complejas se incluyen el Análisis de Factores, el Análisis de Grupos, el Escalado Multidimensional, etcétera.

Antecedente: Cuando una asociación entre dos variables es definida, el primer artículo (o el del lado de la mano izquierda) se llama el antecedente.

API: Es una interface de programa de aplicación. Cuando un sistema del software ofrece un API, proporciona los medios por los que los programas escritos fuera del sistema pueden unirse con el sistema para realizar funciones adicionales. Por ejemplo, un software para minería de datos puede tener un API que permite a los programas hechos por el usuario realizar tareas como extraer datos realizar análisis estadístico adicional, crear mapas especializados, generar un modelo, o hacer una predicción de un modelo. Aprendizaje: Modelos de entrenamiento basados en datos existentes.

Aprendizaje No Supervisado: Este término se refiere a la colección de técnicas donde se definen agrupaciones de los datos sin el uso de una variable dependiente.

Aprendizaje Supervisado: La colección de técnicas donde el análisis usa una variable dependiente conocida. Se dirige a todas las técnicas la regresión y clasificación.

Árbol de decisión: Estructuras de forma de árbol que representan conjuntos de decisiones. Estas decisiones generan reglas para la clasificación de un conjunto de datos.

Asociaciones: Un algoritmo de asociación crea reglas que describen eventos que a menudo han ocurrido juntos. Backpropagation: Un método de entrenamiento usado para calcular pesos en una red neuronal de datos

Base de datos multidimensional: Base de datos diseñada para procesamiento analítico on-line (OLAP). Estructurada como un hipercubo con un eje por dimensión.

CART Árboles de clasificación y regresión: Una técnica de árbol de decisión usada para la clasificación de un conjunto da datos. Provee un conjunto de reglas que se pueden aplicar a un nuevo (sin clasificar) conjunto de datos para predecir cuáles registros darán un cierto resultado. Segmenta un conjunto de datos creando 2 divisiones. Requiere menos preparación de datos que CHAID. CART es un método de fraccionamiento de las variables independientes en grupos pequeños para aplicar una función constante a los grupos pequeños de datos. En árboles categóricos, la función constante es una que toma un grupo pequeño de valores (Ej.,

Y o N, bajo o medio o alto). En árboles de la regresión, el significado del valor depende del pequeño grupo de datos

CHAID Detección de interacción automática de Chi cuadrado: Una técnica de árbol de decisión usada para la clasificación de un conjunto da datos. Provee un conjunto de reglas que se pueden aplicar a un nuevo (sin clasificar) conjunto de datos para predecir cuáles registros darán un cierto resultado. Segmenta un conjunto de datos utilizando tests de chi cuadrado para crear múltiples divisiones. Antecede, y requiere más preparación de datos, que CART.

Chi-Cuadrado: Una estadística que evalúa la probabilidad del comportamiento de los datos. En minería de datos, es normalmente usada para encontrar subconjuntos homogéneos para ajustar árboles categóricos como en CHAID.

Clasificación: Proceso de dividir un conjunto de datos en grupos mutuamente excluyentes de tal manera que cada miembro de un grupo esté lo "más cercano" posible a otro, y grupos diferentes estén lo "más lejos" posible uno del otro, donde la distancia está medida con respecto a variable(s) específica(s) las cuales se están tratando de predecir. Por ejemplo, un problema típico de clasificación es el de dividir una base de datos de compañías en grupos que son lo más homogéneos posibles con respecto a variables como "posibilidades de crédito" con valores tales como "Bueno" y "Malo".

Clustering (agrupamiento): Proceso de dividir un conjunto de datos en grupos mutuamente excluyentes de tal manera que cada miembro de un grupo esté lo "más cercano" posible a otro, y grupos diferentes estén lo "más lejos" posible uno del otro, donde la distancia está medida con respecto a todas las variables disponibles.

Confianza: Confianza de la regla "B dado A" más probablemente es una medida de probabilidad de que ocurra B si A ocurrió. Se expresa como un porcentaje, con 100 % siempre significando que B ocurre si A ha ocurrido. Estadísticamente se refieren a la probabilidad condicional de B dada A

Consecuente: Cuando una asociación entre dos variables se define, el segundo artículo (o el lado diestro) se llama el consecuente.

Continuo: Los datos continuos pueden tener algún valor en un intervalo de números reales. Es decir, el valor no tiene que ser un entero. Continuo es el opuesto de discreto o categórico.

Criterio De Optimización: Una función positiva de la diferencia entre las predicciones y los datos estimados que son escogidos para perfeccionar la función o el criterio. Los mínimos cuadrados y probabilidad del máximo son ejemplos.

Data cleaning: Proceso de asegurar que todos los valores en un conjunto de datos sean consistentes y correctamente registrados.

Data Mining: La extracción de información predecible escondida en grandes bases de datos.

Data Warehouse: Sistema para el almacenamiento y distribución de cantidades masivas de datos.

Datos: Los valores coleccionados a través del registro ya sean guardados, observados, o medidos, típicamente organizados para análisis o fabricación de decisiones. Más simplemente, los datos son hechos, transacciones y figuras.

Datos anormales: Datos que resultan de errores o que representan eventos inusuales.

Datos Categóricos: Los datos categóricos se ajustan a un número pequeño de categorías discretas. Los datos categóricos son nominales como género o ciudad, u ordinales como alto, medio, o las temperaturas bajas.

Datos Externos: Datos no coleccionados por la organización, como datos disponibles en un libro de referencia, una fuente gubernamental o un banco de datos propio. ***Datos Interiores***: Datos coleccionados por una organización como operativos y datos del cliente.

Datos No-Aplicables: Valores perdidos que serían lógicamente imposibles (Ej., los varones embarazados).

Datos Perdidos: Valores de datos que pueden perderse porque no son moderados, eran desconocidos o estaban perdidos. Los métodos de Minería de Datos varían la manera en que tratan valores perdidos. Normalmente, ignoran los valores perdidos, u omiten cualquier archivo que contiene valores perdidos, o reemplaza valores perdidos con el modo o promedio, o infiere valores perdidos de valores existentes.

Dimensión: En una base de datos relacional o plana, cada campo en un registro representa una dimensión. En una base de datos multidimensional, una dimensión es un conjunto de entidades similares; por ej. una base de datos multidimensional de ventas podría incluir las dimensiones Producto, Tiempo y Ciudad. Cada atributo de un caso o ocurrencia en los datos minados. Guardado como un campo en un registro del archivo plano o una columna de la base de datos relacional.

Discreto: Un tipo de datos que tiene un grupo finito de valores. Discreto es el opuesto de continuo.

Entropía: Una manera de medir variabilidad. Una decisión que se toma por los datos que presentar menor entropía, es decir los datos menos variables

Estandarización: Una colección de datos numéricos es estandarizada substrayendo una medida de situación central y dividiendo por alguna medida de cobertura (como la desviación normal).

Exactitud (accuracy): La exactitud es un factor importante para evaluar el éxito de la minería de datos. Cuando la aplico a los datos, la exactitud se refiere a la proporción de valores correctos en los datos. Cuando aplico a modelos, la exactitud se refiere al grado de ajuste entre el modelo y los datos. Esto mide el posible error en las predicciones del modelo. Puesto que la exactitud no incluye información del costo, es posible para un modelo menos exacto ser más rentable.

Función De Activación: Es una función usada por un nodo para transformar datos de entrada de cualquier dominio de valores en un rango finito de valores. La idea original era aproximar los datos mas altos, y la función de activación asumió el valor 0 hasta que la entrada es bastante grande y el valor salta a 1.

GUI: Interface Gráfica de usuario.

Hoja: Es un nodo terminal o final en el árbol de decisión.

KDD: Descubrimiento de conocimiento en bases de datos.

Matriz De Confusión: Una matriz de confusión muestra el valor actual contra valores de la clase predecida. No sólo muestra qué bien el modelo predice, también indica donde pudo fallar el programa. Método De Minería De Datos: Los procedimientos y algoritmos diseñados para analizar los datos en bancos de datos.

Minería De Datos: Una actividad de extracción de información cuya meta es descubrir hechos escondidos contenidos en bancos de datos. Usando una combinación de aprendizaje de máquina, análisis estadístico, técnicas modeladas y tecnologías del banco de datos, Minería de datos encuentra los modelos y las relaciones sutiles en datos e infiere reglas que permiten la predicción de resultados futuros. Las aplicaciones típicas incluyen segmentación del mercado, perfil del cliente, el descubrimiento del fraude, evaluación de promociones, y análisis de riesgo de crédito.

Mínimos Cuadrados: El método más común de entrenamiento (estimando) por pesos (parámetros) del modelo escogiendo los pesos que minimizan la suma de la desviación cuadrada de los valores del modelo de observación de los datos.

Modelo: Una función importante de minería de datos es la producción de un modelo. Un modelo puede ser descriptivo o predictivo. Un modelo descriptivo entiende procesos o conductas subyacentes. Por ejemplo, un modelo de asociación describe la conducta del consumidor. Un modelo predictivo es una ecuación o grupo de reglas que hacen posible predecir un valor inadvertido (la variable dependiente o rendimiento) de otros, valores conocidos (variables independientes o entradas).

Nodo: Un punto de decisión en un árbol de decisión. También, un punto en una red neuronal que combina entradas de otros nodos y produce salidas a través de la aplicación de una función de activación.

Outlier: Un ítem de datos cuyo valor cae fuera de los límites que encierran a la mayoría del resto de los valores correspondientes de la muestra. Puede indicar datos anormales. Deberían ser examinados detenidamente; pueden dar importante información.

Precisión: Es una medida de cómo la variable estimada estaría encima de otros grupos de

datos similares. Una estimación muy precisa sería una que no varía mucho respecto de los grupos de datos diferentes. La precisión no mide exactitud.

Promedio: Valor promedio aritmético de una colección de datos numéricos.

Prueba de Datos (Test Data): Un grupo de datos independiente de los datos de entrenamiento, usados para comparar el comportamiento del modelo

Prueba de Error (Test Error): La estimación de error se basa en la diferencia entre las predicciones de un modelo con un grupo de datos prueba y los valores observados en los datos de la prueba cuando el grupo de datos de prueba no se usó en el entrenamiento del modelo.

Prunning (Quitar): Eliminar subárboles con niveles bajos en un árbol de decisión

Rango: El rango de los datos es la diferencia entre el valor del máximo y el valor mínimo.

Redes neuronales-NN (Neural Networks): Las redes neuronales son técnicas analíticas que permiten modelar el proceso de aprendizaje de una forma similar al funcionamiento del cerebro humano, básicamente, la capacidad de aprender a partir de nuevas experiencias. Estas técnicas han tenido un desarrollo impresionante en la última década, con aplicaciones tanto a la medida como generales (comúnmente llamados shell) y tienen como objetivo fundamental sustituir la función de un experto humano.

Ruido: La diferencia entre el modelo y sus predicciones. A veces el dato es llamado ruidoso cuando contiene errores como muchos valores perdidos o incorrectos o cuando hay columnas extrañas.

Significancia: Una medida de probabilidad de cuanto apoyan los datos un cierto resultado (normalmente de una prueba estadística). Se dice que la significancia de un resultado es .05, significa que hay sólo una .05 probabilidad que el resultado puda pasar por casualidad exclusivamente

UML: Es un estándar de lenguaje de diseño en el campo de la igeniería de software(por sus siglas en inglés Unified Modeling Language)

Validación Cruzada: Es un método de estimar la exactitud de una clasificación o modelo

de regresión. El grupo de datos es dividido en varias partes, y a cada parte se le aplica el modelo y se compara con los resultados de los otros grupos.

Variable Dependiente: Las variables dependientes (salidas) de un modelo son aquellas que se obtienen por medio de ecuaciones o reglas del modelo aplicadas a variables independientes

Variable Independiente: Las variables independientes (entradas) de un modelo.

Validación: El proceso de comprobación los modelos con un grupo de datos diferente del grupo de datos de entrenamiento.

Varianza: La medida estadística normalmente usada de dispersión. El primer paso es cuadrar las desviaciones de los datos de su valor medio. Entonces el promedio de las desviaciones cuadradas se calcula para obtener una medida global de variabilidad.