UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

ÁREA ACADÉMICA DE COMPUTACIÓN Y ELECTRÓNICA

Modelo Híbrido Predictivo y de Recomendación con Técnicas de Minería de Datos e Inteligencia Artificial

TESIS

Que para obtener el grado de

Doctor en Ciencias Computacionales

P r e s e n t a:

M.C.C. René Cruz Guerrero

Directora: Dra. María de los Ángeles Alonso Lavernia

Coasesora: Dra. Anilú Franco Árcega

Mineral de la Reforma, Hidalgo, marzo de 2018

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO
Instituto de Ciencias Básicas e Ingeniería
Institute of Basic Sciences and Engineering
Área Académica de Computación y Electrónica
Computer Science and Electronics Department

Mineral de la Reforma, Hgo., a 21 de marzo de 2018
Número de control: ICBI-AACYE/442/2018
Asunto: Impresión de tesis.

M.C.C RENÉ CRUZ GUERRERO
PRESENTE

Por este conducto le comunico que el jurado asignado para la revisión de su trabajo de tesis titulado **"Modelo Híbrido Predictivo y de Recomendación con técnicas de Minería de Datos e Inteligencia Artificial"**, dirigido por la Dra. María de los Ángeles Alonso Lavernia y la Dra. Anilu Franco Árcega que, para obtener el grado de Doctorado en Ciencias Computacionales, fue presentado por Usted, ha tenido a bien en reunión de sinodales **autorizarlo para impresión.**

A continuación, se integran las firmas de conformidad de los integrantes del jurado.

Dr. Omar López Ortega	Presidente	UAEH	
Dr. Ruslan Faizovich Gabbasov	Secretario	UAEH	
Dra. María de los Ángeles Alonso Lavernia	Vocal	UAEH	
Dra. Anilu Franco Árcega	Suplente	UAEH	

Agradeciendo su puntual asistencia, quedo a sus órdenes.

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

ATENTAMENTE.
"AMOR, ORDEN Y PROGRESO"

Dr. Omar López Ortega
Coordinador del DCC

Instituto de Ciencias Básicas e Ingeniería
Área Académica de Computación y Electrónica

c.c.p. Archivo OML/apl

Ciudad del Conocimiento
Carretera Pachuca - Tulancingo km. 4.5
Colonia Carboneras
Mineral de la Reforma, Hidalgo, México, C.P. 42184
Tel. +52 771 7172000 exts. 2250 y 2251
aacye_icbi@uaeh.edu.mx

www.uaeh.edu.mx

Índice

Índice de Cuadros

Capítulo 1. Marco Teórico

Capítulo 2. Estado del arte

Capítulo 5. Modelación de reglas de asociación de clase en esquema ontológico

Capítulo 7. Validación del modelo

Índice de Figuras

Capítulo 1. Marco Teórico

Capítulo 2. Estado del arte

Capítulo 3. Modelo propuesto

Capítulo 4. Generación de reglas de asociación

Capítulo 5. Modelación de reglas de asociación de clase en esquema ontológico

Capítulo 6. Selección del clasificador

Capítulo 7. Validación del modelo

Índice de Tablas

Capítulo 2. Estado del arte

Capítulo 4. Generación de reglas de asociación

Capítulo 5. Modelación de reglas de asociación de clase en esquema ontológico

Capítulo 6. Selección del clasificador

Capítulo 7. Validación del modelo

Resumen

El uso de modelos predictivos tiene gran importancia debido a que permiten aprender del pasado para pronosticar situaciones futuras, para esto existen varios enfoques de solución. La capacidad de este tipo de sistemas se debe a que se basan en un modelo que es considerado capaz de generalizar el conocimiento que ha aprendido para aplicarlo a nuevas situaciones.

Para crear este tipo de modelos, se utilizan diversas técnicas: de tipo Estadístico con métodos de regresión, de Minería de Datos que contiene métodos de diferentes enfoques como Lógico Combinatorio, Árboles de Decisión, Redes Neuronales y Máquinas de Soporte Vectorial entre otros y finalmente, las técnicas de Inteligencia Artificial cuya solución se basa en el conocimiento brindado por el experto.

El presente trabajo propone un modelo híbrido de predicción que además de usar técnicas predictivas y descriptivas de Minería de Datos, aplica técnicas de Inteligencia Artificial sobre el conocimiento extraído de los datos. Esta solución permitirá enriquecer los resultados que proporciona un sistema predictivo convencional ya que al brindar el resultado de la predicción, proporciona conocimiento para identificar comportamientos particulares en los datos.

Las técnicas de Minería de Datos utilizadas en el modelo son la generación de reglas de asociación de clase y la clasificación. La primera técnica propone extraer conocimiento mediante la búsqueda de correlaciones entre los valores de los atributos, para ello se utiliza el algoritmo Top-K Rules Non Redundant (TNR), ya que después de evaluar su desempeño junto con otros métodos, resultó ser el método más eficiente para eliminar la redundancia. Respecto a la segunda técnica, se utiliza para clasificar nuevas instancias a partir de sus características. En este caso, se evaluó el desempeño de los algoritmos J48, Naive Bayes, KNN y MLP en bases de datos con diferentes características, obteniéndose un conjunto de criterios que sirven de apoyo para elegir entre estos, el mejor clasificador en correspondencia con las características de los datos en tratamiento.

Respecto a las técnicas de Inteligencia Artificial, se desarrolló un método para modelar un conjunto de reglas de asociación de clase en un esquema ontológico conformando una base de conocimientos que entre sus componentes, cuenta con un conjunto de axiomas para efectuar razonamiento.

Palabras Claves: Modelo predictivo, Minería de Datos, Inteligencia Artificial, Reglas de asociación de clases, modelo ontológico.

Abstract

The use of predictive models has great importance because they allow learning from the past to predict future situations, for this there are several techniques among which are Data Mining and Artificial Intelligence. The capacity of this type of systems is due to the fact that they work based on a model that is considered capable of generalizing the knowledge that has been learned to apply it to new situations.

To create this type of models, there are basically the following techniques: Statistical type with regression methods, Data Mining that contains methods of different approaches such as Combinatorial Logic, Decision Trees, Neural Networks and Vector Support Machines among others and finally, Artificial Intelligence techniques whose solution is based on the knowledge offered by the expert.

The present work proposes a hybrid model because in addition to using predictive and descriptive techniques of Data Mining, it applies Artificial Intelligence techniques on the knowledge extracted from the data. This combined solution enriches the results provided by a conventional predictive system, because in addition to providing the result of the prediction, it extracts knowledge through the generation of class association rules to identify particular behaviors in the data.

The Data Mining techniques used in the model are generation of class association rules and classification. The first technique proposes to extract knowledge through the search of correlations between the values of the attributes, for it the Top-K Rules Non Redundant (TNR) algorithm is used, since after evaluating its performance together with other methods, it turned out to be the method more efficient to eliminate redundancy. Regarding the second technique, it is used to classify new instances based on their characteristics. In this case, the performance of the algorithms J48, Naive Bayes, KNN and MLP in databases with different characteristics was evaluated, obtaining a set of criteria that serve as support to choose among these, the best classifier in correspondence with the characteristics of the data under treatment.

Regarding Artificial Intelligence techniques, a method was developed to model the set of class association rules generated in an ontological scheme that forms a knowledge base with necessary components to carry out the reasoning.

Keywords: Predictive model, Data Mining, Artificial Intelligence, Rules of class association, ontological model.

Introducción

Actualmente, cada vez son más las organizaciones que tratan de aprovechar la información disponible en sus bases de datos, con el objetivo de identificar riesgos y oportunidades con anticipación, cuya solución requiere del uso de técnicas de predicción.

El análisis predictivo es un área de la Minería del Datos que pretende extraer conocimiento que le permita al usuario predecir tendencias y patrones de comportamiento (Finlay, 2014).

La construcción de modelos predictivos ayuda a saber con antelación comportamientos o tendencias de situaciones, permitiendo la toma de decisiones proactivas y facilitando la solución de problemas que tradicionalmente ocupan demasiado tiempo en resolverse mediante el análisis de patrones ocultos en grandes bases de datos.

Los modelos predictivos permiten estimar valores futuros o desconocidos de variables objetivos o dependientes usando otras variables independientes o predictivas. Para lograr esta capacidad requieren de un conjunto de pruebas y de interacciones de entrenamiento para la generación del modelo.

En las últimas décadas, se han desarrollado diversos métodos analíticos aplicados en los modelos predictivos. La Estadística es la primera ciencia que históricamente extrae información de los datos, algunos métodos predictivos estadísticos son Regresión Lineal Múltiple, Regresión no Lineal, Regresión Logística, Análisis Discriminante y Árboles de Regresión, entre otras (Cuadras, 2007).

Posteriormente, debido al incremento de los datos y a la complejidad de los problemas a resolver surge la Minería de Datos, la cual hace uso de métodos de solución como Redes Neuronales (Clifford, 1991), Bayesianos (Devroye, 1996), basados en vecindad (1) y Máquinas de soporte vectorial (Vapnik, 1995), entre otros.

Por otra parte, existen también técnicas de Inteligencia Artificial que tienen la capacidad de solucionar tareas de predicción y recomendaciones haciendo uso del conocimiento brindado por expertos donde se utilizan esquemas basados en reglas de producción o en ontologías.

Tomando en consideración las características de las técnicas mencionadas anteriormente, el presente trabajo propone un modelo híbrido predictivo como una solución, debido a que además de usar técnicas predictivas y descriptivas de Minería de Datos se incorporan técnicas de Inteligencia Artificial sobre el conocimiento extraído de los datos. El modelo obtiene, en una primera fase, los patrones de comportamiento mediante la técnica de generación de Reglas de Asociación de Clases (RAC), esto con el objetivo de identificar las situaciones generales y también aquellas que son particulares pero relevantes en el conjunto de datos bajo estudio. En una segunda fase, se modelan los patrones identificados en un sistema basado en conocimiento bajo un esquema ontológico, lo que permitirá de manera natural, realizar recomendaciones pertinentes al usuario. En la tercera fase, se clasifica una instancia a partir de sus características y la clase brindada por el clasificador ayudará al modelo a emitir una serie de recomendaciones en caso de que se requieran.

La denominación de híbrido se establece en el uso de diversas técnicas, de Minería de Datos (descriptivas y predictivas) y de Inteligencia Artificial (recomendación).

Antecedentes

En los últimos años, se han desarrollado diversos trabajos sobre modelos predictivos, los cuales se caracterizan por utilizar de manera independiente ya sea técnicas de Minería de Datos o de Inteligencia Artificial y enfocarse a una solución específica. Respecto a la primera técnica, los trabajos predicen la categoría de clase mediante un método de clasificación. En estos trabajos se utilizan algoritmos como Árboles de Decisión (Timarán, 2012), Redes Neuronales (Nachev, 2016), Máquinas de Soporte Vectorial (Anbuselvan & Balamuralithara, 2014), Naive Bayes (Shweta & Sunita, 2016), entre otros.

Por su parte, los trabajos que hacen uso de técnicas de Inteligencia Artificial obtienen el conocimiento a partir del experto humano y lo modelan mediante esquemas de representación del conocimiento ontológicos (Espín, Hurtado, & Noguera, 2016), (Gopalachari & Sammulal, 2014) o reglas de producción (Li & Roy, 2015).

Problemática

En los modelos predictivos antes mencionados, se han detectado las siguientes limitantes:

- Los modelos se desarrollan para casos específicos, no se demuestra su aplicabilidad a otras áreas del conocimiento.
- No se dan a conocer posibles cambios de comportamiento en la predicción.
- Utilizan uno o dos clasificadores, pudiendo no ser los que brindan el mejor desempeño.

Estas limitantes conllevan a que los usuarios que utilizan este tipo de modelos cuenten con pocos elementos para la toma de decisiones.

Solución propuesta

El presente trabajo propone una solución a la problemática mencionada aplicando, por una parte, técnicas predictivas como descriptivas de Minería de Datos y por otra, técnicas de Inteligencia Artificial, lo cual representa un modelo híbrido que da una alternativa novedosa en el proceso de predicción y recomendación.

Tomando en cuenta los inconvenientes mencionados en la problemática, se propone un modelo que dé respuesta a los mismos, considerando:

- Un modelo genérico aplicable en cualquier área del conocimiento.
- La adquisición de la mayor cantidad de conocimiento posible a partir de los datos fuente, utilizando tanto técnicas predictivas como descriptivas de Minería de Datos.
- La obtención la categoría de la clase en la predicción y los patrones frecuentes encontrados.

- El modelo sugiere el método de clasificación que brinde mejor desempeño de acuerdo a las características del conjunto de datos.

Hipótesis

El modelo de predicción híbrido basado en técnicas de Minería de Datos e Inteligencia Artificial permite enriquecer los resultados brindados al usuario, proporcionando los patrones más frecuentes de comportamiento o recomendaciones, lo cual facilitará la toma de decisiones.

Objetivo general

Definir un modelo híbrido predictivo basado en técnicas de Minería de Datos e Inteligencia Artificial para facilitar la toma de decisiones por parte del usuario.

Objetivos específicos

- Evaluar diversos algoritmos de reglas de asociación para aplicar en el modelo aquél que mejor se desempeña en cuanto a redundancia.
- Modelar reglas de asociación en un esquema ontológico para efectuar razonamiento y poder generar recomendaciones.
- Evaluar algoritmos de clasificación con diversos enfoques para recomendar aquellos que brindan mejores resultados de acuerdo a las características del conjunto de datos.
- Validar el modelo de predicción mediante casos de estudio para demostrar su buen desempeño.

Justificación

Con la propuesta de solución, el usuario contará con un modelo que le proporcione los resultados de forma sencilla y fácil de comprender, sin requerir conocimientos avanzados sobre técnicas de Minería de Datos o Inteligencia Artificial.

Con el modelo propuesto, en los casos donde se brinden recomendaciones, el usuario contará con conocimiento adicional y personalizado para la toma de decisiones. Esto

último debido a que, para brindar las recomendaciones, del total de patrones frecuentes que se detectan en los datos, sólo se utilizan aquellos que coinciden o se relacionan con las características de la nueva instancia a clasificar.

Aportaciones

Las aportaciones más relevantes del presente trabajo son:

- Modelo que combina técnicas de Minería de Datos con técnicas de Inteligencia Artificial.
- Procedimiento para modelar reglas de asociación de clase en una base de conocimientos integrada por conceptos, propiedades de datos y un conjunto de axiomas.
- Conjunto de criterios para seleccionar un clasificador dependiendo de la naturaleza de los datos.

Estructura del documento

El documento está conformado por siete capítulos, conclusiones y las referencias bibliográficas. A continuación, se muestra un pequeño resumen de lo que contendrá cada uno de los capítulos.

En el capítulo uno se expone el marco teórico, el cual contiene las bases teóricas y algoritmos de diferentes tipos de enfoques.

En el capítulo dos en una primera sección se exponen los antecedentes, mostrando un estudio de diversos trabajos de modelos predictivos realizados hasta el momento con uso de técnicas de Minería de Datos e Inteligencia Artificial, explicando su finalidad, métodos utilizados y los resultados obtenidos en cada caso. En las siguientes secciones, se exponen trabajos relacionados con análisis comparativos del desempeño de algoritmos de reglas de asociación, con la representación de reglas de asociación en esquema ontológico y finalmente, con el comportamiento de algunos clasificadores frente a conjuntos de datos de diferentes características.

En el capítulo tres se presenta el modelo híbrido propuesto, los distintos módulos que lo conforman, la forma en que se relacionan, el flujo de la información en el proceso, los tipos de técnicas de Minería de Datos combinadas y las ventajas que presenta el utilizar técnicas de Inteligencia Artificial.

En el capítulo cuatro se expone como generar las reglas de asociación y el trabajo experimental desarrollado para evaluar el comportamiento de varios algoritmos con diferentes conjuntos de datos, esto con la finalidad de seleccionar el que brinde el mejor desempeño en el tratamiento del problema de redundancia.

En el capítulo cinco se expone el proceso que se debe efectuar para modelar un conjunto de reglas de asociación de clase obtenido anteriormente, en un esquema ontológico.

En el capítulo seis se expone cómo seleccionar el método de clasificación a utilizar de acuerdo a la naturaleza de los datos, se describen las características de las bases de datos a considerar, el trabajo experimental de diversos algoritmos de clasificación con varios conjuntos de datos, los criterios obtenidos y pruebas realizadas.

En el capítulo siete se explica el proceso efectuado para validar el modelo mediante seis casos de estudio, uno con datos referentes a la predicción de deserción de alumnos de nivel licenciatura y otros cinco utilizando bases de datos obtenidas del repositorio UCI.

Finalmente, se exponen las conclusiones, el trabajo futuro que se recomienda desarrollar y las referencias sobre las que se fundamenta el presente trabajo.

Capítulo 1
Marco teórico

El objetivo de este capítulo es exponer una reseña sobre lo que son los modelos predictivos, así como los fundamentos teóricos tanto de técnicas de Minería de Datos como de Inteligencia Artificial, los cuales se utilizarán en el presente documento.

1.1 Modelos predictivos

Los modelos predictivos permiten obtener tendencias futuras, en ellos se hace inferencia sobre los datos conocidos con el objetivo de prever el comportamiento de otros nuevos. Estos requieren de un proceso de aprendizaje supervisado y pretenden estimar valores futuros o desconocidos de una variable. Cuando el valor por predecir es de tipo categórico éste se denomina *Modelo de Clasificación* y cuando es de tipo numérico se llama *Modelo de Regresión* (Hand, 2001). A continuación, se hace una descripción de cómo fueron surgiendo.

El primer modelo predictivo fue desarrollado desde los años treinta por Thompson y Bell (Thompson & Bell, 1934), debido a que se basa en numerosos cálculos matemáticos su mayor aplicación la tuvo hasta que se introdujo el uso de las computadoras. Años más tarde, Beverton y Holt propusieron un modelo similar considerado una mejora del anterior denominado *Rendimiento por Recluta* (Beverton & Holt, 1957).

En los años 70, se crearon otros modelos de enfoque estadístico, los cuales se basaban en métodos de regresión, siendo los más usados el de regresión lineal y logística. El modelo de regresión lineal tuvo gran aceptación debido a su capacidad de estudiar la relación entre distintas variables, siendo éste de dos tipos: *Regresión Simple* y *Regresión Múltiple*. Por otra parte, el modelo de regresión logística se creó para casos en los que se requiriera estimar la probabilidad de un evento dicotómico en función de un conjunto de variables predictoras que podían ser discretas o categóricas (Light & Margolin, 1971).

Posteriormente, en los años 80 surgieron modelos desarrollados con técnicas de árboles de decisión y técnicas basadas en vecindad (Coomans & Massart, 1982).

En la década de los 90, surge una disciplina que permite extraer conocimiento a partir de grandes bases de datos, denominada *Minería de Datos* (Frawley, Shapiro, & Matheus, 1992). Con esta disciplina se han desarrollaron modelos predictivos con enfoques de solución como el Probabilístico (Kononenko, 1990), (Genkin, 2004), *Redes Neuronales* (Lu & Setiono, 1997), *Máquinas de Soporte Vectorial* (Vapnik, 1995) y Basados en Reglas (Cohen, 1995), (Holte, 1993), (Eibé & Witten, 1998), entre otros.

En las siguientes secciones, se abordan los conceptos fundamentales de las técnicas que se utilizan en el modelo híbrido predictivo propuesto en este trabajo.

1.2 Minería de Datos

La Minería de Datos se considera como el proceso de descubrimiento de nuevas y significativas relaciones, patrones y tendencias al examinar grandes cantidades de datos (Pérez & Santín, 2007). Por otro lado, Fayyad (2009) la define como un proceso no trivial de identificación válida, novedosa, potencialmente útil y entendible de patrones comprensibles que se encuentran ocultos en los datos.

Las tareas más comunes empleadas en la Minería de Datos son: Clasificación, Agrupamiento, Regresión y Generación de asociaciones. Estas tareas se pueden clasificar en predictivas y descriptivas, donde las primeras tienen el objetivo de estimar valores futuros o desconocidos de variables de interés, mientras que las segundas identifican patrones que explican o resumen los datos de entrada. A continuación, se de una breve explicación de cada técnica.

- *Agrupamiento*: Tarea descriptiva que consiste en descubrir cómo es que ciertas instancias se agrupan en determinadas clases de tal manera que las mismas comparten ciertas características. Su objetivo es dividir un conjunto de instancias en varios grupos similares entre sí. Estos métodos se dividen en tres grupos básicos: *jerárquicos*, *particionales* y *basados en densidad* (Cabena, 1998).

- *Clasificación*: Tarea predictiva que se encarga de analizar el comportamiento y las variables de grupos predeterminados pues su objetivo es clasificar nuevos datos examinando datos existentes (Cuello, 2006).
- *Regresión*: Tarea predictiva que se diferencia de la técnica de clasificación en que predice valores continuos, generalmente utiliza métodos estadísticos como la regresión lineal y logística (Hernández, Ramírez, & Ferri, 2013).
- *Reglas de Asociación*: Tarea de tipo descriptivo que consiste en encontrar patrones de asociación interesantes que existen entre los atributos de una base de datos. Para obtener los patrones que se presentan con mayor frecuencia se utilizan dos parámetros denominados *Soporte* y *Confianza* (Pérez & Santín, 2007).

El modelo predictivo propuesto hace uso de las técnicas de clasificación y reglas de asociación, las cuales se detallan en las siguientes secciones.

1.2.1 Clasificación

El objetivo de esta tarea es obtener un valor particular de un atributo a partir de los valores de otros atributos. El atributo a obtener es comúnmente llamado *Clase* o *Variable Dependiente*, mientras que los atributos usados para hacer la predicción se llaman *Variables Independientes* (Cuello, 2006).

Dentro de las tareas predictivas, la clasificación es una de las más utilizadas por su capacidad de identificar para una nueva instancia, la clase a la cual está asociada, utilizando para ello, el conocimiento que relaciona las características de instancias existentes con sus respectivas clases.

Existe una gran diversidad de algoritmos que hacen uso de esta técnica, en las siguientes secciones se explican algunos de los más referenciados en la literatura.

1.2.1.1 Método C4.5

El método C4.5 está basado en árboles de decisión, el cual utiliza la ganancia de información por medio del cálculo de la entropía para medir qué tanto un atributo separa el conjunto de instancias de acuerdo a sus clases (Quinlan, 1996).

En Algoritmo 1, se muestra el procedimiento que utiliza C4.5, en el cual se inicia la construcción del árbol con un nodo raíz vacío. Después de haberse creado la raíz, se comprueba si las diferentes instancias tienen el mismo valor para el atributo clase, de ser así se obtiene sólo un nodo, posteriormente para todos los atributos no clase se verifica mediante el cálculo de entropías cuál es el que proporciona mayor ganancia para ubicarlo en el nodo del nivel más alto del árbol, asignando al nodo el nombre del atributo ganador y a los arcos los valores de dicho atributo. Para el resto de los atributos no clase, esto se realiza de forma iterativa hasta llegar a los nodos hoja (clases).

Algoritmo 1*:* Método C 4.5

```
A: Conjunto de atributos
NA: Número de atributos
NR: Nodo raíz
NV: Número de valores de un atributo
Am: Mejor atributo
    //Crear nodo raíz vacío
    for i=1 to NA do
        Calcular ganancia de Ai
        ∑_1^nv A_i log_2 A_i
    end for
            Am=mejor atributo (A)
            NR=Am
   for i=1 to NV do
        Crear nuevo nodo hijo
   end for
```

Cuando se quiere clasificar una nueva instancia, se utilizan sus diferentes atributos, incluyendo sus valores para recorrer el árbol creado iniciando por el nodo raíz. El recorrido de los nodos y arcos se efectúan de forma descendente hasta encontrar la hoja buscada (clase).

1.2.1.2 KNN

El método KNN (por sus siglas en inglés, K Nearest Neighbors) se basa en efectuar aprendizaje por analogía, consiste en realizar una serie de comparaciones para que a una nueva instancia que contiene *n* atributos se le asigne la clase mayoritaria de sus *k* vecinos más cercanos (Coomans & Massart, 1982). Las comparaciones entre las instancias se realizan mediante algún criterio de vecindad definida en términos de algún tipo de métrica, cuando todas las variables son numéricas se aplican métricas

como la distancia Euclidiana, de Manhattan, de Chebyshev, entre otras, por otra parte, cuando se tienen tanto variables numéricas como categóricas, se puede aplicar la métrica de Gower (Deza & Deza, 2009).

Como se muestra en el Algoritmo 2, el proceso inicia especificando los datos de la nueva instancia. Posteriormente, es importante indicar el número de vecinos a evaluar, así como seleccionar la métrica con la que se calculará la distancia entre las instancias. El siguiente paso consiste en calcular la distancia que existe entre la nueva instancia y las demás. Posteriormente, se ordenan los resultados de manera ascendente y de sus vecinos más cercanos se verifica cuál es la clase más frecuente para asignarla a la nueva instancia.

Algoritmo 2*:* Método KNN

```
D: Datos de entrada
Y: Nueva instancia a clasificar
NI: Número de instancias ya clasificadas
M: Nueva matriz resultante
k: Número de vecinos más cercanos
        // Calcular la distancia o similitud de Y con cada instancia de D
        for i=1 to NI
            Calcular dist (Y, Di)
        end for
        // Se ordenan todos de menor a mayor distancia o mayor a menor similitud
        for i=1 to NI
            Ordenar(M)
        end for
        Seleccionar los k casos M1^k ya clasificados más cercanos a Y
        Asignar a Y la clase más frecuente en los k vecinos
FIN
```

1.2.1.3 Perceptrón multicapa

El algoritmo Perceptrón Multicapa (MLP por sus siglas en inglés, Multi Layer Perceptron) se basa en un método de propagación hacia atrás para clasificar nuevas instancias con el uso de redes neuronales. La red neuronal de retro propagación es esencialmente una interconexión de elementos simples de procesamiento que trabajan juntos para producir una salida. El entrenamiento de la red consiste en calcular de manera iterativa un conjunto de pesos para la predicción de la clase de la instancia analizada (Lu & Setiono, 1997).

Una red neuronal de tipo MLP está formada de una capa de entrada, una o más capas ocultas y una capa de salida. Cada capa de la red neuronal está compuesta por un conjunto de unidades (neuronas).

Como se muestra en el Algoritmo 3, se tiene que indicar un patrón de entrada en su primera capa. Posteriormente, para cada entrada se maneja un peso y se activa su conexión a la siguiente capa con una función de activación, repitiéndose este proceso con las capas ocultas. Las entradas a la red se conforman por los valores de los atributos de la instancia en tratamiento. Las entradas son alimentadas simultáneamente en las neuronas que forman la capa de entrada y luego se ponderan y alimentan las capas ocultas.

Algoritmo 3*:* Método Perceptrón Multicapa

```
NE: Número de entradas
NV: Número de variables
NI: Número de iteraciones de aprendizaje
Alfa: valor de aprendizaje
W: Pesos de las variables de entrada
//Iniciar los pesos con valores aleatorios
Alfa= valor_inicial
for  i=1 to NE do
    Random (Wi)
end for
for i=1 to NI do
     for j=1 to NV
          Propagar la activación hacia adelante con los valores de Wj
    end for
          Verificar que la activación alcance las neuronas de la capa de salida
          Realizar reajuste de pesos requeridos
          Se repiten los pasos anteriores hasta alcanzar el criterio de parada
end for
```

La capa de salida corresponde al resultado esperado (clase de la instancia de entrada), por lo que la red neuronal puede estar aprendiendo o retroalimentando valores en sus capas intermedias hasta obtener dicho resultado.

Para obtener la clase de una nueva instancia, se utilizan los valores de cada atributo, la cantidad de ellos va a ser igual al número de entradas de la primera capa, se le asigna un valor aleatorio al peso de cada entrada y se realizan los cálculos necesarios hasta obtener los pesos adecuados del patrón de entrada correspondiente a alguna clase.

1.2.1.4 Naive Bayes

El método Naive Bayes permite usar el conocimiento apriori para predecir una suposición mediante el cálculo de probabilidades. El uso del teorema de Bayes en la tarea de clasificación se debe a que permite calcular las probabilidades a posteriori de cualquier hipótesis consistente con el conjunto de datos de entrenamiento para así escoger la hipótesis más probable (Miquelez, Bengoetxea, & Larranaga, 2004).

Como se muestra en el Algoritmo 4, el método consiste en calcular tanto la probabilidad de cada clase como la de los diferentes valores de cada variable independiente. Debido a que este clasificador asume que las características son condicionalmente independientes, evita comparaciones de ganancia de información entre combinaciones de variables o atributos.

Algoritmo 4*:* Método Naive Bayes

NC: Número de clases
C: Conjunto de clases
V: Valor del atributo
A: Atributo
NA: Número de atributos

```
//Calcular la probabilidad de cada clase
 for i = 1 to NC do
      Obtener Probabilidad(Ci)
 end for
 for i = 1 to NA do
        for j=1 to NV
           Para cada posible valor del resultado vj
           Obtener la probabilidad P'(Ai | Vj)
        end for
  end for
    Clasificar nueva instancia descrita por un conjunto de atributos (ai) en conjunto de clases (C)
    Clasificar de acuerdo con el valor más probable dados los valores de sus atributos.
```

1.2.2 Reglas de asociación

La generación de reglas de asociación es una de las técnicas de Minería de Datos para extraer conocimiento oculto y que tiene como objetivo identificar relaciones no explícitas entre los atributos de la base de datos (Agrawal, Mining Associations beetwen sets of items in massive databases, 1993). Una regla de asociación es una implicación de la forma X→Y, donde a X se le denomina *Antecedente* y a Y *Sucedente* o *Consecuente*, en la cual ambos componentes pueden estar conformados por más de un atributo. Dentro de las reglas de asociación existen dos medidas básicas de control denominadas *Soporte* y *Confianza*, las cuales definen el grado de interés de la regla.

Considérese T = $\{t_1, t_2, \ldots, t_n\}$ un conjunto de transacciones, cada una de las cuales contiene un conjunto de k ítems, referidos a los valores de los atributos que la describen. El soporte de una regla de asociación X→Y es la fracción de transacciones de T que contienen X ∪ Y. Por otra parte, la confianza de X→Y es la fracción de transacciones que contienen a X ∪Y de entre el total de aquellas que contienen a X.

Las reglas de asociación deben satisfacer las especificaciones del usuario en cuanto a valores mínimos de soporte y confianza. Para conseguir esto, en el proceso de generación de reglas, el soporte se utiliza para determinar qué conjuntos de pares atributo-valor se presentan con mayor frecuencia en la base de datos y posteriormente, el parámetro de confianza permite determinar qué reglas son válidas a partir de los conjuntos de elementos obtenidos anteriormente. En reglas de asociación, se le denomina *Item* a un par atributo valor e *Itemset* a un conjunto de ítems.

Para generar las reglas de asociación existen diversos algoritmos, en las siguientes secciones se exponen los más referenciados en la literatura.

1.2.2.1 Método Apriori

Este método se llama Apriori porque se basa en el conocimiento previo de un conjunto de elementos frecuentes, utiliza una estrategia de búsqueda primero en amplitud que consiste en realizar un recorrido de forma horizontal (Agrawal, 1994). El algoritmo ocupa básicamente dos pasos para obtener las reglas, primero realiza la obtención de

itemsets frecuentes y después, la obtención de las reglas que cumplen con los parámetros especificados.

Para obtener los itemsets frecuentes, como se muestra en el ejemplo de la Figura 1.1, el algoritmo inicia obteniendo los conjuntos de ítems con un solo elemento (filtrando los que superan el soporte mínimo), ejemplo de ello es el itemset cuatro, el cual no superó el soporte mínimo=0.5. Posteriormente, se repite el mismo proceso con los de dos elementos y se sigue iterando con los restantes ítems hasta que no se encuentren más conjuntos, en este caso como con tres elementos sólo se obtuvo un conjunto, ahí termina el proceso.

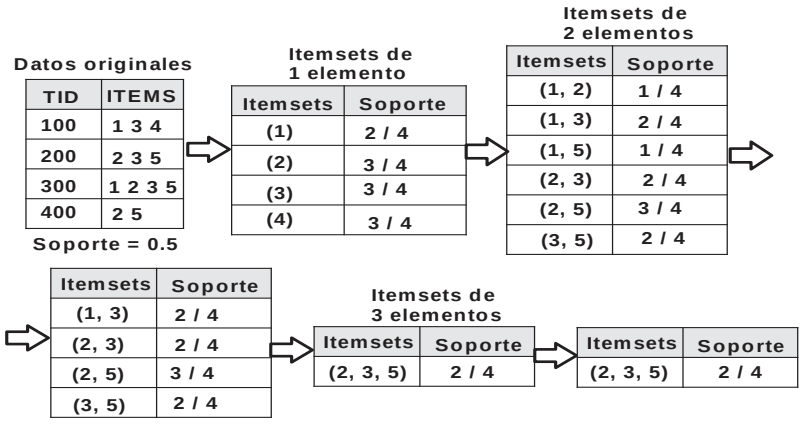


Figura 1.1 Etapa de obtención de itemsets frecuentes del método Apriori

Una vez obtenidos los itemsets frecuentes, el algoritmo pasa a la etapa de generación de reglas utilizando dichos itemsets y verificando cuáles superan la confianza mínima especificada por el usuario. En la Figura 1.2, se muestra el total de las reglas obtenidas a partir de los itemsets frecuentes, de las cuales se filtran sólo las que superan el valor de confianza, como en este caso es 0,9, únicamente se filtrarán las reglas 1, 4, 9, 11 y 14.

Figura 1.2 Etapa de obtención de reglas del método Apriori

En el Algoritmo 6, se muestran los principales pasos del método.

Algoritmo 6*:* Método Apriori

```
L[ ]: Conjunto de itemsets frecuentes
C[ ]: Conjunto de itemsets candidatos
LR[ ]: Conjunto de reglas generadas
K: Número de ítems
TotalItems: Número total de items en la BD
NumMaxItems: Número máximo de items de un itemset
//Obtención de itemsets frecuentes
k=1
for i=1 to i=TotalItems do
    if Ci[k].Soporte ≥ minsop then
        Li[k] = Ci[k]
    end if
end for
k++
while k ≤ NumMaxItems OR L[k] ≠ Ø do
    Ci[k] = Li[k-1]* Li[k-1]
    for i=1 to i=NumTransacciones do
        Ci[k].Soporte++
        if Ci[k].Soporte ≥ minsop then
            Li[k] = Ci[k]
        end if
    end for
    k++
end while
// Obtención de reglas de asociación
for k=2 to NumMaxItems do
    if Li [k].confianza ≥ minconf then
        LRi[k] = Li[k]
    end if
 end for
```

1.2.2.2 Método Eclat

El método Eclat, al igual que el Apriori, se encarga de encontrar inicialmente los conjuntos de elementos frecuentes, pero con la diferencia de que aquí se hace uso del número de transacciones (por sus siglas en inglés TID, Transactions IDentifier), por lo que se considera que el método trabaja de forma vertical (Zaki, 2000).

Primero, se recorren los datos originales construyendo una lista con los TID. Posteriormente, se obtienen los itemsets frecuentes de tamaño uno y luego los conjuntos de elementos frecuentes de tamaño dos en adelante. Los conjuntos Lk, donde

k≥2, surgen a partir de la intersección del conjunto de elementos previo (L_{K-1}), el cual es generado de manera similar al método Apriori, debido a que se almacenan como itemsets frecuentes únicamente aquellos que superan el soporte mínimo especificado.

El algoritmo tiene la ventaja de que calcula el soporte con mayor rapidez que Apriori ya que contabiliza la frecuencia de los TID, para lo cual requiere utilizar listas intermedias. Al igual que los algoritmos anteriores, para el proceso de generación de reglas, se verifican aquellos conjuntos de elementos frecuentes que superen el valor de la confianza mínima requerida.

En la Figura 1.3 se muestra un ejemplo de cómo se obtienen los elementos frecuentes a partir de un conjunto de datos inicial de cinco transacciones, filtrándose conjuntos de uno, dos y tres elementos que superan un soporte mínimo=2. Por ejemplo, cuando se filtran los conjuntos que contienen dos elementos de un total de diez, sólo cinco superan el soporte.

Al igual que el método anterior, una vez obtenidos los itemsets frecuentes, estos son utilizados para obtener todas las reglas que cumplen con la confianza indicada por el usuario.

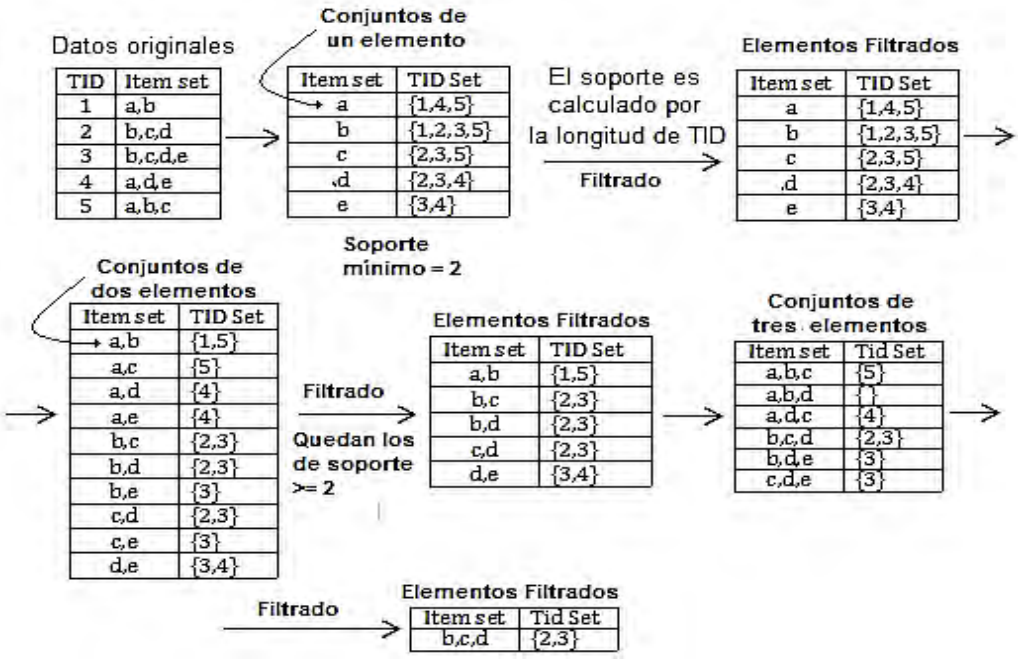


Figura 1.3 Obtención de itemsets frecuentes con Eclat

En el Algoritmo 7 se muestran los principales pasos del método.

Algoritmo 7: Método Eclat

```
L[ ]: Conjunto de itemsets
Matriz [ ][ ]: Matriz de los TID
TotalItems: Número total de items en la BD
NumMaxItems: Número máximo de items de un itemset
   for i=1 to i=TotalItems do        // Representar la matriz de los TID
       for j=1 to j=NumTransacciones do
             Matriz[i][j]=TID
        end for
   end_for
   for i=1 to i=NumItems do        // Obtener el soporte de los itemsets frecuentes
     for j=1 to j=Num_TID do
       for k=1 to j=Num_TID do
         if (Matriz[i][j].TID == Matriz[i+1][k].TID) then
               soporte ++
          end if
        end for
     end for
         if soporte ≥ minsop then
              L_i[k]= Matriz[i] [j].TID+ Matriz[i+1][j].TID
          end if
   end for
   // Obtención de reglas de asociación
   for k=2 to NumMaxItems do
        if  L_i[k].Confianza ≥ minconf then
          LR_i[k] = L_i[k]
        end if
end for
```

1.2.2.3 Método FP-Growth

El método FP-Growth se describe como un esquema de eliminación recursiva y se caracteriza por el uso de estructuras de árbol (Jiawei, 2004). El algoritmo requiere realizar únicamente dos recorridos a la base de datos. El primero, para obtener una lista de items frecuentes almacenándolos en una lista y el segundo, para representar en un árbol, el contenido de la lista obtenida, colocando cada ítem en orden descendente de arriba hacia abajo.

Las transacciones de conjuntos de datos suelen compartir itemsets frecuentes y, por lo tanto, el tamaño del FP-Tree correspondiente suele ser menor que el del conjunto de datos original. Como se muestra en el Algoritmo 8, como primer paso, en una primera etapa, se obtiene la frecuencia de los itemsets que cumplen con el soporte mínimo

requerido y en una segunda etapa se construye la estructura de árbol FP (Frequent Pattern) insertando las instancias según el orden decreciente de su frecuencia en el conjunto de datos, para que posteriormente, el árbol pueda ser procesado rápidamente.

Algoritmo 8: Método FP-Growth

```
L[ ]: Conjunto de items frecuentes
LR[ ]: Conjunto de reglas generadas
NumItems: Número de items que contiene un itemset
NumTransacciones: Total de transacciones en la BD
   TreeB:  Estructura de árbol, cada nodo contiene
           nombre del ítem y soporte
   for i=1 to i= NumItems do  //Verifica itemsets que superan soporte mínimo
       if Item. Soporte ≥ minsop then
           L[i] = item
       end if
    end for
   Tree TreeB= new TreeB( )        // Representación del árbol
   for i=1 to i= NumTransacciones do
     for j=1 to j= NumItems do
       if TreeB. Item  ≠ Ø then
           soporte++
       else
          TreeB.Addnodo (Item, Soporte)
       end if
     end for
    end for
   for k=2 to NumMaxItems do    // Obtención de reglas de asociación
        if  Itemset.Confianza ≥ minconf then
          LR_i[k] = TreeB. Itemset
       end if
end for
```

En la Figura 1.4, se muestra un ejemplo de este método, en el inciso *a* se describen los datos originales de tres transacciones y del inciso *b* al *d* se muestra cómo se genera el árbol, en el inciso *d* se puede corroborar que los ítems quedan ordenados.

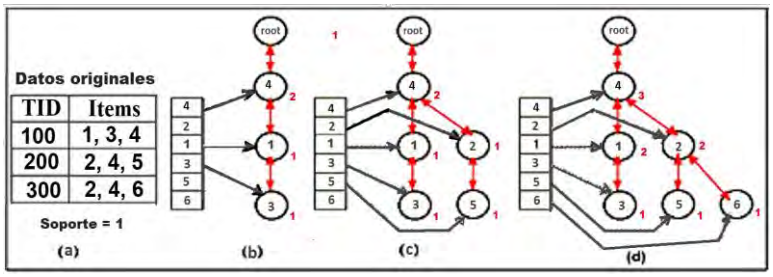


Figura 1.4 Ejemplo de Método FP-Growth

Una vez que ha finalizado el proceso recursivo para la creación del árbol, se han representado todos los conjuntos de elementos que superaron el soporte mínimo. El siguiente paso consiste en generar todas las reglas que superen la confianza especificada por el usuario a partir de los datos representados en el árbol. Por ejemplo, si la confianza es igual a 0.60, las únicas reglas obtenidas serian 4⟹2 y 4⟹ 1.

1.2.2.4 Método Close

El método Close fue propuesto por Nicolás Pasquier (2004), la idea de implementar este método surgió debido a la falta alguna estrategia en los métodos anteriores para eliminar aquellas reglas sobrantes que no aportan algún conocimiento agregado (redundancia) y que dificultan la interpretación de los resultados.

En la etapa inicial, trabaja de manera similar al método Apriori, primero obtiene un conjunto de itemsets generadores y posteriormente a partir de ellos se verifica cuáles son itemsets frecuentes cerrados (FCI por sus siglas en inglés, Frecuent Close Itemsets). Un itemset frecuente se considera cerrado si contiene uno o más itemset frecuentes de menor número de elementos y tienen el mismo soporte.

En la Figura 1.1(a), se muestra un conjunto de datos origen, en el inciso b se muestran los itemsets frecuentes que se obtienen con Apriori y en el inciso c, los que se obtienen con Close. Como se puede observar, la cantidad de itemsets frecuentes obtenidos con el método Close es menor y por consecuencia, como se muestra en la Figura 1.2 (b), el número de reglas generadas también disminuye, por lo cual se puede constatar que el método Close ayuda a eliminar reglas redundantes con el uso de los FCI.

Figura 1.5 Datos origen e itemsets frecuentes

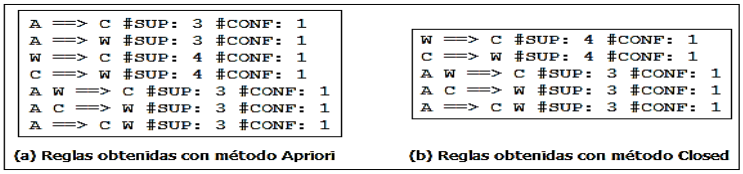


Figura 1.6 Reglas de asociación generadas

En el Algoritmo 9, se muestran los principales pasos del método.

Algoritmo 9: Método Close

```
L[ ]: Conjunto de itemsets frecuentes
C[ ]: Conjunto de itemsets candidatos
FCI[ ]: Conjunto de itemsets frecuentes cerrados
    for i=1 to i=TotalItems do      //Obtención de itemsets frecuentes
        if Ci[k].Soporte ≥ minsop then
            Li[k] = Ci[k]
        end if
    end for
    while k ≤ NumMaxItems OR L[k] ≠ Ø do
        Ci[k] = Li[k-1]*Li[k-1]
        for i=1 to i=NumTransacciones do
            Ci[k].soporte++
            if Ci[k].Soporte ≥ minsop then Li[k] = Ci[k] end if
            if ((Li[k].Soporte >= Li[k-1].Soporte) &&        // Verifica si el itemset es frecuente cerrado
                (Li[k-1] ⊆ Li[k])) then FCIi[k] = Li[k]
            end if
        end for
        k++
    end while
    for k=2 to NumMaxItems do // Obtención de reglas de asociación
        if FCIi[k].Confianza ≥ minconf then
            LRi[k] = FCIi[k]
        end if
    end for
```

1.2.2.5 Charm

El método Charm, al igual que Close, permite obtener FCI, pero con la diferencia de que en el método Charm la obtención de los itemsets se realiza mediante una representación vertical del conjunto de datos original (Zaki & Hsiao, 2002). Para obtener todos los FCI, enumera los conjuntos cerrados utilizando un árbol de búsqueda de los identificadores de transacciones.

Como el método utiliza una técnica que permite trabajar con los identificadores de cada transacción, le permite reducir el consumo de memoria en los cálculos intermedios. Para trabajar con los FCI utiliza una estrategia de búsqueda de Hash para eliminar cualquier conjunto *no cerrado* que se encuentra durante el cálculo, lo cual se muestra en el Algoritmo 10.

Algoritmo 10: Método Charm

```
Tree: Estructura de árbol donde cada nodo contiene Itemset y TID
X: Itemset auxiliar
t(X): TID de Itemset
TotalItems : Número total de items en la BD
for i=1 to i= TotalItems do
          if Xi. Soporte ≥ minsop then
              Tree.Add(Xi, t(Xi))
           end if
  end for
 forall (Xi, t(Xi))
      X = Xi
       forall (Xj, t(Xj))
                X= Xi∪Xj && Y=t(Xi)∩t(Xj)
                    if t(Xi)=t(Xj) then //Propiedad 1
                       Tree.Delete(Xj)     Xi=X
                    else if t(Xi) ⊂ t(Xj) then // Propiedad 2
                       Xi= X
                    else if t(Xi) ⊃ t(Xj) then // Propiedad 3
                       Tree.Delete(Xj)  Tree.Add(X, Y)
                     else if t(Xi) = t(Xj) then //Propiedad 4
                       Tree.Add(X, Y)
                    end if
       end for
end for
```

1.2.2.6 Método Top-K Rules

El método Top-K Rules fue propuesto por Philippe Fournier, el cual a diferencia de los métodos anteriores no utiliza dos etapas (generación de itemsets frecuentes y obtención de reglas) sino que las reglas se generan en una sola. El proceso consiste en iniciar con una regla que contiene un único elemento tanto en el antecedente como en el consecuente y posteriormente, en cada iteración se van agregando nuevas reglas o a las existentes se le adicionan nuevos elementos al antecedente o consecuente agregándose a una lista, en el Cuadro 1.1 se muestra un ejemplo de expansión en el antecedente (Viger & Philippe, 2012).

Como se muestra en el Algoritmo 11, el método inicia obteniendo los identificadores de transacciones de todos los pares de elementos con un solo elemento en i, j siendo i el antecedente y j sucedente.

{ }
{ a } ⟹ { g }
{a, b} ⟹ { g }
{a, b, d } ⟹ { g }

Cuadro 1.1 Expansión de reglas en TopK-Rules

Algoritmo 11: Método Top-K Rules

```
LR[ ]: Conjunto de reglas generadas
R: Regla temporal
T: Total de transacciones de la BD
 forall  item(i,j) do
  if (TID(i).Soporte ≥ minsop) && (TID(j) .Soporte ≥ minsop) then
      soporte({i}→{j}) = TID(i)∩TID(j)/T
      soporte ({j}→{i}) = TID(i)∩TID(j)/T
      confianza({i}→{j}) = TID(i)∩TID(j)/TID(i)
      confianza({j}→{i}) = TID(i)∩TID(j)/TID(j)
     if (soporte({i}→{j}) ≥ minsop) then
          if (confianza({i}→{j}) ≥ minconf) then
             Ban_Expanción_{i}→{j} = true
              R = R∪{{i}→{j}, {j}→{i}}
           end if
           if confianza ({j}→{i}) ≥ minconf then
               Ban_ Expanción _{j}→{i} = true
               R = R∪{{i}→{j}, {j}→{i}}
           end if
     end if
  end for
 while ((R ∈ LR) && (soporte(R) ≥ minsop)) do
    if rule.Ban_Expanción = true then
       Expansión_Izquierda(R)
       Expansión _Derecha(R)
          if soporte(R) < minsop then
             Delete  R
           end if
 end while
```

Si el soporte del par de elementos no supera el mínimo soporte y confianza requerida, la regla no puede ser válida con i y j, de lo contrario es válida y almacenada en una lista temporal denominada *L*.

Si la lista L contiene más de k reglas y el soporte de las reglas es mayor que el mínimo soporte, las que son igual al mínimo soporte son removidas hasta tener sólo k reglas, finalmente el mínimo soporte es actualizado al más bajo de las reglas en L. Este proceso se continúa realizando de forma recursiva hasta no encontrar más elementos i, j con mayor soporte y confianza.

1.2.2.7 Algoritmo TNR

Se adicionan más restricciones respecto a las que contiene el método Top-K Rules, debido a que cada vez que se quiere adicionar una regla r_a a una lista de reglas denominada *LR*, además de comprobar si supera el valor de minsop, se debe verificar

si no es redundante con respecto a otra regla r_b ya almacenada en la lista. Para esto, el método efectúa alguno de las siguientes dos estrategias:

- Si una regla r_a generada tiene un soporte ≥ minsop y es redundante con respecto a otra regla r_b ya almacenada, r_a no se agrega a LR, de lo contrario, se adiciona.
- Si una regla r_a generada tiene un soporte ≥ minsop y se comprueba que una regla r_b almacenada es redundante con respecto a r_a, entonces se adiciona r_a y r_b es removida de LR.

El procedimiento general de TNR se muestra en el Algoritmo 12.

Algoritmo 12: Método TNR

```
LR[ ]: Conjunto de reglas generadas
R: Regla temporal
T: Total de transacciones de la BD
 forall  item(i,j) do
   if (TID(i)  ≥ minsop && TID(j)  ≥ minsop) then
      soporte({i}→{j}) = TID(i)∩TID(j)/T
      soporte ({j}→{i}) = TID(i)∩TID(j)/T
      confianza({i}→{j}) = TID(i)∩TID(j)/TID(i)
      confianza({j}→{i}) = TID(i)∩TID(j)/TID(j)
     if (soporte({i}→{j}) ≥ minsop) then
           if (confianza({i}→{j}) ≥ minconf) then
              Ban_Expanción_{i}→{j} = true
              R = R∪{{i}→{j}, {j}→{i}}
            end if
            if confianza ({j}→{i}) ≥ minconf then
               Ban_ Expanción _{j}→{i} = true
               R = R∪{{i}→{j}, {j}→{i}}
             end if
      end if
    end if
   While regla_ra ≠ Ø do
     if (soporte(ra) ≥ minsop) && (regla_ra ⊆ regla_rb)
            then LR.Delete(regla_ra)
            else  LR.Guardar(regla_ra)
      end if
   end while
   While regla_rb ≠ Ø do
       if (soporte(ra) ≥ minsop) && (rb ⊆ ra)  then
           LR.Delete(regla_rb)
           LR.Guardar(regla_ra)
        else
           LR.Delete(regla_ra)
         end if
   end while
end for
```

1.3 Inteligencia Artificial

En esta sección, se abordan conceptos sobre Inteligencia Artificial, esquemas de representación del conocimiento y ontologías, los cuales se utilizarán en el modelo propuesto.

La Inteligencia Artificial se define como la automatización de las actividades que se vinculan con procesos del pensamiento humano, actividades como la toma de decisiones y resolución de problemas (Russell & Norvig, 2004). También se define como el campo de estudio que busca explicar y emular el comportamiento inteligente en términos de procesos computacionales (Schalkoff, 1990).

Dentro de la Inteligencia Artificial, existen diversas áreas de estudio como: Sistemas basados en conocimiento, Procesamiento del lenguaje natural, Reconocimiento de patrones y Redes neuronales, entre otras.

Los Sistemas Basados en Conocimiento (SBC) tienen como finalidad resolver problemas de cierto dominio en el cual se tiene un conocimiento específico. Algunos autores los definen como:

- Aplicación informática en la que aparece representado, como estructura de información procesable, el conocimiento necesario para solucionar un determinado tipo de problemas de forma separada del procedimiento para resolverlas (Pajares, 2010).
- Sistema de Software capaz de soportar la representación explícita del conocimiento de un dominio específico y de explotarlo a través de los mecanismos apropiados de razonamiento para proporcionar un comportamiento de alto nivel en la resolución de problemas (Guida & Tasso, 1994).
- Programas informáticos que están diseñados para emular el trabajo de expertos en áreas específicas del conocimiento (Kendall, 2008).

Según Palma y Marín (2008), existen diversas propiedades o características de los SBC, entre las cuales se encuentran las siguientes:

- Permiten separar el conocimiento del dominio de los mecanismos de deducción.
- Permiten describir y justificar los pasos de razonamiento, para facilitarle al usuario a entender ciertos comportamientos.
- Realizan el procesamiento a través de mecanismos de inferencia.
- El programa que procesa una aplicación es el mismo para cualquier dominio de conocimiento.

Los SBC requieren de una estrategia para estructurar su contenido denominada *Esquema de representación del conocimiento* y de algún medio para hacer uso de dicho contenido conocido como *Lenguaje de representación del conocimiento*.

1.3.1 Esquemas de representación del conocimiento

Una representación del conocimiento es una combinación de estructuras de datos y procedimientos interpretativos que, utilizados de forma adecuada, conducirán a un comportamiento inteligente (Pajares, 2010).

Zeigier (1986) define los esquemas de representación del conocimiento como instrumentos para codificar la realidad en una computadora y Palma (2008), los define como modelos que permiten representar una situación del mundo real.

La elección del esquema o forma de representación del conocimiento es una tarea crucial que deberá facilitar tanto la interpretación del mismo por parte de los humanos como su tratamiento por parte de las máquinas.

El objetivo de los esquemas es formalizar y organizar dicho conocimiento, estos pueden estructurarse y representarse de diversas formas, dependiendo del problema que se intenta resolver. Algunas de las características que debe seguir la representación del conocimiento son: la generalidad de la representación para la resolución de problemas, la incorporación de nuevos conocimientos de manera sencilla, la naturalidad de la representación y la reutilización de sentencias.

Existen diferentes formas de clasificar los esquemas de representación del conocimiento, Pajares (2010) los clasifica en esquemas de representación lógica,

basados en reglas de producción, de redes semánticas y ontológicos. En las siguientes secciones, se explican los cuatro tipos.

1.3.1.1 Esquemas de representación lógica

Los esquemas de representación lógica fueron los primeros formalismos usados para representar estructuras de conocimiento, el propósito de usar la lógica como medio de representación de conocimiento es aportar una notación que tenga la facilidad de cubrir los conceptos deseados y al mismo tiempo, sea capaz de soportar procesos deductivos. Son modelos en los que el programador es responsable de especificar las relaciones lógicas básicas, ejemplos de este tipo de esquemas son la lógica proposicional y lógica de predicados.

1.3.1.2 Esquemas basados en reglas de producción

Esta forma de representación es la más popular usada para modelar el conocimiento dentro del paradigma declarativo, es más similar al razonamiento humano y se puede entender fácilmente. Su configuración permite construir sistemas en los que suele resultar sencillo incorporar nueva información o modificar la ya existente. La utilización de reglas permite o facilita la aplicación de distintos tipos de razonamiento para inferir nuevo conocimiento. Una regla de producción se enuncia de la siguiente forma: Si *antecedente* entonces *consecuente,* donde el antecedente (parte izquierda de la regla) es un conjunto de condiciones que se deben satisfacer en el dominio de la aplicación para evaluar la regla y el consecuente (parte derecha de la regla) es un conjunto de conclusiones o acciones que se derivan del antecedente.

1.3.1.3 Esquemas de redes semánticas

Los esquemas de representación aportados por Quillian y Shapiro (1968), son un método útil para la representación gráfica donde los esquemas pueden ser fácilmente traducibles a forma simbólica. Contienen dos elementos básicos, en primer lugar, están las estructuras de datos en *nodos* que representan conceptos unidas por *arcos* que representan las relaciones entre los conceptos y en segundo lugar, está un conjunto de procedimientos de inferencia que operan sobre dichas estructuras de datos. Ejemplos de este tipo de modelos son: Redes IS-A y grafos conceptuales (Brachman, 1983).

1.3.1.4 Esquemas ontológicos

Los esquemas ontológicos permiten representar el conocimiento de forma tal que los conceptos, las relaciones y las restricciones conceptuales de un determinado dominio son representados mediante formalismos. En el esquema ontológico, los conceptos se describen explícitamente para entender su significado mediante estándares establecidos, con ello un usuario que quiera reutilizar una ontología desarrollada por otro, puede obtener información a partir de algunos de sus componentes como clases, relaciones o axiomas (Nirenburg, 2004).

El amplio desarrollo de la investigación en el campo de las ontologías, ha permitido un aumento notable de su uso, ejemplos de ello son: la denominada *Web semántica* y la Ingeniería del Conocimiento.

Una ontología es una especificación explícita de una conceptualización, la cual es una abstracción o una vista simplificada del mundo que queremos representar, también se puede ver como un modelo de datos de conocimiento que especifica una forma de ver un grupo de entes y contiene definiciones que proveen un vocabulario para referirse a un dominio (Uschold, 1996).

Las ontologías no son formalismos cerrados y están sujetos a procesos evolutivos, algunas de sus ventajas son:

- Permiten reutilizar el conocimiento
- Facilitan la comunicación
- Permiten aplicar razonamiento
- Acceso más rápido a la información del conocimiento.

Las ontologías proporcionan un vocabulario común de un dominio y definen diferentes niveles de formalismo, el significado de los términos y las relaciones entre ellos (Gruber, 1993). El conocimiento en ontologías se formaliza mediante el uso de los siguientes componentes:

- *Clases o conceptos*: Estos elementos se conforman de ideas básicas que se intentan formalizar, contienen individuos, pueden ser organizadas dentro de una jerarquía de clases y subclases, lo cual se conoce como *Taxonomía*.
- *Relaciones*: Elemento ontológico que representan la interacción y enlace entre los conceptos de un dominio, suelen formar la taxonomía de las clases, por ejemplo: subclase-de, parte-de, conectado-a, entre otros. Por lo general, contienen relaciones binarias, donde el primer argumento de la relación se conoce como *Dominio* y el segundo como *Rango*.
- *Instancias*: Componente utilizado para representar individuos pertenecientes a una clase o concepto.
- *Funciones*: Tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de una ontología.
- *Axiomas*: Afirmaciones que no requieren demostración, se utilizan para aplicar razonamiento.

La clasificación de las ontologías respecto al tipo de conocimiento que contienen consta de los siguientes tres tipos:

- *Ontologías terminológicas o lingüísticas*: Especifican los términos usados para representar conocimiento en el dominio.
- *Ontologías de información*: Especifican la estructura de los registros de la base de datos, ejemplo de ello son los esquemas de bases de datos.
- *Ontologías para modelar conocimiento*: Especifican conceptualizaciones de conocimiento, tienen una estructura interna mucho más rica que los dos tipos anteriores y son las que utilizan los desarrolladores de SBC.

1.3.2 Lenguajes de representación del conocimiento

El lenguaje de representación del conocimiento permite almacenar y recuperar la información que se guarda en una Base de Conocimientos. Estos contienen dos componentes básicos: la sintaxis que se constituye por posibles sentencias o formas de construir y combinar los elementos del lenguaje para representar los hechos del

dominio real y la semántica que se encarga de determinar la conexión o relación entre los elementos o sentencias del lenguaje y su interpretación en el dominio.

Los diferentes tipos de lenguajes de representación del conocimiento son los siguientes:

- *Lenguajes basados en redes semánticas*: Este tipo de lenguajes se han enfocado en su mayoría a aplicaciones sobre el área de lenguaje natural, algunos ejemplos son *SNePS* y *KL-ONE***.**
- *Lenguajes basados en marcos*: Debido a que este tipo de lenguajes en su etapa de programación tiene que tratar con un conjunto de ranuras o propiedades, se consideran más estructurados que los que tratan con redes semánticas, ejemplos de este tipo son: KRL **(**Knowledge Representation Language) y FRL **(**Frame Representation Language).
- *Lenguajes basados en reglas*: Son considerados lenguajes de mayor uso en la creación de sistemas expertos por la facilidad que tienen para hacer inferencias, algunos de los ejemplos de este tipo de lenguaje son: PROLOG y CLIPS.
- *Lenguajes para el desarrollo de ontologías*: Además de la sintaxis y la semántica del lenguaje, es importante considerar algunas características que contiene un lenguaje de manejo de ontologías como las siguientes: debe tener suficiente expresividad para poder capturar varias ontologías, ser fácilmente traducible desde y hacia otros lenguajes ontológicos y ser eficiente a la hora de realizar razonamiento. Entre los lenguajes utilizados para representar ontologías están los siguientes:
    - *OWL* (*Web Ontology Language*): Es un lenguaje de marcado semántico desarrollado para publicar y compartir Ontologías sobre la Web. Es una extensión del vocabulario de RDF y se deriva de DAML+OIL. Está diseñado para ser utilizado por aplicaciones que necesitan procesar el contenido de la información en lugar de sólo presentarla a las personas (Harmelen, 2008). Se compone de tres lenguajes diseñados para ser

utilizados por comunidades específicas de desarrolladores y usuarios, los cuales son: OWL Lite, OWL DL y OWL Full.

Este lenguaje se basa en lógica descriptiva, la cual es un tipo de formalismo de representación del conocimiento que define los conceptos relevantes de un dominio y los relaciona para especificar propiedades. Se diseñó como una extensión de marcos (Minsky, 1975) y redes semánticas, los cuales carecían de semántica basada en lógica formal (Nardi & Brachman, 2002).

En OWL, la representación del conocimiento integra recursos de marcos y lógica descriptiva, incluyendo cinco tipos de componentes: clases, relaciones, funciones, instancias y axiomas. La lógica descriptiva es un formalismo de representación del conocimiento caracterizado por describir los conceptos relevantes de un domino particular y utilizar estos para especificar las propiedades de los objetos e individuos del dominio (Baader, 2010).

- *OIL* (*Ontology Interface Layer*): Es un lenguaje de representación de ontologías basado en Web y capas de inferencia, que combina las primitivas de representación de conocimiento de los lenguajes basados en marcos con la semántica formal y los servicios de razonamiento proporcionados por la lógica descriptiva. Incluye una semántica precisa para describir el significado de los términos. OIL unifica tres aspectos importantes como son: lógica descriptiva, sistemas basados en marcos y estándares Web (XML y RDF).

1.3.3 Herramientas para crear ontologías

La mayoría de las herramientas existentes para crear ontologías soportan los lenguajes OWL, RDF y XML, algunas de las más utilizadas son las siguientes:

- *Protegé*: Es una herramienta que permite la construcción de ontologías de dominio, es capaz de operar como una plataforma para acceder a otros sistemas

basados en conocimiento o aplicaciones integradas, también como una librería que puede ser usada por otras aplicaciones para acceder y visualizar bases de conocimiento con otros formatos. La herramienta ofrece una interfaz gráfica que permite al desarrollador de ontologías enfocarse en la modelación conceptual sin que requiera de conocimientos avanzados sobre la sintaxis de los lenguajes de salida.

Su editor posibilita al usuario la creación y población de ontologías basadas en marcos, utilizando el protocolo OKBC (por sus siglas en inglés, Open Knowledge Base Connectivity). Según este protocolo, una ontología consta de un conjunto de clases organizadas jerárquicamente que representa los conceptos del dominio, un conjunto de slots asociados a las clases para describir sus propiedades y relaciones, y un conjunto de instancias de dichas clases, que son los individuos.

- *Ontolingua***:** Es una herramienta de desarrollo para crear, editar, modificar, y evaluar ontologías. Contiene una librería de ontologías cuyas definiciones, axiomas y términos no-lógicos, pueden ser reutilizadas en la construcción de nuevas ontologías. Está basado en el lenguaje KIF cuyo propósito es solucionar el problema de la heterogeneidad del lenguaje en la representación del conocimiento y permitir el intercambio del conocimiento entre diversos sistemas de información. Es una versión en notación prefija del cálculo de predicados de primer orden, es decir, el símbolo que define la operación a realizar se antepone a los operandos, con varias extensiones para incrementar su expresividad.

1.4 Resumen

Respecto a los algoritmos de clasificación, se eligieron cuatro de los más referenciados en la literatura con diferentes enfoques de solución como son: Naive Bayes de enfoque probabilístico, C4.5 de Árboles de Decisión, KNN basado en vecindad y Perceptrón Multicapa basado en Redes Neuronales. Estos métodos brindan ventajas de precisión, como se verá en el capítulo 6. Sin embargo, el usuario se encuentra con el dilema sobre

cuál de ellos brinda mejor desempeño frente a un determinado conjunto de datos. Por esta razón, se realizó un trabajo experimental donde se evaluó el desempeño de estos métodos frente a diversos conjuntos de datos con diferentes características.

Por su parte, para seleccionar el algoritmo de generación de reglas de asociación a utilizar en el modelo, en la experimentación se evaluaron por una parte los métodos Apriori, Apriori TID, Eclat, FP-Growth y Charm por ser de los más referenciados en la literatura y por otra, los algoritmos Top-K Rules y TNR por ser de los más recientes y de los que profundizan el problema de redundancia en reglas. Debido a que el modelo requiere que el conjunto de reglas generadas por este tipo de algoritmos no sea repetitivo, se efectuó un trabajo experimental para detectar cuál de ellos brinda mejor desempeño frente a dicho problema, esto se expone en la sección 4.2.3.

Finalmente, respecto a los esquemas de representación del conocimiento, esencialmente existen los enfoques basados en marcos o frames, basados en reglas de producción y en ontologías. Para la modelación de reglas de asociación en una base de conocimientos, se utilizó el esquema ontológico, ya que incluye más componentes porque integra tanto elementos de los esquemas basados en marcos o frames como los basados en reglas de producción. Por su parte, el tipo de ontología que se utilizó en el modelo propuesto es de modelación de conocimiento, por contener una estructura más completa y es la más adecuada para utilizar en los Sistemas Basados en Conocimiento.

Capítulo 2
Estado del arte

Debido a la importancia que tienen los modelos predictivos, se han llevado a cabo múltiples investigaciones y desarrollos en diversos campos de aplicación que han derivado en diferentes trabajos, estos se dividen en dos grupos: soluciones basadas con técnicas de Minería de Datos y soluciones basadas en técnicas de Inteligencia Artificial. A continuación, se exponen trabajos relacionados con ambos tipos de soluciones.

2.1 Trabajos de modelos predictivos con técnicas de Minería de Datos

Los trabajos que a continuación se presentan, se clasifican en dos tipos: aquellos que se denominan *Híbridos* debido a que utilizan más de una técnica (primeros dos trabajos) y aquellos que sólo utilizan una técnica.

2.1.1 Hybrid data mining models for predicting customer churn

Este trabajo fue desarrollado por el departamento de Información Tecnológica de la Universidad de Jordán, con el objetivo de proponer tres modelos que permiten predecir la pérdida de clientes (Hudaib, Dannoun, & Harfoushi, 2015). Los modelos se consideran híbridos debido a que hacen uso de dos técnicas de Minería de Datos: Agrupamiento y Clasificación, como se muestra en la Figura 2.1.

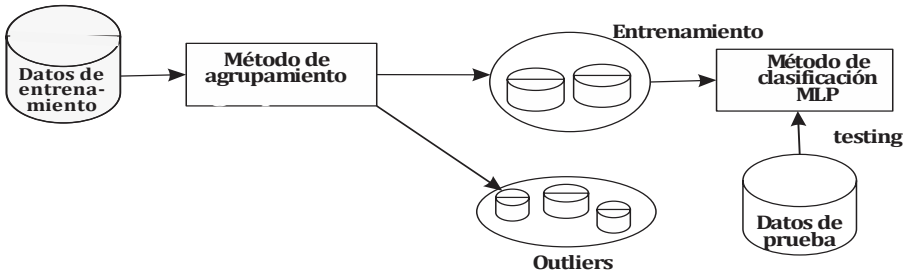


Figura 2.1 Modelo propuesto para predecir pérdida de clientes

En la Figura 2.2, se puede observar que los tres modelos constan de dos etapas. En la primera, cada modelo utiliza un algoritmo de agrupamiento distinto (*K-means*, *Jerárquico* y *SOM*) para filtrar datos y detectar los outliers existentes.

En la segunda etapa (Clasificación), se ocupan como entrada los datos resultantes de la etapa anterior, utilizando en los tres casos el mismo algoritmo MLP (por sus siglas en inglés, Multi Layer Perceptron).

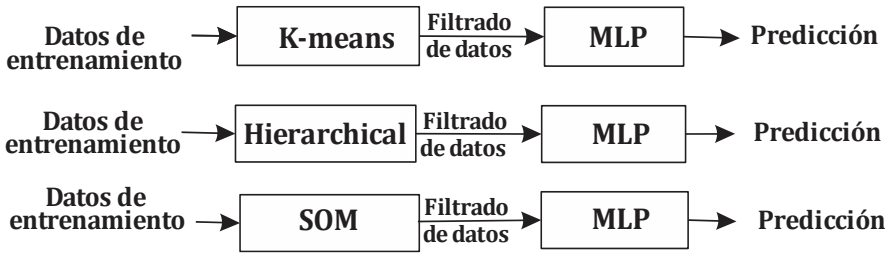


Figura 2.2 Modelos propuestos

En el trabajo experimental, se utilizó un solo conjunto de datos que contiene un total de 5000 instancias con 11 atributos que contiene información referente a empleados de una compañía de Telecomunicaciones. Las medidas de recuperación de información utilizadas fueron precisión y porcentaje de pérdida de clientes, obteniendo los resultados que se muestran en la Tabla 2.1.

Tabla 2.1 Porcentajes de precisión y de pérdida de clientes

	MLP-ANN	K-means + MLP	Hierachical + MLP	SOM + MLP
Precisión	94.3%	97.2%	94.8%	95.9%
Pérdida Clientes	68.3%	73.0%	74.2%	74.6%

De acuerdo con los resultados mostrados en la Tabla 2.1, la combinación de algoritmos Agrupamiento-Clasificación que mostró los mejores resultados respecto a precisión fue *K-Means* + *MLP* y respecto al porcentaje de pérdida de clientes fue SOM + *MLP*. En este trabajo, se propone una alternativa de solución que fortalece la etapa de preparación de datos por medio de la aplicación de la técnica de agrupamiento. El inconveniente del modelo propuesto es que se realizan pruebas con pocos algoritmos de clasificación y la interpretación de los resultados la tiene que realizar el usuario.

2.1.2 Hybrid data mining technique for knowledge discovery from engineering materials

Este trabajo fue desarrollado por el departamento de Ciencias Computacionales de la Universidad de Mangalore la India, tiene el objetivo de predecir los tipos de material que se pueden usar en procesos de ingeniería (Doreswamy & Hemanth, 2011). El

modelo es híbrido, debido a que combina algoritmos de dos enfoques, uno de Minería de Datos (*Naive Bayes*) y otro de Máquinas de Aprendizaje (Pearson correlation coefficient method).

El método Naive Bayes se ocupó para la tarea de clasificación y el método de correlación de Pearson para obtener relaciones entre variables, para lo cual se utilizó una sola base de datos de 5600 instancias y 25 atributos.

En este trabajo, también se desarrolló el software para que el usuario pueda realizar la predicción, permitiéndole seleccionar de una lista algunos valores de atributos sugeridos que le ayuden en la selección del tipo de material.

El modelo tiene como ventaja que le proporciona al usuario un software que le permite efectuar la predicción seleccionar algún tipo de material dentro de una lista de sugerencias, sin embargo, para efectuar la clasificación sólo se realizaron pruebas con el método Naive Bayes.

2.1.3 Applied data mining patterns to discovery survival in women with cervical cancer

Este trabajo fue desarrollado por la Universidad de Nariño Colombia, con la finalidad de detectar los factores que influyen en el diagnóstico de cáncer de cuello uterino utilizando técnicas de Minería de Datos, para esto se utilizó información del Registro Poblacional de San Juan Colombia (Timarán, 2012).

Se trabajó con una base de datos de 507 registros y un total de 22 atributos con datos de tipo personal, socioeconómico y clínico. En este trabajo, se aplicó la metodología CRISP MD y se utilizaron las herramientas Weka y SPSS.

En la etapa de clasificación, se aplicó el método de árboles de decisión J48, obteniéndose como resultado el árbol del Cuadro 2.1, donde el atributo que más influyó fue el número de meses de vida del paciente desde el momento del diagnóstico de cáncer hasta la realización del estudio.

También con el método J48 se obtuvieron algunas reglas de clasificación, las cuales ayudaron a detectar algunos factores socioeconómicos y clínicos que inciden en la enfermedad, dentro de las cuales destacan los siguientes: El 82% de las pacientes

proceden de zona urbana, el 70% conviven con una pareja, el 65% tienen baja escolaridad y el 78% tienen aseguramiento en salud. El conocimiento generado ayudó a tener mayores elementos para la toma de decisiones en el tratamiento de las mujeres que padecen cáncer cervical.

Cuadro 2.1 Resultados del método J48

```
Weka.classifiers.trees.J48 –C 0.7 –M 20
Classiefier model (full trining set) ====
J48 pruned tree
nmeses <= 37
|       cabezaflia <= 0
|       |       nivelsisben = 1: vivo (50.0/20.0)
|       |       nivelsisben = 7: muerto (165.0/38.0)
|       |       nivelsisben = 2: vivo (33.0/6.0)
|       |       nivelsisben = 3: vivo (4.0/1.0)
|       |       nivelsisben = 4: vivo (1.0)
|       cabezaflia > 0: vivonivelsisben = 4: vivo (42.0/4.0)
nmeses > 37: vivo (212.0/15.0)

Number of leaves: 7

Size of the tree: 10
```

En el presente trabajo, se obtuvieron resultados interesantes, pero presenta los siguientes inconvenientes: se limita a realizar el análisis con un solo algoritmo, el método aplicado proporciona algunas reglas que tienen que ser interpretadas por el usuario y finalmente, no se cuenta con una aplicación que ejecute el proceso de clasificación por lo que se tiene que realizar de forma manual en Weka.

2.1.4 Diabetes forecasting using supervised learning techniques

Este trabajo fue desarrollado por el Instituto Africano de Ciencia y Tecnología (Diwani & Sam, 2014), con el fin de detectar los atributos que más influían en la enfermedad de Diabetes Mellitus, para lo cual se utilizó una base de datos denominada *Pime Indians Diabetes* con información de 768 pacientes.

La herramienta utilizada fue Weka, en la etapa de clasificación se utilizó la técnica de validación cruzada con los siguientes clasificadores: BayesNet, OneR y J48.

En la Tabla 2.2, se muestran los resultados obtenidos por los métodos aplicados, el que obtuvo mejores resultados fue Naive Bayes con una precisión del 76,3% de instancias clasificadas correctamente, obteniéndose 422 verdaderos positivos, 104 falsos positivos, 78 falsos negativos y 164 verdaderos negativos. Estos porcentajes

permitieron observar el grado de exactitud que tiene el modelo para ser utilizado en la predicción de los pacientes con diabetes.

Tabla 2.2 Porcentaje de predicción por método

Método	Instancias Clasificadas correctamente	Precisión
Naive Bayes	586	76,3%
J48 (Trees)	567	72,78%
OneR (Rules)	548	71.48%

En el presente trabajo, para seleccionar el método de clasificación, se hacen pruebas con sólo tres métodos y los resultados tienen que ser interpretados por el usuario. Por otro lado, para obtener la clase del nuevo elemento el usuario lo tiene que realizar de forma manual.

2.1.5 Comparación de modelos de clasificación automática de patrones texturales de minerales

Este trabajo fue desarrollado por la Universidad Nacional de Colombia y tiene como objetivo aplicar técnicas de aprendizaje supervisado sobre texturas de minerales presentes en los carbones (López, 2005). Se utilizó una base de datos que contiene información de 400 imágenes con 12 atributos cada una.

En la etapa de preparación de datos, respecto a la textura se trabajó con imágenes 32 x 32 pixeles y se tuvo que aplicar la discretización con el método de rangos iguales. En la etapa de clasificación los métodos aplicados fueron Naive *Bayes*, *Redes Neuronales*, *Máquinas de Soporte Vectorial* y *Árboles de Decisión*. De acuerdo a los resultados obtenidos en la etapa experimental aplicando validación cruzada, el método *Bayesiano* ofreció la mejor capacidad predictiva con un 84%.

En este trabajo, se aplicaron tres enfoques distintos de métodos de clasificación, sin embargo, el usuario tiene que realizar el trabajo necesario de forma manual para efectuar la clasificación e interpretar los resultados.

2.1.6 Churn prediction in Mobile Telecom System using data mining techniques

Este trabajo fue desarrollado por el departamento de Ciencias Computacionales en la Universidad de Anamalai en la India, tiene como objetivo detectar la pérdida de clientes por medio del uso de técnicas predictivas en Minería de Datos (Balasubamanian, 2014). Se utilizó una base de datos con información sobre clientes que contiene un total de 6000 registros con 15 variables, las cuales se clasifican en datos generales de los clientes, formas de pago e historial de compras. Se utilizó el software SPSS, con los algoritmos de árboles de decisión y redes neuronales. En la Tabla 2.3, se muestran los resultados de la matriz de confusión referentes al método de árboles.

Tabla 2.3 Matriz de confusión

Árbol de Decisión		Clase predictora	
		Cliente Perdido	Cliente no Perdido
Actual	Clase = Perdido	833	19
	Clase= No perdido	48	5100

Sobre los resultados conseguidos, se obtuvo un 98.8% de precisión y una tasa de error de 1.11% en el modelo de árboles de decisión, donde en los falsos positivos se obtuvo un 0.93% y en falsos negativos 2.23%.

2.1.7 Predicting students drop out: a case study

Este trabajo fue desarrollado por el departamento de Ingeniería Eléctrica de la Universidad de Eindhoven Holanda (Gerben, 2009), con la finalidad de identificar los atributos que más influyen en la deserción de alumnos de dicha institución, para lo cual se utilizó la base de datos de alumnos que ingresaron en 2008 y 2009.

Se utilizaron datos de 658 alumnos, las variables utilizadas se pueden clasificar en atributos preuniversitarios y universitarios. Los primeros incluyen datos generales como sexo, edad y calificaciones de las materias principales en bachillerato. Los datos universitarios, constan de las calificaciones obtenidas en los primeros dos semestres.

Se utilizó la herramienta Weka con técnica de validación cruzada (10 Fold). En la etapa de clasificación, se utilizaron los siguientes algoritmos: Árboles de decisión CART (SimpleCart) y C4.5 (J48), Bayesiano (BayesNet), Basado en Reglas (OneR) y de conjuntos aleatorios (Forest Random). En la Tabla 2.4 se muestran los resultados

obtenidos por los métodos aplicados con variables de nivel licenciatura, se observa como los métodos que proporcionaron mejores resultados fueron CART y RF.

Tabla 2.4 Porcentajes de algoritmos de clasificación

	Clasificador				
	OneR	j48	CART	BayesNet	Forest Random
Presición	76 %	79 %	**81 %**	75 %	**80 %**

En la Tabla 2.5, se presentan los resultados sobre los atributos que más influyen en la deserción de los alumnos.

Tabla 2.5 Atributos que más influyen en la deserción

	ALGORITMO		
	J48	CART	BayesNet
Matriz de Confusión	212 41 651 98	201 52 57 206	160 93 31 232
Nodo raíz	LinAlgAB <= 5	LinAlgAB < 5.5	LinAlgAB < 5
Primer nodo	NetwB <=5.7 CompB	CalcA <5.15 VWO	CalcA <5 VWO

Como se observa en esta tabla, los atributos más determinantes respecto a los datos universitarios son las calificaciones de las materias de Algebra Lineal, Cálculo y Redes. Respecto a los datos preuniversitarios el que mayor influyó fue el promedio de la materia de Ciencias. El presente trabajo tiene como ventaja que se aplicaron diferentes algoritmos de clasificación y la precisión máxima alcanzada fue del 81%, la desventaja que se visualiza es que no se proporciona una herramienta que evite realizar el proceso de forma manual.

2.1.8 Predicting of school failure using data mining techniques

Este trabajo fue desarrollado por investigadores de la Universidad de Córdoba España y la Universidad Autónoma de Zacatecas México (Márquez, 2013), con el objetivo de utilizar técnicas de Minería de Datos para detectar los factores que más influyen en la deserción de los estudiantes y quienes están involucrados.

Los datos sobre los alumnos se obtuvieron del departamento académico de la Universidad Autónoma de Zacatecas (UAPUAZ) y de encuestas realizadas a estudiantes.

Originalmente, se tenía una base de datos con 77 atributos sobre 670 alumnos, posteriormente se utilizaron métodos de selección de características en la herramienta Weka, quedando finalmente con un total de 15 atributos. Los datos obtenidos, se dividieron en 10 ficheros de entrenamiento y 10 de prueba; por otro lado, para que los datos estuvieran balanceados se aplicó el algoritmo SMOTE (Synthetic Minority Oversampling Technique) con Weka.

Para el trabajo de experimentación, se aplicó la técnica de con reglas de inducción y árboles de decisión, los resultados obtenidos se muestran en la Tabla 2.6. Los métodos que obtuvieron los valores máximos son: ADTree y SimpleCart.

El presente trabajo tiene como ventaja que se aplicaron varios algoritmos de clasificación, lo cual permite comparar cuál es el que brinda mejores resultados, sin embargo, tiene la desventaja de que las diferentes tareas se tienen que hacer de forma manual en la herramienta Weka.

2.1.9 Predictive modeling of hospital readmissions using metaheuristics and data mining

Este trabajo fue desarrollado por la Universidad de Binghamton en Nueva York (Bichen, Jinghe, & Sang, 2015), con la finalidad de crear un modelo predictivo para el riesgo de readmisiones hospitalarias utilizando técnicas de Minería de Datos.

Este modelo se propuso debido a que en el sistema de salud de EEUU existe un alto porcentaje de readmisiones evitables que derivan en una baja calidad de atención durante las estancias de los pacientes en el hospital**.** Algunas variables interesantes que considera el riesgo de readmisión del paciente son: *duración de estancia*, *nivel de gravedad*, *condiciones de comorbilidad* y *uso de salas de emergencia***.**

Los métodos de clasificación utilizados fueron *Redes Neuronales*, *Árboles de decisión* (*Random Forest*) y *Máquinas de Soporte Vectorial*. Respecto a las métricas de rendimiento, las que se evaluaron fueron: *Exactitud*, *sensibilidad* y *especificidad*. De acuerdo a los resultados obtenidos en la experimentación, el método *Máquinas de Soporte Vectorial* fue el que brindó mejores resultados obteniendo un 78.4% de precisión de la predicción global y un 97.3% de sensibilidad.

El resultado de este trabajo sirvió de apoyo para reducir las tasas globales de reingreso hospitalario y permitir que los hospitales utilicen sus recursos de manera más eficiente y mejorar la atención con los pacientes de alto riesgo.

2.2 Trabajos de modelos predictivos con técnicas de Inteligencia Artificial

Los trabajos que se muestran en las siguientes secciones tienen un enfoque semántico, para poder emitir resultados, requieren realizan la representación del conocimiento en esquema ontológico.

2.2.1 Nutrition for Elder Care: a nutritional semantic recommender system for the elderly

Este trabajo fue desarrollado por el Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada, tiene como objetivo desarrollar un sistema recomendador sobre dietas saludables para personas de edad avanzada (Espín, Hurtado, & Noguera, 2016)

El conocimiento se obtuvo a partir de personal experto en nutrición y se representó mediante una ontología en lenguaje OWL donde se integró la taxonomía de los alimentos con sus propiedades nutricionales y el conocimiento extraído de otra ontología existente llamada *AGROVOC* (Organización de Agricultura y Alimentos).

El modelo propuesto permite, por una parte demostrar como el uso del conocimiento extraído a partir de otras ontologías existentes se puede reutilizar y homologar con una ontología propia y por otra parte, también permite demostrar como con el uso de un lenguaje declarativo como OWL se tiene la ventaja de poder realizar inferencias utilizando una serie de axiomas para efectuar razonamiento y poder obtener nuevo conocimiento.

La arquitectura del modelo propuesto está basada en los elementos principales de un sistema recomendador semántico. La base de conocimientos se construyó mediante el uso de una ontología, incluyendo un conjunto de axiomas y el conjunto de instancias. Gracias a las descripciones lógicas integradas, el sistema permite inferir nuevo conocimiento.

2.2.2 Personalized web page recommender system using integrated usage and content

Este trabajo fue desarrollado por el departamento de ingeniería de la Universidad de Karimnagar la India, su finalidad es proponer un modelo de sistema recomendador para páginas web basado en un enfoque semántico (Venu & Sammula, 2014).

Este trabajo presenta un algoritmo para integrar tanto conocimiento del dominio, como conocimiento utilizado en una página web a través de métodos semánticos, además permite construir una jerarquía de datos registrados en ambos tipos de conocimiento.

Este modelo tiene dos fases, la primera es para generar el conocimiento del dominio representado con la ontología del sitio web y la segunda, para generar las asignaciones entre las páginas web y términos de dominio. La ontología se representa en lenguajes OWL o RDF, donde se maneja la generalización, la especialización, la definición de las relaciones para generalizar los conceptos y la cardinalidad de las relaciones. Se realizó el trabajo experimental con tres conjuntos de datos aplicando la métrica de Precisión.

2.2.3 An ontology-based recommender system for Semantic searches in vehicles sales portals

Trabajo desarrollado por el Departamento de Ingeniería Eléctrica de la Universidad de Natal Brasil, con el objetivo de desarrollar un sistema recomendador basado en un enfoque semántico para venta de vehículos por Internet (Paiva, Figueiras, & Lima, 2014). El modelo consta de una arquitectura de cuatro capas: de contexto, de descubrimiento, de recomendación y de ontología. A continuación, se expone cada una de ellas:

- *Capa de contexto*: Permite administrar los diferentes perfiles y los datos relacionados con el usuario así como la función de contexto de dominio específico.
- *Capa de descubrimiento*: Permite la retroalimentación entre los diversos usuarios involucrados.
- *Capa de recomendación*: Esta capa constituye el componente del motor de recomendación.

- *Capa de ontología*: Contiene el repositorio ontológico, cuyo componente es responsable de almacenar artefactos representando instancias desde modelos usados.

Se propuso un modelo donde el sistema recomendador hace uso tanto de una ontología como de la integración de perfiles semánticos de los usuarios. Se diseñó e implementó un portal en línea para la venta de vehículos, el sistema le muestra al usuario la información mediante una página Web. Por su parte, la ontología incluye conceptos relacionados con el dominio de recomendaciones de los vehículos.

2.3 Análisis de los trabajos sobre modelos predictivos

De los trabajos analizados, la mayoría de los que aplican técnicas de Minería de Datos coinciden en hacer uso de algoritmos como C4.5, Naive Bayes y KNN, respecto a las herramientas o aplicaciones, las que más se utilizaron fueron Weka y Rapid Miner. Por otra parte, los que utilizan técnicas de Inteligencia Artificial coinciden en utilizar el lenguaje OWL para representar las ontologías. En la Tabla 2.6, se presenta una comparativa con las principales características de cada uno de los trabajos anteriormente expuestos, incluyendo el modelo propuesto.

En los trabajos que utilizan técnicas de Minería de Datos, se detectaron los siguientes inconvenientes: no muestran evidencia de que el método de clasificación utilizado es el que brinda el mejor desempeño, la predicción se efectúa de forma manual con herramientas como Weka, Rapid Miner o SPSS sin contar con un sistema propio, se limitan a la solución de un caso específico sin demostrar su funcionamiento en otras áreas de aplicación y se enfocan únicamente a proporcionar la clase buscada.

Respecto a la combinación de técnicas utilizadas en el modelo, los trabajos que se consideran híbridos son únicamente los primeros dos de la Tabla 2.6. El primero de ellos se considera híbrido porque combina dos técnicas de minería de datos: clasificación y agrupamiento; el agrupamiento o clustering se utilizan con la finalidad de detectar outliers para detectar instancias que contienen valores atípicos y el segundo, porque combina un método de algoritmos genéticos con uno de redes neuronales para efectuar la predicción teniendo como objetivo principal aumentar la

precisión. Ninguno de estos dos trabajos representa el conocimiento extraído en un lenguaje que pueda ser interpretado por el ordenador, como se sugiere en el modelo de este trabajo.

Tabla 2.6 Tabla comparativa de las características de trabajos analizados

Trabajo / Institución	Técnica aplicada	Modelo híbrido	Emite Recomendaciones	Aplicación genérica
Hybrid Data Mining Models for Predicting Customer Churn. The University of Jordan, Amman, Jordan	Clasificación y Agrupamiento	Si	No	No
Hibrid data mining technique for knowledge discovery from engineering materials data sets. Universidad de Magnalore en la India.	Clasificación y Máquinas de Aprendizaje	Si	No	No
Applied Data Mining patterns to discovery survival in women with invasive cervical cancer. Universidad de Nariño Colombia.	Clasificación	No	No	No
Diabetes Forecasting Using Supervised Learning Techniques. Instituto Africano de Ciencia y Tecnología	Clasificación	No	No	No
Comparación de modelos de clasificación automática de patrones texturales de minerales en los carbones colombianos. Universidad Nacional de Colombia	Clasificación	No	No	No
Applying Data Mining Techniques in Telecom Churn Prediction. Universidad de Anamalai en la India	Clasificación	No	No	No
Predicting of school failure using data mining techniques. Universidad de Córdoba España	Clasificación	No	No	No
Predictive modeling of hospital readmissions using metaheuristics and data mining. Universidad de Binghamton Nueva York	Clasificación	No	No	No
Nutrition for Elder Care: a nutritional semantic recommender system for the elderly. Universidad de Granada	Técnicas de IA	No	Si	No
Personalized Web Page Recommender System using integrated Usage and Content Knowledge. Universidad de Karimnagar la India	Técnicas de IA	No	Si	No
An Ontology-Based Recommender System Architecture for Semantic Searches in Vehicles Sales Portals. Universidad de Natal Brazil	Técnicas de IA	No	Si	No
Modelo híbrido propuesto	Clasificación Reglas de asociación Técnicas de IA	Si	Si	Si

Por su parte, los trabajos que utilizan técnicas de Inteligencia Artificial sólo hacen uso del conocimiento proporcionado por el experto y utilizan como esquema de representación del conocimiento las ontologías, teniendo como inconveniente que no se utiliza como fuente de datos información real o histórica que es de suma importancia para que el modelo brinde resultados más precisos debido a que, si bien es cierto que el conocimiento del experto es de gran valor, pueden omitirse comportamientos o patrones que se presentan con alta frecuencia.

En las siguientes tres secciones, se exponen trabajos relacionados con los métodos utilizados en el modelo propuesto en este trabajo, los cuales se exponen detalladamente en los capítulos cuatro, cinco y seis.

2.4 Trabajos desarrollados sobre análisis comparativo de métodos de reglas de asociación

En los últimos años, se han llevado a cabo algunos estudios dirigidos al análisis comparativo de los algoritmos de generación de reglas de asociación, considerando diferentes características como la cantidad de recorridos efectuados, la velocidad y el manejo de reglas redundantes. Un análisis de los trabajos donde se presentan estos estudios, se describe a continuación.

Entre los trabajos que analizan la cantidad de recorridos realizados para obtener el conjunto los Itemsets Frecuentes, se encuentra el realizado por Vivekananth (2012) a través de un estudio comparativo de los algoritmos Apriori y FP-Growth. En este trabajo, se muestra que FP-Growth es mejor ya que ejecuta sólo dos recorridos a la base de datos mientras que Apriori requiere efectuar varios debido a que necesita generar previamente un conjunto de itemsets denominados Candidatos para obtener los FI. Otro trabajo similar lo llevaron a cabo Solanki & Soni (2013) donde además de Apriori y FP-Growth también incluyeron el método Eclat; en este estudio se demostró al igual que el anterior que para obtener los FI, Apriori requiere recorrer la base de datos varias veces, pero además se revela que este algoritmo demanda más recursos computacionales cuando la cantidad de itemsets a generar es grande. Por su parte Eclat inicia el proceso realizando un solo recorrido sobre el conjunto de datos original y los

restantes los ejecuta sobre una lista auxiliar, lo cual demanda menos tiempo de ejecución que Apriori que contiene los identificadores de transacciones, mientras que FP-Growth realiza dos recorridos porque se auxilia de una estructura de árbol. De manera general, los autores concluyen que FP-Growth es el que brinda mejores resultados respecto a la velocidad, aunque su estructura es más compleja por lo que demanda más memoria.

En otro trabajo, Saxena & Gadhiya (2014) efectuaron un estudio con los mismos algoritmos que el trabajo anterior, además de analizar la cantidad de recorridos realizados, compara si se trabaja con itemsets o identificadores de transacciones. Hace un análisis de cómo los tres algoritmos obtienen los FI de manera distinta puesto que Apriori realiza los recorridos de forma horizontal, Eclat de forma vertical (donde se trabaja con los identificadores de transacciones en lugar de los itemsets) y FP-Growth un recorrido sobre árboles, el autor concluye que Eclat quien trabaja de manera vertical ocupa menos espacio y tiempo que Apriori y que FP-Growth es el más rápido de los tres.

Bonde & Gore (2014) realizaron otro estudio para analizar la forma la forma de obtener los itemsets frecuentes y la velocidad con la que trabajan los algoritmos Apriori y FP-Growth para obtener los itemset frecuentes, propusieron una mejora a FP-Growth en la forma de estructurar los datos, utilizando una técnica llamada Sistólica que consiste en reordenar el árbol según la prioridad de sus nodos, con lo cual logra decrementar el número de iteraciones. Los resultados demuestran que con la modificación propuesta, el algoritmo es aproximadamente un 12% más rápido que el algoritmo FP-Growth original.

Girotra et. al. realizaron un estudio para analizar tiempo de ejecución y memoria con la que trabajan los algoritmos Apriori, Eclat, FP-Growth, Apiori TID, Apriori Hybrid, AIS, SETM y Recursive Elimination (Girotra, Nagpal, & Se Minocha and Ne Sharma, 2013). Ellos concluyen que AIS es lento debido a que genera conjuntos de itemsets candidatos muy grandes, que Apriori TID es mejor que SETM con cualquier conjunto de datos y mejor que Apriori cuando se trata con bases de datos pequeñas, que Apriori Hybrid es mejor que Apriori y Apriori TID con bases de datos grandes, que Recursive Elimination es más rápido que Apriori pero más lento que Eclat, que Eclat es de los que ocupa menos

memoria y que FP-Growth aunque ocupa una estructura más compleja (estructura de árbol) requiere menos tiempo que todos los demás. Un trabajo similar al anterior lo realizaron Prithiviraj & Porkodi (2015) con los mismos algoritmos a excepción del método Recursive Elimination, ellos concluyen en su estudio que Apriori Hybrid y FP-Growth son los que trabajan mejor con conjuntos de datos grandes, siendo aún más rápido FP-Growth.

Divya & Kumar (2012) realizaron un estudio de los algoritmos Apriori y FP-Growth, donde utilizaron tres conjuntos de datos y se enfocaron principalmente en detectar las desventajas de cada método. Ellos concluyeron que la principal desventaja de Apriori se presenta en la etapa de obtención de itemsets candidatos debido a que requiere realizar más recorridos a la base de datos, lo cual demanda más recursos computacionales en conjuntos de datos que contienen miles de transacciones y por otro lado, la desventaja de FP-Growth es que se requiere repetir todo el proceso de creación del árbol cuando se cambia el valor del soporte o cuando se agrega una nueva transacción.

Respecto a estudios que se enfocan únicamente en analizar los tiempos de ejecución, Sinha & Ghosh (2014) compararon los algoritmos Apriori, Eclat y FP-Growth, utilizando la base de datos Pime Diabetes y aplicando siete valores distintos en soporte y confianza. Ellos también comprobaron la superioridad en velocidad de FP-Growth sobre el resto de los algoritmos analizados. Por otra parte, Vani (2015) comparó los mismos algoritmos que el trabajo anterior, pero experimentó con cuatro conjuntos de datos combinando diferentes valores en soporte y confianza, sus resultados demuestran que tanto FP-Growth como Eclat fueron más rápidos.

Por último, respecto a estudios que se enfocan principalmente en cómo trabajan los algoritmos la redundancia en las reglas, Jayavani & Nawaz (2014) presentan un análisis comparativo con los algoritmos Apriori, FP-Growth, Close y Charm, destacando que los dos últimos permiten obtener itemsets frecuentes que no presentan redundancia, denominándose a los mismos Itemsets Frecuentes Cerrados (FCI por sus siglas en inglés Frequent Close Items), la estrategia utilizada por estos algoritmos decrementa la cantidad de itemsets frecuentes generados. Los autores analizaron cómo trabaja cada

algoritmo y concluyen que Charm es más rápido cuando los valores del soporte son más bajos.

De igual forma, Zaki & Hsiao (2002) realizaron un estudio de los algoritmos Apriori, Close, Mafia y Charm, utilizando seis bases de datos donde demostraron que a excepción de Apriori, todos estos algoritmos son capaces de generar FCI para eliminar redundancia. De acuerdo a sus resultados experimentales, Charm y Close manejan mejor la redundancia debido a que obtuvieron una menor cantidad de reglas, logrando descartar las que no aportan información nueva.

2.5 Trabajos relacionados con representación de reglas en esquema ontológico

En los últimos años, se han desarrollado trabajos que involucran el uso de reglas de asociación y ontologías, la mayoría de ellos se enfocan a integrar el contenido de reglas de asociación en una ontología existente para enriquecer el conocimiento sobre el dominio. Algunos de estos trabajos se describen a continuación.

En el trabajo de Ghezaiel et al. (2012), se propone una teoría para tratar los términos relevantes identificados en documentos de texto. Estos términos se encuentran a través de la generación de reglas de asociación aplicadas a dichos documentos. Ello se realiza con la finalidad de enriquecer el conocimiento de una ontología existente con patrones de comportamiento identificados. El proceso se divide en tres etapas: detectar nuevos términos y sus relaciones, agregar los elementos localizados en la BC y por último, evaluar la vecindad con el fin de vincular los nuevos términos extraídos con los ya existentes.

Paiva et al. (2014) desarrollaron un trabajo que tiene como finalidad utilizar la minería de reglas de asociación para descubrir patrones en fuentes no estructuradas. Los patrones identificados se comparan con los conceptos de una ontología existente a través de una medida de similitud y dependiendo de la semejanza entre ambos términos, se enriquecerá dicha ontología. Para esto, proponen un proceso que consta de cuatro módulos: análisis de documentos, FP-Growth, reglas de asociación y mapeo de itemsets frecuentes.

Saheb et al. (2016) exponen un nuevo método que utiliza el conocimiento extraído de grandes bases de datos mediante la generación de reglas de asociación, para enriquecer las ontologías existentes en una BC con nuevas relaciones semánticas identificadas entre conceptos, ello con el propósito de incrementar el conocimiento del dominio. El método que proponen contiene los siguientes seis pasos: Representación de las relaciones ontológicas en formato de reglas, extracción de reglas con el algoritmo Apriori, mapeo de los items de las Reglas de Asociación de la base de datos a los conceptos de la ontología, clasificación de las reglas de asociación en tres categorías (nuevas, existentes e inesperadas), validación de las nuevas relaciones por el experto en el dominio y finalmente, el enriquecimiento de la ontología con las nuevas relaciones validadas.

Además de los trabajos expuestos anteriormente, se han desarrollado algunos que utilizan reglas de asociación y ontologías con otros fines, ejemplo de ello es el trabajo realizado por Marinica y Guillet (2009) donde se propone un modelo de post-procesamiento de las reglas de asociación generadas a partir de una base de datos, con la finalidad de eliminar redundancia. Su modelo se compone de dos representaciones de conocimiento: una ontología de dominio que contiene conceptos y relaciones del contexto de la base de datos y otra que se deriva de la anterior, representada con esquemas de reglas. Estas últimas se comparan con las reglas de asociación generadas de la base de datos con el objetivo de desechar aquellas no están presentes en la ontología.

Tatsiopoulos y Boutsinas (2009) propusieron una técnica con la finalidad de aplicar mapeo de ontologías donde, a partir de dos ontologías se puede comparar equivalencia de conceptos entre éstas, sin requerir la intervención del usuario. La técnica propuesta considera que la similitud entre conceptos de dos ontologías está dada por su localización dentro de la estructura ontológica. Para evaluar la similitud, en esta teoría se representa cada ruta de la estructura ontológica como una transacción.

2.6 Trabajos sobre análisis comparativo de algoritmos de clasificación según las características del conjunto de datos

En los últimos años, se han desarrollado algunos trabajos dirigidos al análisis comparativo de algoritmos de clasificación, considerando aspectos como velocidad de ejecución, precisión y tipos de datos tratados.

Akinola y Oyabugbe (2015) realizaron un estudio con los algoritmos Árboles de Decisión (AD), Naive Bayes y Perceptrón multicapa respecto a su velocidad de ejecución, utilizando un solo conjunto de datos. En este estudio, se pudo observar que el algoritmo Naive Bayes consumía menos tiempo en ejecutar el proceso de clasificación, sin embargo, el haber considerado una sola población, no permitió identificar alguna dependencia entre la velocidad y algunas otras características de la población como por ejemplo, la cantidad de instancias o los tipos de atributos. Otro trabajo similar lo realizaron Ashari et al. (2013) con los algoritmos AD, Naive Bayes y k-Vecinos más cercanos (KNN) y con cinco bases de datos. Después de realizar su trabajo experimental, mostraron en sus resultados que el algoritmo AD fue el más rápido, sin embargo estos resultados requieren constatarse con una mayor cantidad de conjuntos de datos, porque con mayores cantidades de instancias o atributos los resultados pueden cambiar.

Respecto a estudios realizados para comparar la precisión de los clasificadores dependiendo de las características del conjunto de datos tratado, Entezari et al. (2009) compararon los algoritmos KNN, LogR, Naive Bayes, AD, C4.5, Máquinas de Soporte Vectorial y Linear Classifier (LC), tomando en cuenta características como: cantidad de instancias, tipo de atributos y número de atributos discretos o continuos. Para realizar este estudio, los autores generaron 29 conjuntos de datos sintéticos, mismos que se organizaron en cuatro grupos de acuerdo al número de atributos que contenían (3, 5, 7 y 10, respectivamente). Con el fin de tener una diversidad de poblaciones, de cada grupo de datos se consideraron para la creación de los conjuntos distintas cantidades de atributos numéricos y discretos y distintas cantidades de instancias (200, 500, 1000, 3000 y 5000). De acuerdo a los resultados, se detectó que mientras más instancias se procesan, más explícita es la diferencia entre la precisión de los clasificadores. Los

algoritmos que mejor desempeño mostraron fueron KNN, SVM, AD y C4.5, no importando si el número de instancias o de atributos aumentaba. Un comportamiento particular se observó con SVM, quien obtuvo mejor precisión que KNN cuando la cantidad de atributos numéricos es mayor y en caso contrario, KNN trabaja mejor con los discretos. Sólo se analizan los atributos número de instancias y tipo de atributo, omitiendo otras que pueden incidir en los resultados que proporciona el clasificador sobre determinada población.

Otro estudio lo desarrollaron Moran et al. (2009), donde se procesaron 39 bases de datos, las cuales se agruparon en un total de 12 conjuntos mediante la combinación de tres de sus características: número de instancias, total de atributos y porcentaje de atributos categóricos. Tomando la cantidad promedio del número de instancias de las poblaciones tratadas (286 instancias), los conjuntos se separaron en dos grupos, los que tenían más de esta cantidad de instancias de los que tienen menos o igual. Cada uno de estos grupos se subdividió en dos grupos de acuerdo al número de atributos, en los que tenían más de 16 y los que tenían menos o igual que 16. Por último, se subdividió cada subgrupo anterior en tres, ahora de acuerdo al número de atributos nominales, con el 100% de atributos de este tipo, con más del 50% y con menos o igual que el 50%. En este estudio sólo se utilizaron clasificadores que se derivan o son variantes del algoritmo Naive Bayes, siendo estos: Averaged One Dependence Estimator (AODE), Tree Augmented Naive Bayes (TAN), BN K2, Genetic Search (BN-GS) y Simulated Annealing (BN-SA). El trabajo se divide en dos etapas, primero se probaron los algoritmos con los distintos conjuntos de datos para verificar cuál funciona mejor y posteriormente, considerando los resultados y con el uso del algoritmo J48 se generó un conjunto de reglas de clasificación que ayudan a seleccionar el mejor clasificador para un conjunto de datos particular.

En los resultados de la primera etapa detectaron que el clasificador que brindó mejores porcentajes de precisión fue Tree Augmented Naive Bayes (TAN) para bases de datos con atributos numéricos o combinados y el algoritmo ODE para datos 100% nominales. Respecto a las reglas obtenidas en la segunda etapa, se comprobó con los conjuntos de datos iniciales obteniéndose un 78% de efectividad en su aplicación sobre dichos

conjuntos. A pesar de los resultados alcanzados, no se efectúa una validación con nuevos conjuntos de datos para probar la efectividad de las reglas de clasificación obtenidas. Además, sería de gran interés observar el comportamiento de estas reglas con clasificadores de enfoques distintos al probabilístico.

2.7 Resumen

La mayoría de los trabajos realizados sobre análisis comparativo entre algoritmos de generación de reglas se enfocan principalmente en verificar qué método efectúa el proceso con mayor velocidad o utiliza menos cantidad de memoria, siendo pocos los que tratan el problema de redundancia. Los trabajos que se enfocan en analizar el comportamiento de los algoritmos respecto a la redundancia, no involucran métodos actuales como Top-K Rules y TNR. Tampoco se hace un estudio exhaustivo de cuáles son los criterios que permitieron obtener menor cantidad de reglas. En este trabajo, es importante realizar un análisis comparativo sobre el desempeño de los algoritmos respecto al problema de redundancia, debido a que cuando se genera una cantidad excesiva de reglas con información repetitiva, hace compleja su interpretación.

Por su parte, los trabajos relacionados con la representación de reglas en esquema ontológico se enfocan principalmente en buscar una equivalencia entre los componentes de una regla de asociación y una ontología; para representar el conocimiento recurren a componentes como conceptos, propiedades y relaciones, presentando el inconveniente de que no utilizan axiomas, los cuales son un componente de suma importancia para el modelo que se propone en este trabajo, debido a que permiten aplicar razonamiento en la base de conocimientos integrada y con ello ayudan a lograr la obtención de las recomendaciones.

Otro inconveniente detectado es que las reglas se generan con algoritmos como Apiori o FP-Growth, teniendo estos métodos el inconveniente de no tratar de manera interna el problema de redundancia.

Finalmente, respecto a los trabajos realizados sobre análisis comparativos de algoritmos de clasificación según las características del conjunto de datos, los métodos más utilizados son Naive Bayes y J48.

Algunos de los inconvenientes que se detectaron en dichos trabajos son: las pruebas se realizaron con conjuntos de datos con pocas características, la cantidad de conjuntos de datos es pequeña y en la etapa de experimentación, no se incluyen algunas características de los conjuntos de datos como número de clases, tipo de balanceo y datos faltantes.

Capítulo 3
Modelo propuesto

Un modelo predictivo típico aplica técnicas de Minería de Datos o de Inteligencia Artificial como se muestra en la Figura 3.1(a), sin embargo, considerando las limitaciones que presentan dichos modelos, en el presente trabajo se propone un modelo híbrido porque incluye ambas técnicas, incluso dentro de la Minería de Datos también es híbrido porque involucra tanto la clasificación como la generación de reglas de asociación., lo cual se muestra en la Figura 3.1(b).

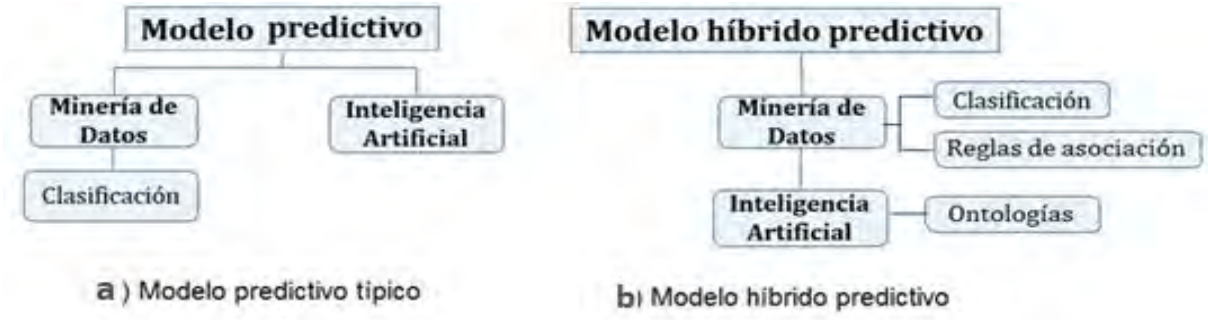


Figura 3.1 Diferencia entre modelo predictivo típico e híbrido

3.1 Componentes del modelo híbrido predictivo

Comúnmente, para crear un modelo predictivo se utiliza una vista minable y un método de clasificación contenido en una herramienta de Minería de Datos. No obstante, el modelo híbrido propuesto involucra más tareas, en una primera fase, obtiene los patrones de comportamiento mediante la técnica de generación de Reglas de Asociación de Clases a partir de una vista minable, esto con el objetivo de identificar las situaciones generales y también aquellas que son particulares pero relevantes en el conjunto de datos bajo estudio. En una segunda fase, se modelan los patrones identificados en un sistema basado en conocimiento bajo un esquema ontológico, lo que permitirá de manera natural, realizar recomendaciones pertinentes al usuario. En la tercera fase, se clasifica una instancia a partir de sus características y la clase brindada por el clasificador ayudará al modelo a emitir una serie de recomendaciones en caso de que se requieran, esto se puede observar en la Figura 3.2.

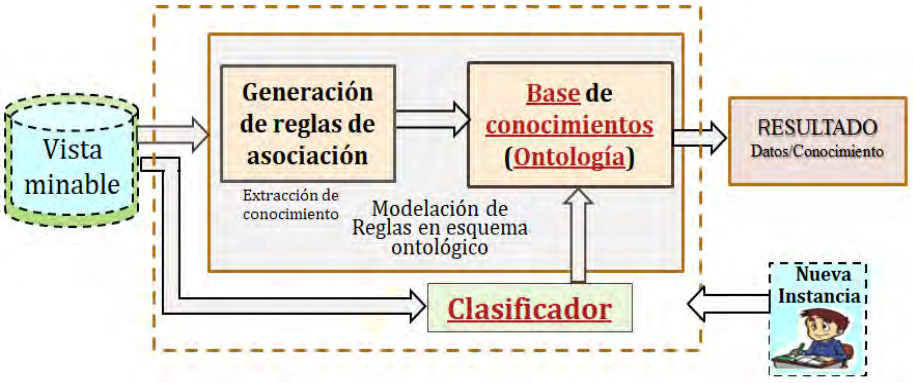


Figura 3.2 Modelo híbrido predictivo

Cada componente del modelo propuesto será descrito en las próximas secciones.

3.1.1 Vista minable

La vista minable es el conjunto de datos que ha sido previamente preprocesado, es la componente determinante para que en los procesos o tareas del modelo se obtenga conocimiento válido y útil.

La calidad del conocimiento descubierto no sólo depende de los algoritmos de Minería de Datos utilizados, sino también de la calidad de los datos a ser minados, por ello para constituir la vista minable primero es necesario recopilar e integrar los datos y posteriormente, prepararlos aplicando las tareas requeridas como limpieza y transformación de datos.

3.1.2 Generación de reglas de asociación

Esta tarea consiste en utilizar los datos de la vista minable para poder encontrar los patrones de asociación que se presentan con mayor frecuencia, cuyo conocimiento servirá de base para brindarle al usuario en los casos que sea posible una serie de recomendaciones.

Actualmente, existe una gran variedad de algoritmos que permiten la generación de este tipo de reglas, teniendo cada uno de ellos características que lo hacen único. Para elegir un algoritmo de generación de reglas de asociación, uno de los principales criterios que debe tomarse en cuenta es su eficiencia al trabajar la redundancia de reglas, ya que evitándola se obtiene una interpretación más clara y sencilla del resultado.

En este modelo, se propone utilizar el método TNR, debido a que después de realizar la experimentación que se muestra en el capítulo cuatro, resultó ser el que brinda los mejores resultados principalmente respecto al problema de redundancia.

3.1.3 Modelación de reglas en esquema ontológico

En esta etapa, se modelan los patrones identificados en un sistema basado en conocimiento bajo un esquema ontológico, lo que permitirá de manera natural, realizar recomendaciones pertinentes al usuario. El proceso de dicha modelación es otra de las aportaciones realizadas en este trabajo y se expone a detalle en el capítulo cinco.

La modelación de las reglas de asociación de clase generadas anteriormente, permite conformar una base de conocimientos que cuente con los componentes necesarios para efectuar razonamiento. Una vez modelada la nueva ontología, es posible representarla en el lenguaje OWL.

3.1.4 Base de conocimientos

Contiene el conocimiento extraído con técnicas de Minería de Datos modelado en una ontología, es un componente indispensable para proporcionar las recomendaciones al usuario. La ontología modelada está constituida por un conjunto de conceptos, propiedades y axiomas, estos últimos con el uso del razonador se le aplican a los datos de la nueva instancia para extraer las sugerencias en los casos que sean posible.

La ontología se representa en lenguaje OWL, debido a las siguientes ventajas que contiene:

- Es un lenguaje con alto poder expresivo para la representación del conocimiento en esquema ontológico.
- Permite ejecutar los axiomas generados con diferentes razonadores.
- Contiene un su lenguaje denominado *OWL DL*, considerado un lenguaje completamente descriptivo.

3.1.5 Clasificador

En los modelos predictivos, se aplica un algoritmo de clasificación sin considerar si es el que brindará los mejores resultados. Aunque actualmente existen diversos algoritmos que ejecutan la tarea de clasificación, uno de los dilemas con los que se enfrenta un usuario al realizar la tarea de predicción es la selección del algoritmo que responda con mayor eficacia de acuerdo a las características de la base de datos en tratamiento.

En este modelo, se propone utilizar un conjunto de criterios para seleccionar el algoritmo a utilizar de acuerdo a las características de la base de datos en uso. Estos criterios son el resultado de un estudio experimental expuesto en el capítulo seis, donde se aplicaron a diversos conjuntos de datos con diferentes características, los algoritmos más referenciados en la literatura.

3.2 Resultado que brinda el modelo

Con el modelo propuesto, cuando se efectúa la tarea de predicción, se le solicitan al usuario los datos de la nueva instancia, a partir de estos se obtiene la categoría de clase y de ser posible una serie de recomendaciones. Para el caso de las recomendaciones, se pueden presentar los siguientes dos escenarios:

- Considerando las características de la nueva instancia, se pueden proporcionar recomendaciones obtenidas a partir de patrones frecuentes detectados, que corresponden a la misma categoría que obtuvo el clasificador.
- Según las características de la nueva instancia, se pueden proporcionar recomendaciones obtenidas a partir de patrones frecuentes detectados, que corresponden a una categoría distinta a la que obtuvo el clasificador.

Capítulo 4
Generación de reglas de asociación

En el modelo propuesto, se ha planteado obtener conocimiento de la base de datos bajo estudio a través de la generación de reglas de asociación, dada su capacidad de encontrar correlaciones de esta base (Rastogi & Shim, 2002). Para elegir el método a utilizar, se analizó la forma en que trabajan los algoritmos más referenciados en la literatura con el fin de detectar aquel que brinda el mejor desempeño.

El presente capítulo inicia mostrando las técnicas que existen para eliminar redundancia en reglas y posteriormente, presenta un análisis comparativo de siete de los algoritmos de generación de reglas más referenciados en la literatura, enfocándose en cómo manejan dicha redundancia.

4.1 Redundancia en reglas de asociación

Como se expuso de manera detallada en el capítulo uno, una regla de asociación es una implicación de la forma $X \Longrightarrow Y$, siendo X el antecedente y Y el sucedente o consecuente, donde ambos componentes pueden estar conformados por más de un atributo.

A pesar de que la extracción de reglas de asociación es una de las técnicas elementales dentro de la Minería de Datos, para poder encontrar información valiosa, los resultados pueden variar dependiendo de qué algoritmos se apliquen y con qué valores de sus parámetros se inicie. Si son establecidos valores muy altos de soporte y confianza, se obtienen pocas reglas y si se decrementan estos valores, puede aumentar considerablemente la cantidad de reglas obtenidas y por consiguiente, aparecer redundancia en las reglas.

Antes de aplicar alguno de los métodos de generación de reglas, uno de los aspectos de mayor importancia a considerar es saber cómo atienden el problema de redundancia.

La redundancia en reglas provoca que la interpretación de los resultados sea una tarea más compleja para el usuario, se ha trabajado en este aspecto, mismo que consiste en eliminar aquellas reglas complementarias que no aportan algún conocimiento

agregado, básicamente las reglas que se superponen con otras y que tienen los mismos valores en sus parámetros de soporte y confianza.

El tratamiento de redundancia puede llevarse a cabo implícitamente dentro del propio algoritmo de generación de reglas de asociación, sin embargo, no todas las implementaciones lo incluyen, siendo necesario realizar esta tarea mediante algoritmos diseñados para tal fin y aplicándose posteriormente a la ejecución del método de generación de reglas.

La mayoría de los métodos desarrollados para generar reglas de asociación, hacen su obtención a partir de la detección de los itemsets frecuentes sin verificar si contienen elementos repetidos, con esto, si hay redundancia en ellos, también lo hay en las reglas.

Dependiendo del momento en que se aplica el proceso de eliminación de la redundancia, existen dos formas:

- *Interna*: Cuando la eliminación de redundancia se efectúa dentro del propio algoritmo.
- *Externa*: Cuando se requiere aplicar un proceso de eliminación externa, después de haberse generado las reglas.

Los algoritmos que incluyen la eliminación de redundancia de forma interna, siguen una estrategia que consiste en verificar que los itemsets frecuentes que se obtienen sean cerrados, lo cual se explica a continuación.

Sea $I = \{i_1, i_2, \dots, i_k\}$ un conjunto de k valores de atributos también llamados Ítems y X un itemset frecuente. Para que un itemset frecuente sea cerrado, se deben cumplir las siguientes condiciones:

- Sea F el conjunto de todos los itemsets frecuentes definido como: $F = \{X \mid X \subseteq I$ y soporte(X)> minsop$\}$.
- Sea FCI el conjunto de itemsets frecuentes cerrados, determinado por: $FCI = \{X \mid X \in F$ y $X_1 \subseteq X_2$, soporte $(X_2) \geq$ soporte $(X_1)\}$.

Para ejemplificar la obtención de un grupo de itemsets frecuentes cerrados se considera el conjunto de datos mostrado en la Figura 4.1(a), el cual contiene cinco diferentes items I={B,C,D,T,W} y seis transacciones T={1,2,3,4,5,6}. Si por ejemplo, a minsop se le asigna un valor de 0.50, se generarían todos los conjuntos de itemsets frecuentes que se muestran en 4.1 (b).

Datos Originales

Transacción	Items
1	BCTW
2	CDW
3	BCTW
4	BCDW
5	BCDTW
6	CDT

(a)

Itemsets generados con soporte = 50%

Soporte	Itemsets
100%	C
83%	W, CW
67%	B, D, T, AC, BW CD, CT, BCW
50%	BT, DW, TW, BCT, BTW, CDW, CTW, BTCW.

(b)

Figura 4. 1 Datos origen e itemsets frecuentes

Para soporte igual a 0.50, hay dos FCI: BCTW y CDW, debido a que para el primer caso los itemsets BT, TW, BCT, BTW y CTW están contenidos en BCTW y para el segundo, DW lo está en CDW.

Por otra parte, respecto a la eliminación de redundancia de forma externa, Liu (1999) propone una técnica de post-procesamiento para algoritmos que obtienen las reglas a partir de un conjunto de itemsets frecuentes y que no atienden el problema de redundancia internamente. Esta técnica consiste en aplicar restricciones a las reglas de asociación que se caracterizan por contener un solo item en el sucedente.

El criterio básico consiste en que si una primera regla X_i es subconjunto de otra llamada X_j y X_i tiene mayor o igual soporte y confianza, entonces se elimina X_j por considerarse a X_i más general. Por ejemplo, si se tienen las siguientes dos reglas:

X_1: A=1 → C=1 [sop = 60%, conf = 90%]

X_2: A=1, B=1 → C=1 [sop = 40%, conf = 90%]

La regla X_1 generaliza la segunda ya que ambas contienen A=1 y el soporte de X_1 es mayor que el de X_2, por lo que se elimina X_2 ya que resulta irrelevante.

Con el objetivo de contrastar las estrategias de tratamiento de redundancia interna y externa, en la siguiente sección se presenta la experimentación con los algoritmos Apriori, Eclat, Apriori TID, FP- Growth, Close, Charm, Top-K Rules y TNR, no sin antes profundizar en la teoría en la que se basan los algoritmos utilizados.

Como se expuso en la sección 2.4, la mayoría de los trabajos coinciden en considerar para el estudio los métodos Apriori, FP-Growth y Eclat, los cuales tienen todo un aval de uso, sin embargo, no consideran algoritmos más recientes como Top-K Rules o TNR. También se detectó que la mayoría de los estudios que comparan algoritmos de reglas de asociación se enfocan más al análisis de cómo se obtienen los itemsets frecuentes o la rapidez para obtener sus resultados sin hacer énfasis en la cantidad de reglas generadas.

Debido a esto, uno de los objetivos del presente trabajo es evaluar y comparar el desempeño respecto al problema de redundancia de diversos algoritmos de generación de reglas de asociación como Apriori, Apriori TID, FP-Growth, Eclat, Closet y Charm, así como algunos más recientes como Top-K Rules y TNR. Además, se mostrará cómo estos algoritmos tratan la redundancia de reglas.

Siendo T el universo de transacciones de una base de datos, cada una de las cuales contiene un conjunto de k ítems, referidos a los valores de los atributos, el soporte de una regla de asociación $X \rightarrow Y$ es la fracción de transacciones de T que contienen $X \cup Y$ y la confianza de $X \rightarrow Y$ es la fracción de transacciones que contienen a $X \cup Y$ de entre el total de aquellas que contienen a X.

El soporte se utiliza para determinar qué conjuntos de items se presentan con mayor frecuencia en la base de datos y posteriormente, la confianza permite identificar qué reglas son válidas a partir de los conjuntos de elementos obtenidos anteriormente.

Las reglas de asociación deben satisfacer las especificaciones del usuario en cuanto a valores mínimos de soporte y de confianza, que en lo adelante denominaremos Minsop y Minconf respectivamente. No obstante, aún controlando los parámetros de soporte y confianza, en el conjunto de reglas resultantes, se pueden presentar reglas que pudieran estar contenidas en otras, lo cual redundaría acerca de estas relaciones.

4.2 Trabajo experimental

Para analizar el proceso de generación de reglas, considerando el tratamiento de la redundancia, se realizó una experimentación para evaluar como los algoritmos efectuan las tareas para alcanzar los resultados, esto es, la lista de reglas de asociación. El proceso de experimentación se dividió en dos tareas, mismas que se muestran a continuación:

- Obtención de itemsets frecuentes
- Generación de reglas de asociación

En ambas tareas, se utilizaron 19 bases de datos del repositorio UCI (Lichman, 2013), considerando como criterio para su elección, el que contaran con características diferentes en número de instancias, atributos, clases e ítems (cantidad de los distintos valores de los atributos), lo cual se describe en la Tabla 4.1.

Tabla 4.1 Descripción de las bases de datos utilizadas

Base de datos	Instancias	Atributos	Clases	Items
Chees	3196	37	2	74
Primary Tumor	339	18	22	59
Mushroom	124	23	7	28
Diabetes	768	8	2	29
Spect	187	23	2	46
Ballons	20	4	2	10
Breast_Cancer	150	17	2	53
Car	1728	6	4	25
Hayes_Roth	28	4	3	19
Weather	14	5	2	12
Nursery	12960	8	5	32
Titanic	2201	4	2	10
Tic-tac-toe	958	18	2	41
Solar	1066	13	3	28
Shuttle	15	6	2	16
Led7	200	7	10	38
Vote	435	16	2	24
Zoo	101	18	7	37
Dermatology	24	4	3	12

En cuanto a los métodos utilizados en la experimentación, la primera tarea se realizó con seis de los métodos descritos en la sección anterior, siendo estos: Apriori, Eclat, Apriori TID, FP-Growth, Close y Charm. Se excluyeron Top-K Rules y TNR debido a que los mismos no obtienen itemsets frecuentes para la generación de reglas. No obstante, en la segunda tarea fueron incluidos todos los métodos.

La ejecución de los métodos se realizó en un procesador Core i7 de 4 GB de memoria RAM, haciendo uso de la herramienta SPMF (Fournie, Gomariz, & Soltani, 2014), debido a que contiene todos los algoritmos que se aplicaron.

A continuación, se detallan las dos tareas realizadas en el trabajo de experimentación.

4.2.4.1 Obtención de itemsets frecuentes

El objetivo de esta tarea es analizar la cantidad de itemset frecuentes que generan los métodos considerados en las pruebas realizadas, así como corroborar el comportamiento de aquellos que trabajan con FCI.

La cantidad de itemsets frecuentes identificados por un método depende directamente del valor de soporte mínimo definido por el usuario, pues la frecuencia de aparición de un conjunto de itemsets debe ser mayor o igual a este valor. De ahí, que la frecuencia con que se presenta la mayoría de los patrones que contienen dos conjuntos de datos diferentes pueda ser distinta, lo cual implica que el valor de soporte para generar los itemsets frecuentes en cada uno de ellos sea diferente también. Debido a esto, aparecen valores de soporte diferentes aplicados a los conjuntos de datos en la etapa de experimentación.

Para poder analizar el comportamiento de los métodos con relación a los itemsets frecuentes obtenidos en correspondencia con el valor de soporte, se utilizaron dos valores de soporte mínimo por cada conjunto de datos, realizando 38 pruebas por método para un total de 228 ejecuciones. En la Tabla 4.2 se resume la información de los experimentos realizados.

De acuerdo a la descripción de cada método en la sección de introducción, los algoritmos Charm y Close a diferencia de los restantes manejan FCI, luego deben

generar menor cantidad de itemsets. En la Tabla 4.2, la cantidad de itemsets frecuentes obtenidos se presenta en dos columnas por haberse obtenido resultados idénticos por un lado con los métodos Charm y Close y por otro, con los métodos restantes.

Tabla 4.2 Datos de las pruebas de obtención de itemsets frecuentes

Num.	Base de datos	Soporte	Itemsets frecuentes	
			Apriori, Eclat, Apriori TID, FP-Growth	Charm y Close
1	Chees	0.98	23	21
2		0.88	1195	922
3	Primary Tumor	0.49	620	619
4		0.39	1759	1718
5	Mushroom	0.97	7	4
6		0.87	15	7
7	Diabetes	0.22	156	155
8		0.12	610	579
9	Spect	0.46	429	428
10		0.36	2594	2552
11	Ballons	0.40	16	10
12		0.30	28	22
13	Breast_Cancer	0.19	196	193
14		0.09	853	770
15	Car	0.33	12	10
16		0.23	84	71
17	HayesRoth	0.50	3	2
18		0.40	25	13
19	Weather	0.20	42	34
20		0.10	104	60
21	Nursery	0.33	16	14
22		0.23	63	57
23	Titanic	0.39	11	9
24		0.29	18	14
25	Tic-tac-toe	0.11	394	392
26		0.01	19849	17685
27	Solar	0.99	3	2
28		0.89	31	16
29	Shuttle	0.52	7	4
30		0.42	11	5
31	Led7	0.12	350	343
32		0.02	2577	1246
33	Vote	0.50	17	16
34		0.40	118	111
35	Zoo	0.70	23	22
36		0.60	79	50
37	Dermatology	0.90	7	6
38		0.80	54	31

El procedimiento seguido para la experimentación fue el siguiente:

1. Se toma valor de Soporte = 1
2. Se verifica si los métodos dan como resultado alguna cantidad de itemsets frecuentes diferentes para los dos grupos de métodos bajo estudio.
3. Si es positivo, Soporte 1 = Soporte, se salta a Paso 5.
4. Si es negativo Soporte 1 = Soporte 1 – 0.01, se salta a Paso 2.
5. Soporte 2 = Soporte 1 – 0.1

Considerando como ejemplo la base de datos Chess: De inicio, al soporte se le asignó un valor de 1 con el cual, los métodos no arrojaron ningún itemset frecuente, posteriormente se decrementó el soporte en 0.01 (0.99) y ambos grupos de métodos generaron la misma cantidad de itemsets frecuentes (11), al decrementar de nuevo en 0.01 (0.98) los dos grupos generaron una cantidad distinta de estos, los métodos que no consideran FCI generaron 23 y los que si lo consideran, obtuvieron 21. Este valor de soporte (0.98) se tomó como el primer valor de las pruebas con la base de datos Chess, tal como se muestra en la Tabla 4.2. Para obtener el segundo valor, se decrementó el primero en 0.1 (0.88), con ello la diferencia entre los dos grupos incrementó de manera considerable, obteniendo el primer grupo 1195 itemsets y el segundo 922, estos resultados se pueden constatar en la Tabla 4.2.

Después de realizar un análisis de los resultados alcanzados en la experimentación, se pudo constatar que los métodos que manejan FCI (Close y Charm) son los que se desempeñan mejor, lo cual se pudo comprobar a partir de la cantidad de itemsets frecuentes obtenidos por estos algoritmos en las 38 pruebas realizadas. En la Tabla 4.2 se puede observar que en todos los casos, los métodos Close y Charm obtuvieron menor número de itemsets frecuentes, lo cual corrobora que estos métodos trabajan la redundancia de los itemsets frecuentes.

Para examinar con mayor detalle cómo es que Close y Charm obtuvieron menor cantidad de reglas, se presenta un caso de la experimentación. Para ello, se eligió un conjunto de datos con pocos itemsets frecuentes, siendo ésta la base de datos Mushrom

con el soporte que obtuvo más itemsets (0.87). En este caso, los algoritmos que no consideran FCI (Apriori, Apriori TID, Eclat y FP-Growth) obtuvieron 15 itemsets frecuentes, mismos que se muestran en la Tabla 4.3, mientras que los algoritmos Close y Charm eliminaron de estos, ocho itemsets redundantes que son señalados en la misma tabla con una flecha a su izquierda.

Tabla 4.3 Itemsets obtenidos por métodos que tratan o no FCI

# Itemset	Itemsets frecuentes
1	→ gill-att='f' #SUP: 791
2	class='p' #SUP: 812
3	→ veil-color='w' #SUP: 792
4	→ ring-number='o' #SUP: 748
5	gill-att='f' class='p' #SUP: 791
6	→ gill-att='f' veil-color='w' #SUP: 790
7	→ gill-att='f' ring-number='o' #SUP: 729
8	class='p' veil-color='w' #SUP: 792
9	class='p' ring-number='o' #SUP: 748
10	→ veil-color='w' ring-number='o' #SUP: 728
11	gill-att='f' class='p' veil-color='w' #SUP: 790
12	gill-att='f' class='p' ring-number='o' #SUP: 729
13	→ gill-att='f' veil-color='w' ring-number='o' #SUP: 728
14	→ class='p' veil-color='w' ring-number='o' #SUP: 728
15	gill-att='f' class='p' veil-color='w' ring-number='o' #SUP : 728

Luego, los itemsets generados por Close y Charm son sólo aquellos siete restantes. A continuación, se describe cómo dichos métodos eliminan los itemset frecuentes redundantes considerando el caso del itemset #1.

El itemset 1 es considerado redundante respecto al itemset 5 de acuerdo a los criterios manejados por Close y Charm. Por un lado, el itemset 1 (gill-att='f') es subconjunto del itemset 5 (gill-att='f' class='p') y por otro, ambos itemsets tienen el mismo soporte (#SUP: 791). Los siete itemsets restantes se eliminaron mediante la aplicación de estos mismos criterios.

4.2.4.2 Generación de reglas de asociación

Esta tarea se realiza con el objetivo de analizar la cantidad de reglas generadas por un conjunto de métodos, algunos de los cuales consideran el tratamiento de redundancia y otros no, haciendo énfasis en los criterios utilizados por los primeros.

Para esta tarea, se agregaron los algoritmos Top-K Rules y TNR a los seis utilizados en la primera tarea, sumando un total de ocho métodos utilizados para comparar el comportamiento de los mismos en relación a la redundancia.

En esta parte de la experimentación, primero se procesaron las bases de datos de pruebas con los algoritmos que no emplean ninguna estrategia interna para detectar redundancia: Apriori, Apriori TID, Eclat, FP-Growth y Top-K Rules. Posteriormente, a los resultados obtenidos se les aplicó el método de post-procesamiento, el cual se expuso en la sección 4.2. A continuación, se aplicaron a las mismas poblaciones, los métodos que manejan redundancia mediante la obtención de FCI (Close y Charm) y finalmente, se aplicó el algoritmo TNR que maneja un criterio diferente a los anteriores para tratar la redundancia. Todo este proceso para poder contrastar los resultados alcanzados en cada alternativa.

Para cada aplicación de los métodos, se utilizó un valor de soporte y dos de confianza con la finalidad de evaluar el comportamiento de los métodos cuando crece la cantidad de reglas, esto es, al disminuir la confianza.

En lo que respecta al soporte, se utilizó el segundo valor que se aplicó a cada conjunto de datos de la primera tarea pues con dicho valor el tratamiento de redundancia se hizo más notorio. Para los conjuntos de datos Car, Hayes Roth, Nursery y Zoo, los valores aplicados no permitían obtener una cantidad diferente de reglas entre los grupos de métodos con y sin tratamiento de redundancia, por ello, en estos casos se redujo aún más el valor de soporte hasta obtener cantidades diferentes de reglas entre ambos grupos de métodos.

En cuanto a los dos valores de confianza, se utilizó el mismo criterio con el que se obtuvieron los valores de soporte de la tarea anterior. Para obtener el primer valor, se inició con un valor de 1 y se decrementó en 0.01 las veces que fue necesario hasta generar cantidades distintas de reglas entre los conjuntos de métodos. Referente al segundo valor, se obtuvo decrementando el primero en 0.1.

Bajo las condiciones con las que se realizó la experimentación, esto es, dos valores de confianza para cada uno de los 19 conjuntos de datos, se tenían 38 casos diferentes, a

los cuales se aplicaron ocho métodos de generación de reglas y uno de post-procesamiento, efectuándose así un total de 342 ejecuciones en esta tarea. En la Tabla 4.4, se resume las ejecuciones realizadas.

Tabla 4.4 Resultados de la ejecución de los métodos de generación de reglas

# Prueba	Base de datos	Soporte	Confianza	Reglas de asociación generadas			
				Apriori, Eclat, Apriori TID, FP-Growth y Top-K Rules	Método de Postprocesamiento	Close y Charm	TNR
1	Chees	0.88	1	295	290	273	251
2			0.90	25196	25134	21047	18888
3	Primary Tumor	0.39	0.99	297	296	294	291
4			0.89	4746	4720	4696	4647
5	Mushrom	0.87	0.90	47	45	30	19
6			0.80	50	48	32	21
7	Diabetes	0.12	0.97	282	281	277	272
8			0.87	738	735	707	678
9	Spect	0.36	0.97	657	655	647	646
10			0.87	6653	6605	6489	6468
11	Ballons	0.30	0.66	16	15	10	8
12			0.56	28	27	22	20
13	Breast Cancer	0.09	1	97	92	83	72
14			0.90	648	639	550	505
15	Car	0.11	0.33	137	132	115	103
16			0.23	150	143	128	116
17	HayesRoth	0.28	0.57	18	17	16	14
18			0.47	19	18	17	15
19	Weather	0.10	1	58	56	44	33
20			0.85	59	53	45	34
21	Nursery	0.16	0.33	74	72	66	64
22			0.23	74	72	66	64
23	Titanic	0.29	0.51	33	32	28	23
24			0.41	46	44	37	32
25	Tic-tac-toe	0.01	1	2204	2169	2164	2124
26			0.90	3654	3606	3510	3422
27	Solar	0.89	0.99	37	36	30	23
28			0.89	180	175	130	80
29	Shuttle	0.42	0.66	15	14	10	8
30			0.56	20	19	13	10
31	Led7	0.02	1	3739	3536	1331	552
32			0.90	8683	8578	3358	1686
33	Vote	0.40	0.98	37	36	36	35
34			0.88	232	227	212	208
35	Zoo	0.40	1	635	620	229	101
36			0.90	1563	1529	828	526
37	Dermatology	0.80	0.90	170	166	89	81
38			0.80	194	188	110	95

Al igual que en la primera tarea, algunos de los algoritmos generaron la misma cantidad de reglas, por ello se agrupan en la tabla. Se muestran cuatro columnas con las cantidades de reglas generadas por cada grupo de métodos: la primera integrada por Apriori, Apriori TID, Eclat, FP-Growth y Top-K Rules, mismos que no incluyen

internamente una estrategia para manejar la redundancia; la segunda que resulta al aplicar un método de postprocesamiento a los métodos de la primera columna; la tercera constituida por Close y Charm que aplican manejo de redundancia y la cuarta donde aparece TNR, que también trata la redundancia, pero con criterios diferentes a las anteriores.

Después de la obtención de las reglas mediante la aplicación de los diferentes métodos, se compararon los resultados obtenidos por los grupos conformados, considerando los siguientes tres casos:

1. Análisis de la cantidad de reglas generadas por los métodos que no incluyen internamente una estrategia para manejar la redundancia con relación a la que resulta de aplicar un método de postprocesamiento (1ra columna & 2da columna).
2. Análisis de la cantidad de reglas generadas por los métodos que no incluyen internamente una estrategia para manejar la redundancia con relación a aquellos que aplican manejo de redundancia, en este caso; Close y Charm (1ra columna & 3ra columna).
3. Análisis de la cantidad de reglas generadas por los métodos que aplican tratamiento de redundancia pero que obtienen diferentes cantidades de reglas, esto es, Close y Charm, y TNR (3ra columna & 4ta columna).

Con relación al primer análisis, los resultados muestran que la cantidad de reglas generadas después de haber aplicado el método de postprocesamiento a los algoritmos que no trataban la redundancia, es ligeramente menor que la que obtuvieron estos métodos por si solos. Ello demuestra que el método de postprocesamiento no resulta muy eficiente para resolver el problema de redundancia y esto se debe a que el mismo utiliza un criterio de eliminación de reglas, que considera solamente aquellas que tienen un ítem en el sucedente, lo cual ocurre en la minoría de las reglas. De lo anterior resulta que, la complejidad en la interpretación de las reglas que se generan al aplicar este método no disminuya de manera considerable.

Respecto al segundo análisis, los resultados muestran que la cantidad de reglas generadas por los métodos Close y Charm (3ra columna) fue menor que la que generaron los algoritmos que no manejan redundancia (1ra columna).

El buen desempeño que mostraron estos dos métodos frente al problema de redundancia se debe a que las reglas se obtienen a partir de los FCI, los cuales se obtienen aplicando un criterio que permite eliminar todo aquel itemset frecuente I_1 que tiene menor o igual soporte que I_2, evitando así generar todas aquellas reglas que provienen de los itemsets frecuentes que no son cerrados.

El buen desempeño de los métodos Close y Charm, se refleja no sólo comparado con los métodos que no manejan redundancia (1ra columna), sino que inclusive, después de haber aplicado un método externo de postprocesamiento a estos últimos (2da columna) aún sigue siendo mayor la cantidad de reglas generadas por los mismos. Este comportamiento se debe a que el método de postprocesamiento sólo considera para la posible eliminación de reglas aquellas que tienen un solo ítem en el sucedente, mientras que Close y Charm consideran todas las reglas generadas.

Finalmente en el tercer análisis, comparando los resultados obtenidos por TNR (4ta. columna) con los que obtuvieron Close y Charm (3ra columna), se puede observar que en todos los casos TNR logró eliminar mayor cantidad de reglas. Este comportamiento se debe a que TNR, además de aplicar criterios equivalentes a los que utilizan Close y Charm, hace una inspección exhaustiva del contenido de cada regla, para identificar aquellas donde coinciden los elementos del antecedente pero también los del sucedente, sin embargo, la diferencia entre estas reglas está en el orden en que se presentan. Una vez detectados estos casos, se eliminará aquella regla que presenta menor o igual soporte y confianza. Este criterio extra es el que hace que TNR elimine aún más reglas que Close y Charm y por tanto, brinde un mejor desempeño.

Para ilustrar el comportamiento anterior, se retoma el mismo caso que se utilizó en la tarea de obtención de itemsets frecuentes (base de datos Mushrom con soporte = 0.87) con el segundo valor de confianza (0.80). Bajo estas condiciones, Close y Charm obtuvieron las 32 reglas que se muestran en la Tabla 4.5. Por su parte, el método TNR

eliminó 11 de las 32 reglas, las cuales aparecen en la misma tabla marcadas en la parte izquierda con una flecha.

TNR eliminó estas 11 reglas por ser redundantes dentro de las 32 generadas. Analizando la regla 7, la cual fue eliminada por considerarse redundante respecto a la regla 10, se puede observar según los criterios definidos por TNR que:

- El antecedente está conformado por los mismos items en ambas reglas, sólo que en orden distinto. Siendo el sucedente el mismo.

  R7. class='p' veil-color='w' ==> gill-att='f'
  R10. veil-color='w' class='p' ==> gill-att='f'

- Las dos reglas tienen el mismo soporte y confianza (#SUP: 790 #CONF: 0.99).

Bajo estos mismos criterios se eliminaron las 10 reglas restantes.

Tabla 4.5 Reglas generadas por Close, Charm y TNR

# Regla	Reglas
1	class='p' ==> gill-att='f' #SUP: 791 #CONF: 0.97
2	gill-att='f' ==> class='p' #SUP: 791 #CONF: 1.0
3	veil-color='w' ==> class='p' #SUP: 792 #CONF: 1.0
4	class='p' ==> veil-color='w' #SUP: 792 #CONF: 0.97
5	ring-number='o' ==> class='p' #SUP: 748 #CONF: 1.0
6	class='p' ==> ring-number='o' #SUP: 748 #CONF: 0.92
7	→ class='p' veil-color='w' ==> gill-att='f' #SUP: 790 #CONF: 0.99
8	gill-att='f' veil-color='w' ==> class='p' #SUP: 790 #CONF: 1.0
9	→ gill-att='f' class='p' ==> veil-color='w' #SUP: 790 #CONF: 0.99
10	veil-color='w' class='p' ==> gill-att='f' #SUP: 790 #CONF: 0.99
11	class='p' ==> gill-att='f' veil-color='w' #SUP: 790 #CONF: 0.97
12	gill-att='f' ==> ring-number='o' veil-color='w' #SUP: 790 #CONF: 0.99
13	→ class='p' ring-number='o' ==> gill-att='f' #SUP: 729 #CONF: 0.97
14	gill-att='f' ring-number='o' ==> class='p' #SUP: 729 #CONF: 1.0
15	gill-att='f' class='p' ==> ring-number='o' #SUP: 729 #CONF: 0.92
16	ring-number='o' ==> gill-att='f' class='p' #SUP: 729 #CONF: 0.97
17	class='p' ==> gill-att='f' ring-number='o' #SUP: 729 #CONF: 0.89
18	gill-att='f' ==> class='p' ring-number='o' #SUP: 729 #CONF: 0.92
19	→ class='p' veil-color='w' ring-number='o' ==> gill-att='f' #SUP: 728 #CONF: 1.0
20	→ gill-att='f' veil-color='w' ring-number='o' ==> class='p' #SUP: 728 #CONF: 1.0
21	→ gill-att='f' class='p' ring-number='o' ==> veil-color='w' #SUP: 728 #CONF: 0.99
22	→ gill-att='f' class='p' veil-color='w' ==> ring-number='o' #SUP: 728 #CONF: 0.92
23	veil-color='w' ring-number='o' ==> gill-att='f' class='p' #SUP: 728 #CONF: 1.0
24	→ class='p' ring-number='o' ==> gill-att='f' veil-color='w' #SUP: 728 #CONF: 0.97
25	→ class='p' veil-color='w' ==> gill-att='f' ring-number='o' #SUP: 728 #CONF: 0.91
26	gill-att='f' ring-number='o' ==> class='p' veil-color='w' #SUP: 728 #CONF: 0.99
27	gill-att='f' veil-color='w' ==> class='p' ring-number='o' #SUP: 728 #CONF: 0.92
28	→ gill-att='f' class='p' ==> veil-color='w' ring-number='o' #SUP: 728 #CONF: 0.92
29	→ ring-number='o' ==> gill-att='f' class='p' veil-color='w' #SUP: 728 #CONF: 0.97
30	veil-color='w' ==> gill-att='f' class='p' ring-number='o' #SUP: 728 #CONF: 0.91
31	class='p' ==> gill-att='f' veil-color='w' ring-number='o' #SUP: 728 #CONF: 0.89
32	gill-att='f' ==> class='p' veil-color='w' ring-number='o' #SUP: 728 #CONF: 0.92

De acuerdo a los resultados de la experimentación realizada, se pudo detectar que algunos métodos incluyen sus propios criterios para eliminar reglas innecesarias y de estos criterios depende con que efectividad se resuelve el problema. Los algoritmos que incluyen mayor cantidad de restricciones en su proceso de generación de reglas mostraron mejor desempeño.

La inclusión de la técnica de obtención de FCI le permitió a Charm y Close eliminar reglas innecesarias que métodos como Apriori, Apriori TID, Eclat, FP-Growth y Top-K Rules no lograron hacer mientras que TNR, brindó el mejor desempeño entre los ocho algoritmos debido a que además de aplicar las comprobaciones de Charm y Close, añade otros criterios que permiten supervisar la redundancia entre los ítems que componen el antecedente y sucedente de las reglas.

Debido a lo anterior, de manera general se recomienda utilizar el algoritmo TNR, el único inconveniente que muestra éste método es que cuando procesa grandes cantidades de reglas, demanda mayor capacidad de memoria que los siete algoritmos restantes.

Capítulo 5
Modelación de reglas de asociación en esquema ontológico

Una de las fases que contempla el modelo híbrido propuesto en este trabajo es la modelación del conocimiento extraído de las reglas de asociación en una Base de Conocimientos. En este capítulo, se expone el método propuesto para efectuar la modelación de las reglas generadas en un esquema ontológico, su validación y las conclusiones obtenidas.

5.1 Modelación de reglas de asociación en esquema ontológico

Tomando en consideración que la finalidad de realizar el modelado es que en la predicción se puedan detectar comportamientos que influyen para pertenecer a determinada clase, se hace uso de un tipo especial de reglas denominadas *Reglas de Asociación de Clase* (RAC), las cuales se caracterizan por contener un solo ítem en el sucedente correspondiente a la categoría de la clase, el resultado de la aplicación de este paso será un conjunto de reglas compuestas por uno o más par atributo-valor en el antecedente y una categoría de la variable *Clase* en el sucedente.

A diferencia de los trabajos que se han realizado sobre representación de reglas de asociación en ontologías, expuestos en el capítulo dos (sección 2.5), en el método propuesto se crea la Base de Conocimientos a partir de las RAC generadas previamente. Por otra parte, además de modelar los conceptos, representa otros componentes de una ontología, las propiedades, que constituyen las características de las clases y los axiomas, que son aquellos que permiten definir las reglas de inferencia consideradas ciertas para el dominio.

Debido a que el modelo propuesto tiene como tarea efectuar la predicción, los componentes que se modelan en la ontología son: conceptos o clases, propiedades y axiomas. Los pasos para modelar el conocimiento de un conjunto de RAC a un esquema ontológico son los siguientes:

- Paso 1: Generar las RAC
- Paso 2: Identificar en las RAC las categorías de clase y atributos
- Paso 3: Modelar conceptos y propiedades de datos
- Paso 4: Modelar axiomas
- Paso 5: Crear la ontología

Para generar el conjunto de reglas de asociación, se aplica el método TNR a la base de datos en tratamiento. Se seleccionó este método debido a que como se expone en el capítulo cuatro, después de realizar un análisis de algoritmos de este tipo, resultó ser el que brinda los mejores resultados respecto a la eliminación de reglas redundantes. Sin embargo, es posible aplicar algún otro algoritmo de generación de RAC.

5.1.1 Identificar las categorías de clase y atributos

Para identificar las categorías de clase, se recorre cada una de las reglas generadas en el paso anterior y se busca en el sucedente de la regla la categoría que aparece en ella. Al final, se recuperan las categorías del atributo *Clase* del conjunto de datos.

Por otro lado, para identificar los atributos, se inspecciona el antecedente de cada una de las RAC, donde por cada ítem (par atributo-valor) se obtiene el nombre del atributo correspondiente. Como resultado de esta parte del proceso, se obtienen aquellos atributos que aparecen en los patrones más frecuentes de los datos bajo estudio, los cuales no necesariamente conforman el universo de los atributos.

5.1.2 Modelar conceptos y propiedades de datos

Para modelar una ontología, es necesario considerar que la misma es un modelo de conocimiento que está integrada por conceptos o clases, relaciones, propiedades, instancias, funciones y axiomas.

En este paso, se modelan conceptos y propiedades utilizando los elementos de las RAC que se recuperaron en el paso anterior.

En el esquema ontológico cada concepto se declara de forma explícita individualmente, luego por cada una de las categorías del atributo *Clase* se crea un concepto. El conjunto de conceptos modelado está directamente relacionado con la estructura de grupos del conjunto de datos original.

A continuación, se modelan las propiedades de los datos, que se corresponden con los atributos identificados en las RAC y que representan las características del conjunto de datos que están incluidos en los patrones más frecuentes detectados por estas reglas.

5.1.3 Modelar axiomas

Con la finalidad de integrar en la ontología el componente que permite ejecutar el razonamiento, se hace uso de los axiomas. Este elemento es de gran utilidad para detectar aquellas instancias que tienen comportamientos particulares en los datos y que pudieran bajo determinadas condiciones cambiar de concepto (*Clase* en el conjunto de datos) y por consiguiente, el resultado de una predicción.

Los axiomas se modelan a partir de los conceptos y propiedades previamente creados y se relacionan con una clase de equivalencia, lo cual permitirá inferir acerca del concepto incluido en el mismo.

Para modelar el axioma, se consideran todas las RAC cuyo sucedente coincide con el concepto que se incluye en éste. Recuperando de ellas todos los ítems que aparezcan en el antecedente. En particular para este componente de la ontología, se incluye el valor de los atributos que hacen posible la ocurrencia del concepto.

5.1.4 Crear la ontología

Finalmente, una vez que han sido modelados los componentes de la ontología a partir de las RAC, se crea la estructura de la Base de Conocimientos bajo este esquema, la cual se relaciona con la representación del Leguaje OWL. Se eligió éste, debido a que es uno de los lenguajes de representación ontológica más robusto, por poseer gran variedad de componentes y funcionalidades para su esquema.

5.1.5 Representación formal

Sea *R* un conjunto de *i* reglas de asociación de clase, cada una de las cuales está integrada por un antecedente R_A y un sucedente R_s, cada antecedente está constituido por *j* ítems (*I*) y cada uno de los ítems se conforma de un atributo par atributo-valor (A-V) y sea *O* una ontología compuesta por C_k Conceptos, P_k Propiedades y AX_k Axiomas.

La representación de categorías en conceptos viene dada por:

$$\forall R_i \in R : C_k := R_{S_i} \leftrightarrow R_{S_i} \notin C$$

La representación de atributos en propiedades de datos por:

$$\forall R_i \in R,\ \forall A_{ij} \in R_{A_i} : P_k := A_{ij} \leftrightarrow A_{ij} \notin$$

Finalmente, la representación de los axiomas está dada por:

$$\forall R_i \in R,\ \forall I_{ij} \in R_{A_i} : AX_i +:= I_{ij}$$

5.2 Trabajo experimental

Se realizaron pruebas del proceso de modelación con 12 conjuntos de datos del repositorio UCI (Lichman, 2013), cuyas características se muestran en la Tabla 5.1, las reglas se generaron utilizando el algoritmo *TNR*. La representación en lenguaje OWL se realizó con un Framework desarrollado en Java y las pruebas de inferencia de los axiomas se hicieron con los razonadores *Pellet* y *Hermit* [7].

Tabla 5.1 Bases de datos de prueba

Base de datos	Atributos	Clases
Diabetes	8	2
Spect	23	2
Ballons	4	2
Breast_Cancer	17	2
Car	6	4
Hayes_Roth	4	3
Zoo	18	7
Tic-tac-toe	18	2
Solar	13	3
Shuttle	6	2
Vote	16	2
Dermatology	4	3

Para ejemplificar el proceso de modelación propuesto, se utiliza el conjunto de datos Breast Cancer. En el Cuadro 5.1, se muestra el conjunto de RAC generadas después de aplicar el método TNR.

Cuadro 5.1 Reglas generadas del conjunto Breast Cancer

1.- deg-mali=3 menopaus=ge40 ==> class=no_rec
2.- tumor_tm=30-34 ==> class=rec
3.- irradiat=no deg-mali=3 menopaus=ge40 ==> class=no_rec
5.- irradiat=no menopaus=ge40 ==> class=no_rec
5.- menopaus=ge40 ==> class=no_rec
6.- irradiat=no tumor_tm=30-34 ==> class=rec
7.- node_cap=yes menopaus=ge40 ==> class=no_rec
8.- irradiat=no nodes_pa=0-2 node_cap=yes ==> class=rec

La modelación del conjunto de estas reglas se resume en la Figura 5.1.

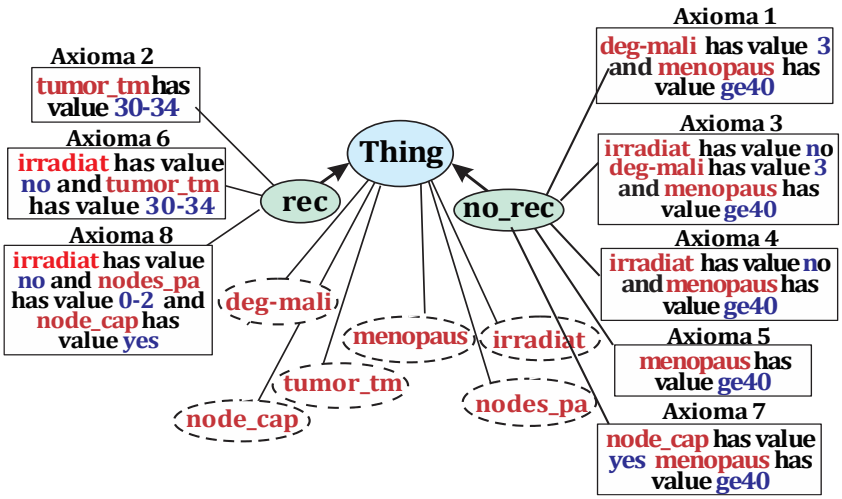


Figura 5.1 Modelación ontológica de las RAC de Breast Cancer

En este conjunto de datos las categorías de la clase son: *rec y no_rec*, mismas que son representadas en la ontología como conceptos o clases. Los seis atributos involucrados en los antecedentes de las reglas se modelan como propiedades de datos y finalmente, se definen los axiomas que se relacionan con cada categoría, tres relacionados con el concepto *rec* y cinco con *no_rec*. Después de efectuar la modelación se llevó a cabo la representación en lenguaje OWL, esta representación se validó con la herramienta Protegé [7]. En la Figura 5.2, se muestran los conceptos o clases creadas y los axiomas correspondientes a la clase *Rec*, ambos componentes modelados a partir de las RAC.

Con la ontología creada, el razonador podrá predecir la clase de nuevas instancias, a partir del valor de las propiedades de los datos e inclusive, brindar recomendaciones en aquellos casos que esta predicción pudiese cambiar.

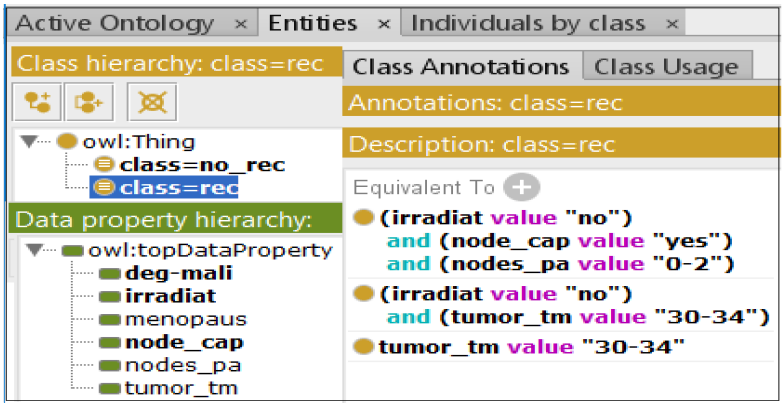


Figura 5.2 Ontología de Breast Cancer en Protegé

En lo que respecta al razonamiento, en la misma herramienta Protegé se comprobó el correcto funcionamiento de los axiomas, aplicándolos a nuevas instancias y haciendo uso de los razonadores Hermit y Pellet que se encuentran instalados en la propia herramienta.

La modelación del conocimiento extraído con técnicas de Minería de Datos a través de una estructura ontológica, proporcionará información útil adicional para la toma de decisiones.

El uso de un esquema ontológico, mediante sus componentes principales, como axiomas, conceptos y propiedades, permitió la integración del conocimiento contenido en las RAC en una Base de Conocimientos. Este esquema fue elegido debido a la facilidad para la modelación del conocimiento en forma de RAC.

Todas las componentes de la ontología, modeladas a partir de las reglas de asociación de clases permiten abordar la predicción con técnicas de Inteligencia Artificial y en especial los axiomas, debido a que son los elementos que permiten inferir conocimiento que no está indicado explícitamente en la taxonomía de conceptos.

La representación abordada en este trabajo permite distinguir aquellos casos que bajo determinadas condiciones pueden variar su comportamiento (clase) y que pueden ser de gran utilidad en la toma de decisiones.

Capítulo 6
Selección del clasificador

Dentro de las técnicas predictivas, la clasificación es una tarea muy socorrida en cualquier área del conocimiento por su capacidad de identificar automáticamente para una nueva instancia, la clase a la cual está asociada, utilizando para ello el conocimiento que relaciona las características de las instancias con sus respectivas clases (Hernández, Ramírez, & Ferri, 2013).

6.1 Introducción

Para llevar a cabo la tarea de clasificación se han desarrollado diversos algoritmos, los cuales ofrecen soluciones bajo diferentes enfoques como son: árboles de decisión, basados en vecindad, probabilísticos, redes neuronales, entre otros. Sin embargo, la diversidad de clasificadores y la versatilidad de las poblaciones que se estudian en el ámbito real, hace compleja la tarea de seleccionar un algoritmo en específico que se ejecute con mayor eficacia.

En investigaciones realizadas, con la intención de identificar relaciones entre clasificadores y conjuntos de datos, se ha encontrado que algunos algoritmos funcionan mejor que otros para ciertos conjuntos de datos.

Como se mencionó anteriormente, uno de los objetivos del presente trabajo tiene la finalidad de estudiar diversos algoritmos y poblaciones de datos para analizar su comportamiento y encontrar relaciones entre ambos, con el propósito de proponer ciertos patrones que aunque no sean absolutas, sirvan de apoyo a momento realizar una selección del clasificador lo más acertada posible en función de las características de la base de datos que se analiza. Para este estudio se ocupan los algoritmos de clasificación más utilizados en la literatura, mismos que fueron probados con diversas bases de datos de características diferentes, considerando la cantidad de instancias, número de clases, número de atributos, tipo de datos, si es o no balanceada, entre otras.

6.2 Trabajo experimental

Con la finalidad de encontrar las particularidades que influyen en la eficacia de los algoritmos de clasificación, se llevó a cabo la ejecución de cuatro algoritmos de este tipo sobre un conjunto de 38 bases de datos y se realizó un análisis de los resultados obtenidos, identificándose comportamientos particulares de desempeño relacionados con las poblaciones en estudio mediante obtención de reglas de clasificación. Posteriormente, se validaron los patrones encontrados con ocho bases de datos distintas.

6.2.1 Aspectos a considerar en el estudio

Para elegir los conjuntos de datos a utilizar en este estudio se consideraron características básicas de las bases de datos como: cantidad de instancias, número de atributos, número de clases y tipos de datos. Adicionalmente, se consideró que existían otras características de las bases de datos que también podrían influir en la elección de un algoritmo de clasificación. Estas características son:

- *Estructura*: Propiedad de la base de datos que permite especificar si tiene una forma vertical, horizontal o mixta. *Vertical* se refiere a que tiene muchas instancias y pocos atributos, *Horizontal* indica lo contrario. Por otra parte, cuando tiene pocos atributos y pocas instancias o muchos atributos y muchas instancias se le denomina *Mixta*.
- *Multivaluada*: Permite especificar si la mayoría de los atributos nominales tiene más de dos valores.
- *Completa*: Permite especificar si una base de datos tiene o no datos faltantes.
- *Balanceada*: Especifica si el conjunto de datos contiene el mismo número de instancias en todas las clases.
- *Cantidad de atributos numéricos o nominales*: Se detalla la cantidad de variables numéricas o nominales.

6.2.2 Bases de datos y clasificadores utilizados

Se utilizaron 30 bases de datos para la etapa de experimentación y 8 para la de validación, obtenidas de los repositorios de UCI Machine Learning (Lichman, 2013),

portal de WEKA (Hall, Frank, Holmes, & Pfahringer, 2009) y portal de Promise (Zwirk & Wigury, 2013). Los algoritmos de clasificación elegidos para este estudio se expusieron en el capítulo uno y son: Árboles de Decisión (C4.5), Naive Bayes, Perceptrón Multicapa y KNN, por ser ellos de los más utilizados en la literatura. La Tabla 6.1 muestra los conjuntos de datos utilizados en la experimentación.

Tabla 6.1 Bases de datos para pruebas

Base de datos	Tipo de datos	Atributos	Numéricos	Nominales	Instancias	Clases	Balanceada	Incompleta	Estructura
Weather_Nom	Nominal	4	0	4	14	2	No	No	Mixta
Car	Nominal	6	0	6	1728	4	No	No	Vertical
Nursery	Nominal	8	0	8	12960	5	No	No	Vertical
Primary_tumor	Nominal	17	0	17	339	22	No	No	Horizontal
Led7	Nominal	7	0	7	200	10	No	No	Vertical
lymphography	Nominal	18	0	18	148	4	No	No	Mixta
Splice	Nominal	60	0	60	3190	3	No	No	Mixta
Breast_Cancer	Nominal	9	0	9	150	2	No	No	Vertical
Spect	Nominal	23	0	23	80	2	No	No	Horizontal
Balance_Scale	Numérico	4	4	0	625	3	No	No	Vertical
Diabetes	Numérico	8	8	0	768	2	No	No	Vertical
Glass	Numérico	9	9	0	214	7	No	No	Vertical
Breast_w	Numérico	9	9	0	699	2	No	No	Vertical
Messidor_features	Numérico	19	19	0	1151	2	Si	No	Mixta
Sonar	Numérico	60	60	0	208	2	Si	No	Horizontal
Iris	Numérico	4	4	0	150	3	Si	No	Vertical
Vehicle	Numérico	18	18	0	846	4	Si	No	Mixta
Waveform_5000	Numérico	40	40	0	5000	3	Si	No	Mixta
Column3	Numérico	7	7	0	310	3	No	No	Horizontal
Ecoli	Numérico	7	7	0	336	6	No	No	Horizontal
Weather_Num	Mixto	4	2	2	14	2	No	No	Mixta
Vowel	Mixto	13	10	3	990	11	Si	No	Mixta
Zoo	Mixto	17	1	16	121	7	No	No	Horizontal
Tae	Mixto	5	3	2	151	3	Si	No	Mixta
Zoo	Mixto	17	1	16	121	7	No	No	Horizontal
Credit-g	Mixto	20	7	13	1000	2	No	No	Horizontal
Bands	Mixto	39	22	17	539	2	No	Si	Horizontal
Credit_Aproval	Mixto	15	6	9	690	2	No	Si	Vertical
Autos	Mixto	25	15	10	205	7	No	Si	Horizontal
Colic	Mixto	22	7	15	368	2	No	Si	Horizontal

Por su parte, la Tabla 6.2 describe los conjuntos de datos utilizados en la fase de validación de las recomendaciones que son emitidas en la etapa de experimentación. Se buscó que estos conjuntos cubrieran las características que identifican a un conjunto de datos (ver Sección 6.1).

Para ejecutar los algoritmos de clasificación se utilizó la herramienta Weka versión 6.6, en un procesador Core i7 de 8 GB de memoria RAM con Windows 7.

Tabla 6.2 Bases de datos para validación

Base de datos	Tipo datos	Atributos	Numéricos	Nominales	Instancias	Clases	Balanceada	Incompleta	Estructura
Vote	Nominal	16	0	16	435	2	No	Si	Horizontal
Monk	Nominal	6	0	6	124	2	Si	No	Horizontal
Letter	Numérico	16	16	0	20000	26	Si	No	Mixta
Sick	NumNom	29	7	22	3772	2	No	Si	Horizontal
Ozone	Numérico	72	72	0	2536	2	No	Si	Horizontal
Segment	Numérico	19	19	0	2310	7	Si	No	Horizontal
Page_Blocks	Numérico	10	10	0	5473	5	No	No	Vertical
Squash_Stored	NumNom	24	21	3	52	3	No	No	Horizontal

6.2.3 Resultados obtenidos

La evaluación de los resultados se hizo usando validación cruzada, obteniéndose los resultados que se muestran en la Tabla 6.3, en donde se puede observar de manera resaltada cuál de los cuatro algoritmos obtuvo un mejor desempeño para cada conjunto de datos. Resultando C4.5 mejor en tres conjuntos, Naive Bayes en 6, KNN en 10 y MLP en 11.

Utilizando estos resultados, se creó un conjunto de datos de entrenamiento con la información que se muestra en la Tabla 6.1, eliminando la columna del nombre de la base de datos y agregando en una última columna como atributo clase, el nombre del clasificador que obtuvo el mejor porcentaje de precisión, cuyo dato se muestra en la Tabla 6.3.

Tabla 6.3 Porcentajes de precisión obtenidos por método

Base de datos	C4.5	Naive Bayes	KNN	MLP
Weather_Nom	50%	57.10%	57.10%	**71.42%**
Car	92.30%	86.70%	93.50%	**99.53%**
Nursery	97%	90.30%	98.37%	**99.72%**
Primary_tumor	39.80%	**50.10%**	39.20%	38.30%
Led7	71.50%	**74%**	70%	69.50%
lymphography	77.02%	83.10%	82.40%	**84.40%**
Splice	94.35%	96.36%	74.67%	**96.55%**
Breast_Cancer	76.52%	**76.67%**	72.37%	64.68%
Spect	71.05%	**71.25%**	53.75%	63.75%
Balance_Scale	76.64%	90.40%	86.56%	**90.72%**
Diabetes	73.80%	**76.30%**	70.18%	76.39%
Glass	66.82%	48.59%	**71.96%**	67.75%
Breast_w	94.56%	96.99%	**96.85%**	96.27%
Messidor_features	64.37%	77.68%	**81.15%**	72.02%
Sonar	71.15%	67.78%	**86.53%**	82.21%
Iris	96%	96%	96.30%	**97.33%**
Vehicle	72.45%	44.79%	69.85%	**81.67%**
Waveform_5000	76.08%	80%	73.62%	**83.56%**
Column3	81.61%	83.22%	78.38%	**86.48%**
Ecoli	84.22%	86.41%	80.35%	**86.01%**
Weather_Num	64.28%	64.28%	**78.57%**	78.37%
Vowel	81.51%	63.73%	**99.29%**	92.82%
Zoo	92.07%	96.04%	**96.03%**	96.90%
Tae	59.60%	54.30%	**62.25%**	54.30%
Zoo	92.07%	96.04%	**96.03%**	96.90%
Credit-g	70.50%	**76.40%**	72.00%	71.50%
Bands	64.74%	73.28%	**78.29%**	77%
Credit_Aproval	**86.08%**	77.60%	81.10%	83.18%
Autos	**81.95%**	56.09%	76.09%	80%
Colic	**86.32%**	77.98%	81.25%	80.43%

Este conjunto de datos, se creó con el objetivo de saber si existen determinados patrones que permitan evidenciar alguna relación o dependencia entre el algoritmo de clasificación y las características del conjunto de datos. Para poder obtener dichos patrones, se aplicaron a los datos de entrenamiento los algoritmos Apriori, BFTree, JRIP rules, J48 y Ridor. A continuación se presenta el conjunto de patrones obtenidos con cada algoritmo.

Algoritmo Apriori

1. Tipo_Datos=Nominal Num_Clases=3_7 4 ==> Clase=MLP 4 conf:(1)
2. Tipo_Datos=Nominal Num_Clases=3_7 Completa=No 4 ==> Clase=MLP 4 conf:(1)
3. Tipo_Datos=Numérico Num_Clases=3_7 7 ==> Clase=MLP 6 conf:(0.86)
4. Tipo_Datos=Numérico Num_Clases=3_7 Completa=No 7 ==> Clase=MLP 6 conf:(0.86)
6. Completa=Si 5 ==> Clase=C4_5 4 conf:(0.8)
6. Tipo_Datos=NumNom Completa=No 5 ==> Clase=KNN 4 conf:(0.8)
7. Tipo_Datos=NumNom Completa=Si 5 ==> Clase= 4 conf:(0.8)
8. Balanceada=No Completa=Si 5 ==> Clase=C4_5 4 conf:(0.8)
9. Tipo_Datos=NumNom Balanceada=No Completa=Si 5 ==> Clase=C4_5 4 conf:(0.8)

Algoritmo BFTree

```
Tipo_Datos = (NumNom)
|  Completa = (No): KNN(4.0/1.0)
|  Completa != (No): C4_5(3.0/1.0)
Tipo_Datos != (NumNom)
|  Num_Clases < 2.5: NB(3.0/4.0)
|  Num_Clases >= 2.5
|  |  Num_Clases < 6.5: MLP(10.0/0.0)
|  |  Num_Clases >= 6.5: NB(2.0/1.0)
```

Algoritmo JRIP rules

```
(Completa = Si) => Clase=C4_5 (4.0/1.0)
(Tipo_Datos = NumNom) => Clase=KNN (6.0/1.0)
 => Clase=MLP (20.0/9.0)
```

Algoritmo C4.5

```
Completa = Si: C4_5 (4.0/1.0)
Completa = No
|  Tipo_Datos = Nominal
|  |  Num_Clases <= 7
|  |  |  Num_Clases <= 2: NB (3.0/1.0)
|  |  |  Num_Clases > 2: MLP (4.0)
|  |  Num_Clases > 7: NB (2.0)
|  Tipo_Datos = Numérico
|  |  Num_Clases <= 2: KNN (4.0/1.0)
|  |  Num_Clases > 2: MLP (7.0/1.0)
|  Tipo_Datos = NumNom: KNN (6.0/1.0)
```

Algoritmo Ridor

```
Clase = C4_5  (29.0/26.0)
    Except (Completa = No) => Clase = NB  (17.0/0.0) [8.0/0.0]
        Except (Num_Clases > 2.5) and (Num_Clases <= 8.5) => Clase = KNN (10.0/0.0) [3.0/0.0]
            Except (Num_Clases <= 6.5) => Clase = MLP  (7.0/0.0) [4.0/1.0]
            Except (Balanceada = Si) => Clase = KNN  (2.0/0.0) [1.0/0.0]
            Except (Tipo_Datos = Numérico) => Clase = MLP  (3.0/1.0) [2.0/1.0]
```

A partir de estos resultados, se detectaron aquellos patrones que coinciden en los resultados generados por los cinco algoritmos. Estos patrones se resumen en la Tabla 6.4.

Tabla 6.4 Patrones más frecuentes por método de clasificación

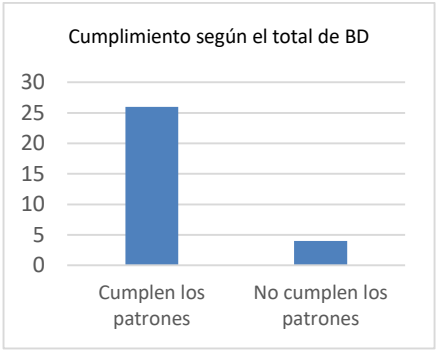
No	Patrones	Algoritmo
1	BD_Incompleta = Si	C4.5
2	((Num_Clases >= 3 and Num_Clases < 7) and (Tipo_Dato = Numérico)) or ((Num_Clases >= 3 and Num_Clases < 7) and (Tipo_Dato=Nominal))	MLP
3	((Num_Clases < 3 or Num_Clases >= 7) and Tipo_Datos = Numérico)) OR (Si BD=Mixta)	KNN
4	(Tipo_Datos = Nominal) and (Num_Clases < 3 or Num_Clases >= 7)	Naive Bayes

Como se muestra en la Tabla 6.4, las características que más influyeron en los patrones detectados fueron: número de clases, tipos de atributos y si la base de datos tiene valores faltantes o no. Por otra parte, en la Tabla 6.5 se detallan los casos donde sí y donde no se cumplió el patrón obtenido en correspondencia con el algoritmo.

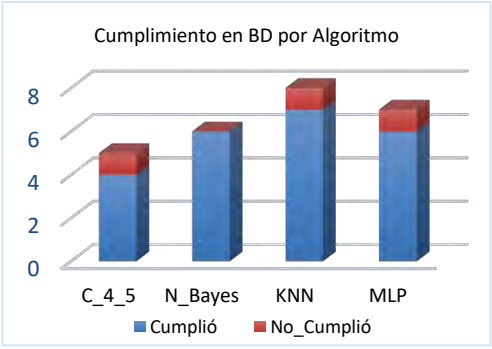
Tabla 6.5 Bases de datos donde se cumplió o no el patrón

# BD	Base de datos	Cumple	# BD	Base de datos	Cumple	# BD	Base de datos	Cumple
1	Weather_Nom	X	11	Diabetes	X	21	Weather_Num	√
2	Car	√	12	Glass	X	22	Vowel	√
3	Nursery	√	13	Breast_w	√	23	Zoo	√
4	Primary_tumor	√	14	Messidor_features	√	24	Tae	√
5	Led7	√	15	Sonar	√	25	Zoo	√
6	lymphography	X	16	Iris	√	26	Credit-g	X
7	Splice	√	17	Vehicle	√	27	Bands	X
8	Breast_Cancer	√	18	Waveform_5000	√	28	Credit_Aproval	√
9	Spect	√	19	Column3	√	29	Autos	√
10	Balance_Scale	√	20	Ecoli	√	30	Colic	√

Los patrones obtenidos anteriormente se cumplieron en 26 de los 30 conjuntos de datos, excepto en las base de datos: Weather_Nom, lymphography, Diabetes, Glass y Credit-g. Debido a que aunque con poca probabilidad, puede haber casos en los que al utilizar estos patrones para seleccionar el algoritmo en correspondencia con el conjunto de datos en tratamiento, el patrón no se cumpla, es importante mencionar que estos patrones le pueden servir al usuario únicamente como conocimiento de apoyo al momento de decidir que algoritmo aplicar con determinado conjunto de datos, lo cual es diferente a tomar la decisión sin tener ninguna idea sobre qué método utilizar.



(a) Cumplimiento de patrones por BD



(b) Cumplimiento de patrones por algoritmo

Figura 6.1 Cantidad de BD donde se cumple el patrón

6.3 Etapa de validación

Con la finalidad de validar los patrones obtenidos en la sección anterior, se utilizaron las bases de datos mostradas en la Tabla 6.2. En esta etapa, se aplicaron los mismos métodos que en la etapa de experimentación. En la Tabla 6.6 se muestran los porcentajes de precisión obtenidos, indicando en la última columna si el resultado coincide con la recomendación emitida por la regla correspondiente.

Tabla 6.6 Porcentajes de precisión en etapa de validación

Base de datos	C4.5	Naive Bayes	KNN	MLP	Cumplió
Vote	**96.30%**	90.10%	92.40%	94.71%	√
Letter	87.98%	64.11%	**96.03%**	82.08%	√
Monk	82.25%	77.41%	76.61%	**96.36%**	√
Sick	**98.80%**	92.60%	96.18%	97.24%	√
Ozone	**96.33%**	70.78%	96.26%	96.67%	√
Page_Blocks	**96.87%**	90.84%	96.01%	96.23%	X
Segment	96.92%	80.21%	**97.14%**	96.14%	√
Squash_Stored	66.38%	61.53%	**73.07%**	63.46%	√

Según los resultados obtenidos con el uso de 30 bases de datos se alcanzó en promedio un 86% de efectividad en las reglas obtenidas, mientras que en la etapa de validación se obtuvo un 87%. En particular, donde no se cumplió (conjunto de datos Page_Blocks de la Tabla 6.5), el algoritmo que debió ser seleccionado según la regla (Perceptrón Multicapa) quedó en segundo lugar y con una diferencia no significativa, lo cual indica que se podría usar este último sin perder precisión en su ejecución sobre los datos tratados. Los porcentajes de efectividad obtenidos muestran un comportamiento adecuado, también la variación mínima en la etapa de validación se cumplió un porcentaje alto en el uso de los patrones.

Para cada algoritmo se detectaron características que influyen en los resultados: el método C4.5 mostró una ventaja sobre el resto de los algoritmos cuando trata con bases de datos con valores faltantes, con Perceptrón Multicapa el factor que más influyó fue el número de clases tratadas y por último, con KNN y Naive Bayes la característica que más influyó fue el tipo de datos. Naive Bayes se comportó mejor con atributos nominales y KNN con numéricos y combinados.

Otro resultado en este estudio es que algunas de las características consideradas no influyeron en los resultados alcanzados, estas son: la estructura de la base de datos, si es o no balanceada, número de atributos y número de instancias.

Estos resultados pueden servir de apoyo para seleccionar el algoritmo de clasificación a aplicar sobre determinado conjunto de datos, aunque no se puede generalizar que funcionen para cualquier conjunto de datos.

Capítulo 7
Validación del modelo

En este capítulo, se demuestra la validez del modelo a partir de su aplicación en dos casos de estudio: predicción de deserción de alumnos de nivel superior y predicción de cáncer en pacientes. Primero se expone la metodología de validación y posteriormente se realiza una discusión sobre los resultados obtenidos.

7.1 Metodología de validación

Para validar el modelo se hace necesario crear una aplicación computacional que permita utilizar las técnicas de Minería de Datos e Inteligencia Artificial en un mismo ambiente, de tal forma que la interacción entre los algoritmos sea transparente para el usuario. Por tal motivo, se ha propuesto una metodología para validar el modelo, cuyos pasos consisten en:

1. Creación de un framework común para técnicas de MD e IA
2. Validación del funcionamiento de los algoritmos del framework
3. Aplicación del modelo a dos casos de estudio
4. Análisis y discusión de los resultados

A continuación, se explican los pasos de esta metodología.

7.1.1 Desarrollo del Framework

Se desarrolló un framework basado en una arquitectura de capas, esto con la finalidad de desacoplar o separar los distintos componentes involucrados (Presman, 2010). Esta arquitectura permite que los cambios posteriores que se requieran sólo afecten al nivel requerido. Considerando lo anterior y las características del framework, el mismo ha sido diseñado con una arquitectura de tres capas, siendo éstas: Capa de código, capa de datos y capa de aplicación.

En la Figura 7.1, se muestran los componentes de cada capa de la arquitectura del framework, los cuales se exponen a detalle en las secciones posteriores.

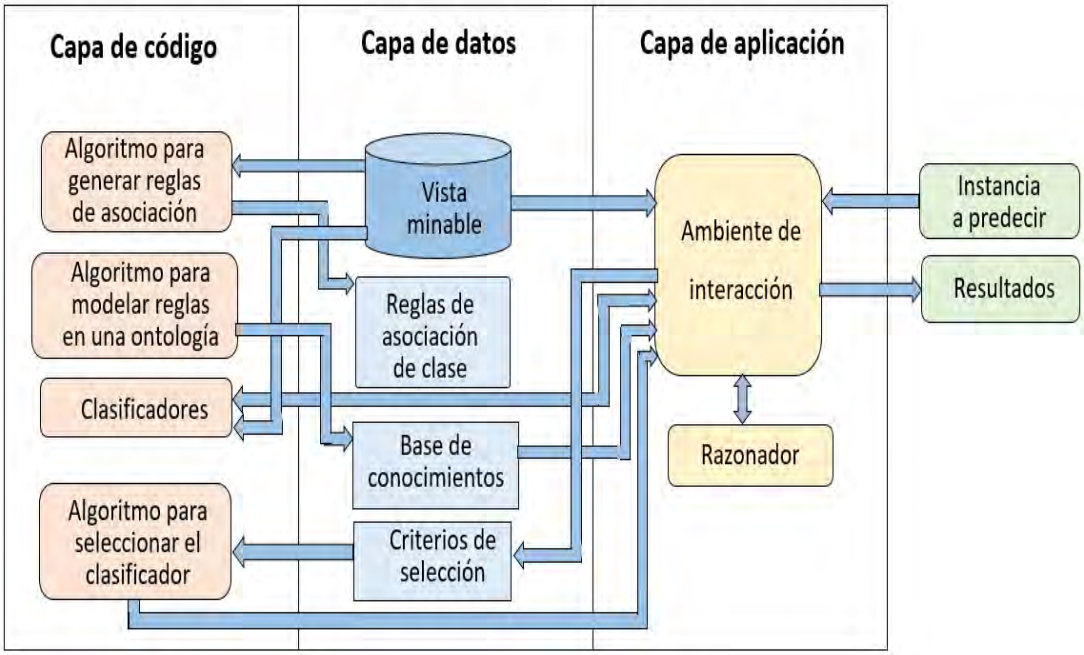


Figura 7.1 Arquitectura por capas del framework

7.1.1.1 Capa de código

La capa de código constituye el estrato donde se encuentran los algoritmos que están involucrados en el modelo, por lo que es considerada la que tiene el mayor nivel de abstracción.

Los algoritmos que aparecen en esta capa son de dos tipos: algoritmos suministrados y algoritmos definidos en este trabajo para la ejecución del modelo. De manera particular, los componentes de esta capa son:

- *Algoritmo para generar las RAC*: Método TNR que permite generar las reglas que se presentan con mayor frecuencia, para lo cual se requiere como entrada los valores de los parámetros de soporte y confianza.
- *Algoritmo para modelar RAC en ontologías*: Método que representa un conjunto de RAC generadas previamente en una ontología, mediante la representación de componentes del tipo: clases, propiedades de datos y axiomas.

- *Clasificadores*: Métodos de clasificación que permiten predecir la categoría de la instancia en tratamiento. En el modelo se tienen disponibles los algoritmos J48, Naive Bayes, MLP y K-vecinos.
- *Algoritmo para seleccionar el clasificador*: Método que selecciona el clasificador que brinda el mejor desempeño tomando en consideración las características del conjunto de datos en tratamiento. Para ello, utiliza los criterios de selección de la capa de datos.

7.1.1.2 Capa de datos

Esta capa se define con el objetivo de almacenar y suministrar datos o conocimiento a los algoritmos utilizados y a la capa de aplicación. A continuación, se explica cada uno de los componentes:

- *Vista minable*: Archivo con formato ARFF que contiene los datos fuente sobre los que trabajarán los algoritmos de clasificación y de generación de RAC.
- *Reglas de asociación de clase*: Conjunto de implicaciones almacenadas en formato de texto que contiene las correlaciones entre ítems más frecuentes de la vista minable.
- *Criterios para seleccionar el clasificador*: Condiciones representadas en una estructura de árbol donde al final de cada rama termina con una hoja que contiene el nombre del clasificador.
- *Base de conocimientos*: Contiene conocimiento representado en un esquema ontológico. Está conformada por un conjunto de conceptos, propiedades y axiomas, este último componente es utilizado para proporcionar las recomendaciones al usuario.

7.1.1.3 Capa de aplicación

Esta capa sirve para programar el funcionamiento del modelo haciendo uso de los recursos de la capa de datos y de código. Utiliza una interfaz gráfica para solicitar y presentar información al usuario. Hace uso de los siguientes componentes:

- *Interfaz gráfica*: Sirve para recibir los datos de la nueva instancia a predecir y mostrar los resultados. Las opciones que incluye este ambiente son:
    - Cargar base de datos
    - Generar base de conocimientos
    - Seleccionar clasificador
    - Predecir.
- *Razonador*: Modelo de inferencias *Pellet* que realiza deducciones con el uso de un conjunto de axiomas, esto con la finalidad de obtener la categoría de la clase de la instancia en tratamiento y las recomendaciones en los casos que lo amerite. Se utilizó este razonador debido a que está implementado en Java y permite hacer inferencias y consultas basadas en ontologías descritas en lenguaje OWL.

7.1.2 Pruebas de funcionalidad del framework

En la capa de código del framework están integrados varios algoritmos que ayudan a efectuar las tareas que constituyen el proceso del modelo. Para validar estos algoritmos se utilizaron siete bases de datos de diferentes características obtenidas del repositorio UCI Machine Learning (Lichman, 2013). En la Tabla 7.1, se muestran las características de cada una.

Tabla 7.1 Bases de datos utilizadas para probar el framework

Base de datos	Número de instancias	Número de Atributos	Tipo de Datos	Número de clases
Breast Cancer	158	9	Nominal	2
Pime Diabetes	768	8	Nominal	2
Car	1228	7	Nominal y numérico	4
Lenses	84	4	Nominal	3
Iris	187	23	Numérico	2
Weather_Nom	14	4	Nominal	2
Nursery	12960	8	Nominal	5

Las pruebas realizadas al framework se dividen en las siguientes tres tareas:

- Generación de las reglas de asociación de clase
- Representación de las ontologías
- Funcionamiento de los clasificadores

De las siete bases de datos mostradas en la Tabla 7.1, se eligió el conjunto de datos Pime Diabetes para detallar como se efectuaron las tareas de validación y que resultados se obtuvieron.

7.1.2.1 Generación de las reglas de asociación de clase

Estas pruebas se efectuaron mediante la comparación de las reglas de asociación de clase generadas por el framework contra las que se generaron con la herramienta SPMF (Fournie, Gomariz, & Soltani, 2014). En las Figuras 7.2 y 7.3, se muestran las reglas generadas en el framework y en la herramienta SPMF respectivamente.

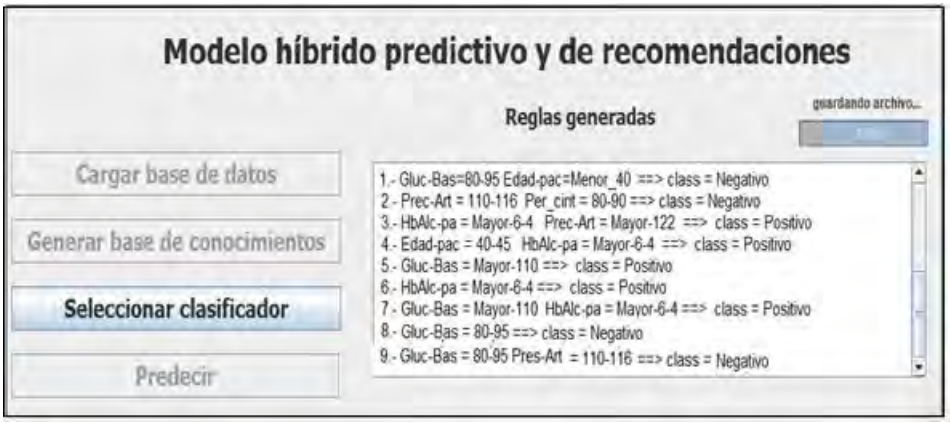


Figura 7.2 Conjunto de Reglas generadas con el framework

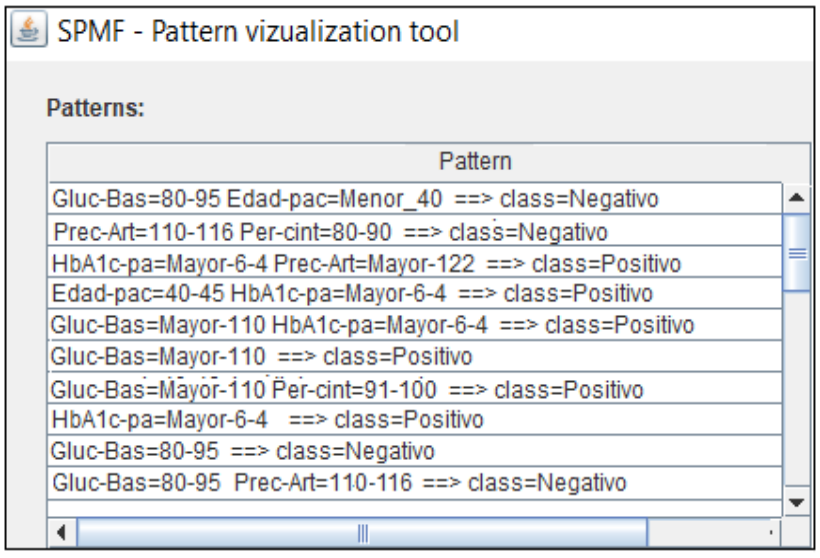


Figura 7.3 Conjunto de Reglas generadas con la herramienta SPMF

7.1.2.2 Modelación de las ontologías

Después de modelar la ontología se obtuvo el documento de tipo OWL, cuyo funcionamiento se comprobó utilizando la herramienta Protegé 5.0 (Musen M. , 2014). En la Figura 7.4 se muestra la ontología resultante ejecutada con dicha herramienta.

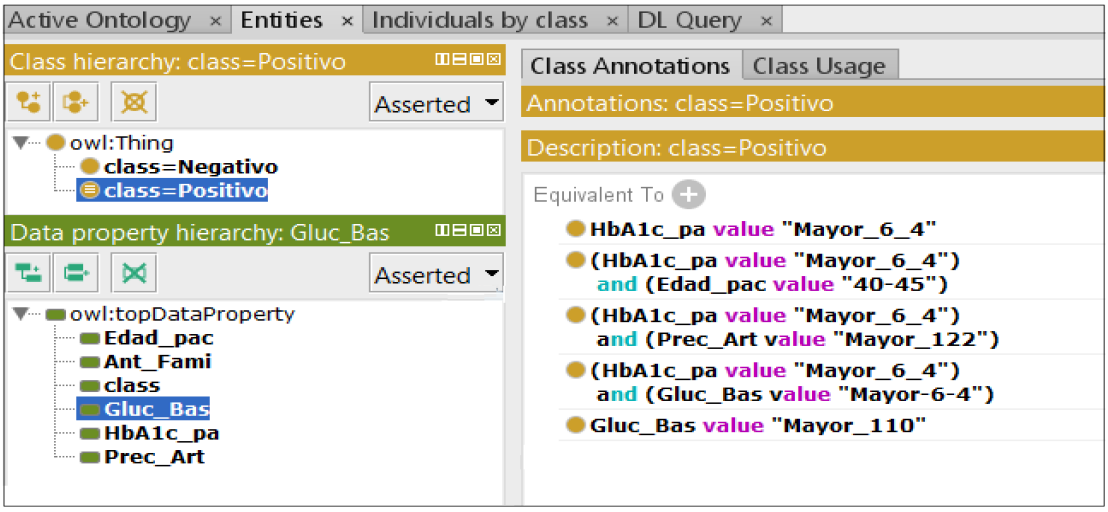


Figura 7.4 Ontología generada para predicción de Diabetes

La ontología generada se compone de dos conceptos o clases de equivalencia y cinco propiedades de datos. Por su parte, la clase de equivalencia *Positivo* contiene cinco axiomas que corresponden a las reglas 3, 4, 5, 6 y 7 mostradas en la Figura 7.2.

7.1.2.4 Funcionamiento de los clasificadores

La validación de los métodos de clasificación integrados, se realizó mediante la comparación de los resultados obtenidos en el framework con los que se proporcionó la herramienta Weka (Hall M. F., 2009). En la Tabla 7.2, se muestran los resultados obtenidos con las instancias de prueba del conjunto de datos Pime Diabetes.

Tabla 7.2 Resultados obtenidos con instancias de prueba del conjunto Pime Diabetes

Instancias probadas	**Categoría obtenida con la herramienta Weka**	**Categoría obtenida con el framework**
40-45, Alguno, Mayor-110, Mayor-6-4, 101-120, Mayor-122	Positivo	Positivo
46-50, Ambos, Mayor-110, Mayor-6-4, 101-120, Mayor-122	Positivo	Positivo
40-45, Alguno, 80-95, 5-8-6-4, 80-90, 110-116	Negativo	Negativo
Menor_40, Alguno, 80-95, Menor-5-7, 80-90, 117-122	Negativo	Negativo

7.1.3 Validación del modelo

Con las pruebas realizadas sobre el framework se corroboró que los algoritmos que se incorporaron funcionan de manera correcta, ahora es posible utilizar este ambiente de programación para validar el modelo propuesto, cuya finalidad es comprobar que en la tarea de predicción el modelo híbrido brinda más elementos al usuario para la toma de decisiones.

Para validar el modelo se utilizaron seis casos de estudio, cuyas bases de datos se refieren a, una de alumnos pertenecientes a una institución educativa de nivel superior y cinco del repositorio UCI, las cuales son: Breast Cancer, Car, Pime Diabetes, Lenses y Nursery.

Para estos casos las predicciones a efectuar se pueden clasificar en los siguientes dos tipos:

- *Tipo 1*: Cuando en la base de conocimientos únicamente se encuentran axiomas pertenecientes a la misma clase que obtuvo el clasificador, esto ocurre con mayor frecuencia con las instancias más representativas de la clase. En estos casos el modelo permite mostrarle al usuario la categoría y los patrones detectados.
- *Tipo 2*: Cuando en la base de conocimientos, además de detectar axiomas correspondientes a la categoría que obtuvo el clasificador, encuentra axiomas de otra categoría, esto ocurre comúnmente con las instancias menos representativas de la clase. Bajo este escenario se le notifican al usuario las recomendaciones que se deben atender para seguir perteneciendo o dejar de pertenecer a determinada clase.

En la siguiente sección, se exponen los tres casos con los que se validó el modelo.

7.1.3.1 Primer caso: Predicción de alumnos en riesgo de deserción

En este caso se utiliza el modelo para predecir cuáles estudiantes se encuentran en riesgo de desertar en su carrera a nivel licenciatura. Para ello, se utiliza un conjunto de datos de alumnos egresados que se caracterizan por 6 atributos y consta de 142 instancias, estructurados en dos categorías. Los datos de los atributos, se muestran en la Tabla 7.3.

Tabla 7.3 Atributos y valores de la base de datos para predecir deserción estudiantil

Atributo	Descripción	Tipo de atributo	Valores
Tipo_ Bachillerato	Bachillerato proveniente	Nominal	CBTIS, CBTA, COBAEH, PARTICULAR
Especial	El tipo de especialidad cursada en bachillerato	Nominal	Computación, Administración, General
Pro_Mate	Promedio de matemáticas en primer semestre	Nominal	< 70, 70-85, 86-100
Pro_Prog	Promedio de programación en primer semestre	Nominal	<70, 70-85, 86-100
Pro_MDis	Promedio de matemáticas discretas en primer semestre	Nominal	<70, 70-85, 86-100
Est_Padr	Nivel de estudios del padre	Nominal	Lic, Prep, Sec
Clase	Categoría de clase	Nominal	Egresa, No_Egresa

Para efectuar la validación del modelo, mediante la opción generar base de conocimientos, por una parte se generaron las reglas de asociación de clase con el método TNR en un documento con formato TXT y por otra, la ontología en un documento con extensión OWL. En el Cuadro 7.1 se presentan las reglas de asociación de clase generadas con un soporte y confianza de 0.85.

Cuadro 7.1 Reglas generadas para predecir deserción estudiantil

```
Pro_Prog=<70 Pro_Mate=<70  ==> class=No_Egresa
Tip_Bach=CBTA Pro_Prog=<70  ==> class=No_Egresa
Pro_Prog=<70  ==> class=No_Egresa
Pro_Prog=86-100  ==> class=Egresa
Pro_Mate=86-100  ==> class=Egresa
Especial=Comp Pro_Prog=86-100  ==> class=Egresa
Est_Padr=Sec Pro_Prog=<70  ==> class=No_Egresa
Tip_Bach=CBTA Especial=Admin Pro_Prog=<70  ==> class=No_Egresa
Est_Padr=Sec Pro_Prog=<70  ==> class=No_Egresa
```

La ontología generada en lenguaje OWL está integrada por conceptos, propiedades de datos y axiomas. En el Cuadro 7.2, se muestra un extracto del código OWL que incluye dos de los axiomas generados.

Cuadro 7.2 Ontología generada para predecir deserción estudiantil

```
<Ontology>
  <Declaration>
   <Class IRI="#class= Egresa  "/>
  </Declaration>
   <Declaration>
    <Class IRI="#class= No_Egresa  "/>
   </Declaration>
   <Declaration>
     <DataProperty  IRI="# Tip_Bach "/>
  </Declaration>
   <Declaration>
    <DataProperty  IRI="# Pro_Mate "/>
   </Declaration>
   <Declaration>
     <DataProperty  IRI="# Pro_Prog "/>
   </Declaration>
   <Declaration>
    <DataProperty  IRI="# Especial "/>
   </Declaration>
   <EquivalentClasses>
      <Class IRI="#class= Egresa "/>
        <ObjectIntersectionOf>
             <DataProperty IRI="# Especial "/>
            <Literal datatypeIRI="&rdf;
             PlainLiteral">Comp</Literal>
            </DataHasValue>
             <DataHasValue>
             <DataProperty IRI="# Pro_Prog "/>
              <Literal datatypeIRI="&rdf;
               PlainLiteral">86-100  </Literal>
            </ObjectIntersectionOf>
     </EquivalentClasses>
      <EquivalentClasses>
       <Class IRI="#class= Pro_Prog "/>
         <ObjectIntersectionOf>
             <DataProperty IRI="# <70 "/>
             <Literal datatypeIRI="&rdf;
              PlainLiteral">Comp</Literal>
             </DataHasValue>
              <DataHasValue>
              <DataProperty IRI="# Pro_Mate "/>
               <Literal datatypeIRI="&rdf;
                PlainLiteral"><70 </Literal>
       </ObjectIntersectionOf>
      </EquivalentClasses>
</Ontology>
```

Después de generarse la base de conocimientos, se introdujeron las características del conjunto de datos (base de datos completa, dos clases y tipo de datos nominal) mostradas en la Figura 7.5. En respuesta, el framework sugirió Naive Bayes como el clasificador que brinda el mejor desempaño según las características de la base de datos.

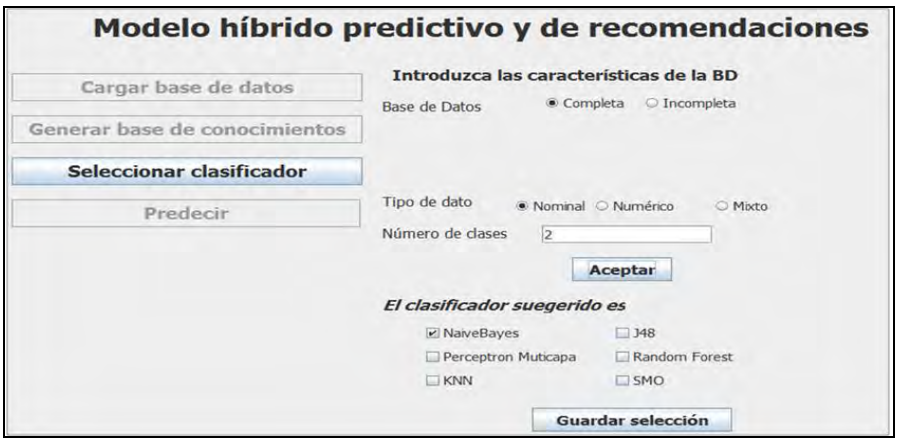


Figura 7.5 Módulo para elegir el clasificador

Para efectuar la etapa de clasificación, del conjunto total de datos se detectaron instancias que estuviesen involucradas en los dos tipos de predicción, incluyendo un total de 21 pruebas, cuyos resultados se muestran en la Tabla 7.4.

Tabla 7.4 Predicciones de deserción estudiantil

#	Instancias probadas	Categoría del clasificador	Detectó axiomas de clase distinta a la del clasificador
1	CBTIS, Comp, 68, 68, Prep, 68	No_Egresa	No
2	PART, Gral, 80, 68, Sec, 70_85	No_Egresa	No
3	CBTA, Gral, 80, 68, Sec, 70_85	No_Egresa	No
4	PART, Admin, 80, 68, Sec, 70_85	No_Egresa	No
5	CBTIS,Comp,65, 85,Prep,94	Egresa	**Si**
6	COBAEH, Admin, 80, 68, Sec, 70_85	No_Egresa	No
7	COBAEH, Gral, 80, 68, Sec, 70_85	No_Egresa	No
8	CBTIS, Comp, 93, 93, Prep, 93	Egresa	No
9	CBTIS, Comp, 93, 80, Prep, 93	Egresa	No
11	CBTIS,Comp,68,83,Lic,96	Egresa	**Si**
12	COBAEH, Gral, 68, 68, Sec, 68	No_Egresa	No
13	CBTIS, Gral, 80, 68, Sec, 80	No_Egresa	No
14	COBAEH, Comp, 68, 80, Lic, 80	No_Egresa	No
15	COBAEH,Gral,83,78,Sec,80	No_Egresa	**Si**
16	CBTIS, Comp, 93, 93, Prep, 93	Egresa	No
17	PART, Comp, 93, 80, Prep, 93	Egresa	No
18	CBTIS, Admin, 93, 93, Prep, 93	Egresa	No
19	CBTA,Gral,90,80,Sec,82	No_Egresa	**Si**
20	CBTA, Comp, 93, 80, Prep, 93	Egresa	No
21	COBAEH, Comp, 93, 80, Prep, 93	Egresa	No

Del total de las pruebas, 17 corresponden al primer tipo de predicción donde axiomas detectados corresponden a la clase proporcionada por el clasificador. Por otra parte, cuatro pruebas corresponden al segundo tipo (5, 11, 15 y 19), debido a que la base de conocimientos logró detectar que estas instancias tienen ciertas características que influyen para pertenecer a otra categoría.

Para las 17 instancias del primer tipo, el modelo le notifica al usuario el patrón que influyó para pertenecer a esa clase. Por ejemplo, en la Figura 7.6 el framework muestra los resultados que se obtuvieron con la primera instancia de la Tabla 7.4, indicando que la Base de Conocimientos proporcionó la categoría No_Egresa y dando a conocer el patrón que encontró para que dicha instancia pertenezca a esa clase.

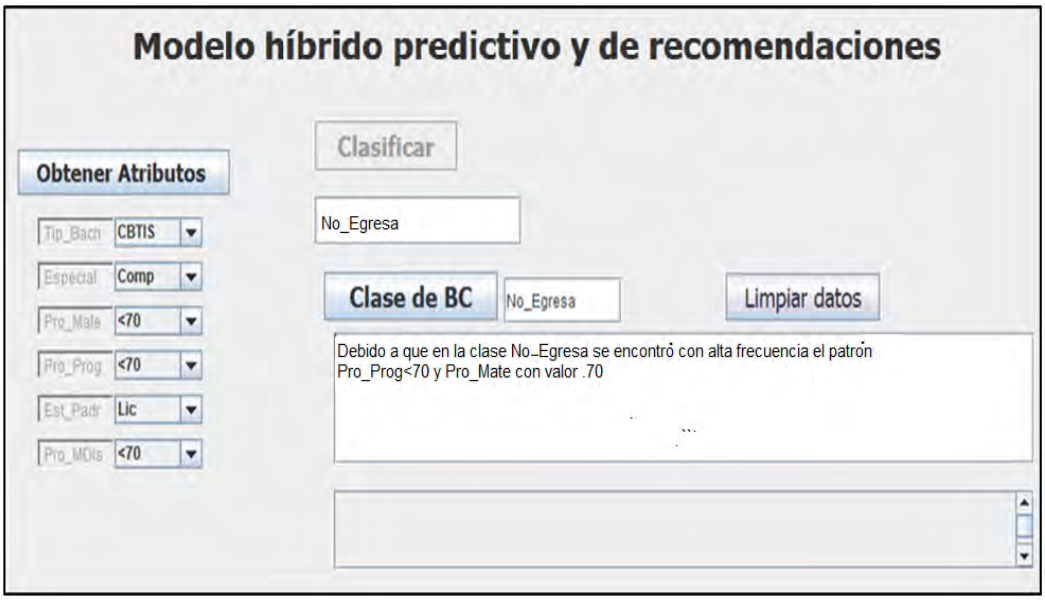


Figura 7.6 Caso donde coincide la categoría en predicción de deserción

Para las instancias donde la categoría de clase proporcionada por el clasificador y la base de conocimientos no coinciden, el sistema es capaz de explicar bajo qué condiciones la nueva instancia puede cambiar de clase. Por ejemplo, en la Figura 7.8 se presentan los resultados que se obtuvieron con la instancia 18 de la Tabla 7.4, el clasificador da como resultado la categoría Egresa y la Base de Conocimientos la categoría No_Egresa, por ello en la parte de recomendaciones le notifica al usuario que aspectos debe atender.

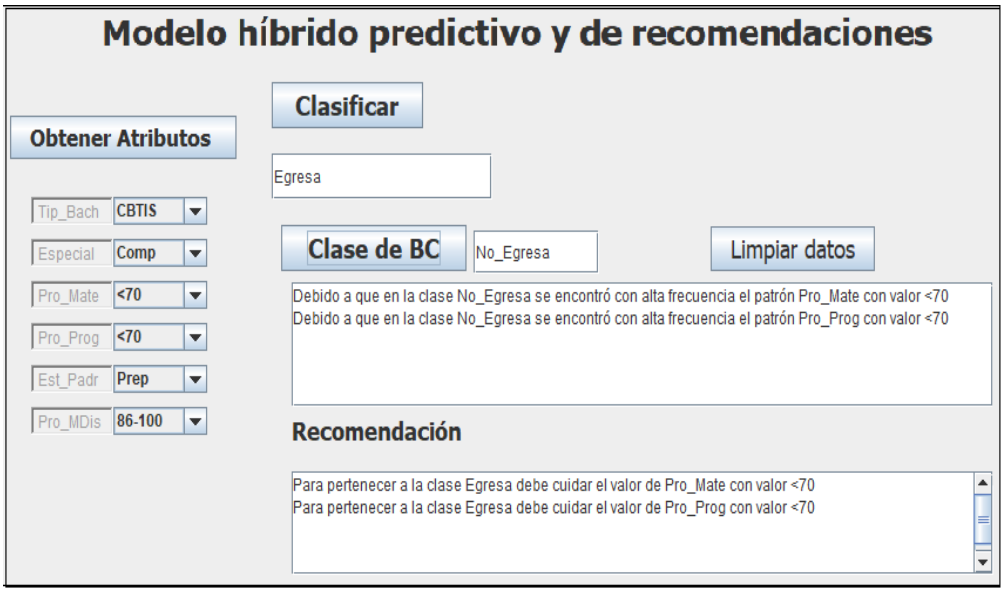


Figura 7.7 Caso donde no coincide la categoría en predicción de deserción

7.1.3.2 Segundo caso: Predicción de cáncer en mujeres

En este caso se utiliza el modelo para predecir en los pacientes si los tumores de cáncer de seno son benignos o no. Para ello, se utiliza la base datos *Breast cancer* del repositorio UCI que contiene información sobre 10 atributos y consta de 158 instancias, estructurados en dos categorías. Los datos de los atributos se muestran en la Tabla 7.5.

Tabla 7.5 Atributos y valores para predicción del cáncer

Atributo	Descripción	Valores
age_paci	Edad del paciente	10-19, 20-29, 30-39, 40-49, 50-59, 60-69
menopaus	Momento de menopausia	lt40, ge40, premeno (antes de los 40, después de los 40 o premenopausia)
tumor_size	Tamaño de tumor en milímetros	0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44,45-59
nodes_pa	Célula en nodos linfáticos	0-2, 3-5, 6-8, 9-11, 12-14, 15-17, 18-20, 21-23, 24-26, 27-29, 30-39
node_cap	Células en capsulas linfáticas	yes, no
deg-mali	Grado histológico del tumor	1, 2, 3
breast_c	Lado afectado	left, right
breast-q	Cuadrante afectado	left_up, left_low, right_up, right_low
irradiat	Radioterapia	yes, no
clase	Indica recurrencia	rec, no_rec (Recurrente y No recurrente)

Después de seleccionar la base de datos, se creó la base de conocimientos resultando dos tipos de archivos, uno con extensión TXT que contiene el conjunto de reglas de clase generadas y otro con extensión OWL que contiene la base de conocimientos, las reglas obtenidas para este caso se muestran en el Cuadro 7.3.

Cuadro 7.3 Reglas generadas para predicción del cáncer

```
menopaus=premeno node_cap=yes  ==> class=rec
breast_c=left deg-mali=3  ==> class=no_rec
menopaus=ge40  ==> class=no_rec
age_paci=40-49  node_cap=yes ==> class=rec
age_paci=40-49 irradiat=yes  ==> class=rec
menopaus=ge40 breast-q=left_low  ==> class=no_rec
age_paci=40-49 irradiat=no node_cap=yes  ==> class=rec
menopaus=ge40  deg-mali=3  ==> class=no_rec
age_paci=40-49 node_cap=yes ==> class=rec
```

La ontología modelada en lenguaje OWL contiene los conceptos, propiedades de clase y axiomas. En la etapa de predicción se realizaron pruebas con las 24 instancias, cuyos resultados se muestran en la Tabla 7.6.

Tabla 7.6 Casos para predecir cáncer donde se obtuvo la misma categoría

#	Instancias probadas	Categoría del clasificador	Detectó axiomas de clase distinta a la del clasificador
1	30-39, ge40, right, no, 3, yes, 30-34, 3-5, r_up	no_rec	No
2	30-39, ge40, right, no, 3, yes, 30-34, 0-2, r_up	no_rec	No
3	30-39, ge40, right, no, 3, yes, 30-34, 3-5, r_up	no_rec	No
4	30-39, lt40, left, yes, 3, yes, 0-4, 0-2, left_up	no_rec	No
5	30-39, premeno, right, no, 2, no, 0-4, 0-2, left_up	rec	**Si**
6	30-39, ge40, left, yes, 3, yes, 5-9, 0-2, left_up	no_rec	No
7	30-39, ge40, right, yes, 3, yes, 5-9, 6-8, left_up	no_rec	No
8	40-49, premeno, right, yes, 3, yes, 15-19, 0-2, left_up	rec	No
9	40-49, lt40, right, yes, 1, yes, 0-4, 0-2, left_up	rec	No
11	30-39, premeno, left, no, 2, no, 0-4, 0-2, left_up	rec	**Si**
12	40-49, lt40, right, yes, 2, yes, 0-4, 6-8, left_up	rec	No
13	40-49, lt40, right, yes, 2, yes, 0-4, 3-5, left_low	rec	No
14	20-29, premeno, right, yes, 2, yes, 5-9, 3-5, r_low	rec	No
15	40-49, ge40, right, yes, 3, yes, 15-19, 0-2, left_up	rec	No
16	30-39, ge40, cent, no, 30-34, 0-2, no, 3, right	rec	**Si**
17	40-49, premeno, right, yes, 2, yes, 5-9, 3-5, r_low	rec	No
18	30-39, ge40, right, no, 3, yes, 30-34, 3-5, r_up	no_rec	No
19	30-39, ge40, right, no, 3, yes, 30-34, 0-2, r_up	no_rec	No
20	20-29, ge40, left, no, 30-34, 0-2, no, 3, left_low	rec	**Si**
21	30-39, ge40, right, no, 3, yes, 30-34, 3-5, r_up	no_rec	No
22	30-39, ge40, right, no, 3, yes, 30-34, 3-5, r_up	no_rec	No
23	30-39, ge40, right, no, 3, yes, 30-34, 0-2, r_up	no_rec	No
24	30-39, ge40, right, no, 3, yes, 30-34, 3-5, r_up	no_rec	No

Con 20 instancias de la Tabla 7.6 se efectuó predicción del primer tipo, donde se detectaron axiomas correspondientes a la categoría proporcionada por el clasificador.

Por ejemplo, en la Figura 7.8 se muestra el resultado obtenido después de efectuar la predicción con la primera instancia de la Tabla 7.6, en este caso se detectó en la Base de Conocimientos un axioma correspondiente a la categoría No_rec, notificándole al usuario el patrón que encontró para que dicha instancia pertenezca a esa clase.

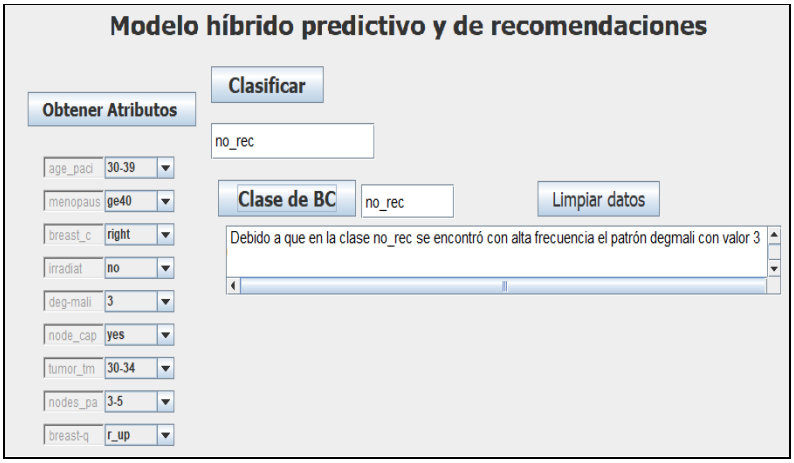


Figura 7.8 Caso donde coincide la categoría en predicción de cáncer de mujer

Por otra parte, las pruebas realizadas con las instancias 5, 11, 16 y 20 corresponden al segundo tipo de predicción, donde se detectaron axiomas que corresponden a una categoría distinta a la que proporcionó el clasificador. Por ejemplo, en la Figura 7.9 se presentan los resultados que se obtuvieron con la instancia 20 de la Tabla 7.6, el clasificador da como resultado la categoría *rec* y notifica el axioma encontrado perteneciente a la categoría *no_rec*, por ello en la parte de recomendaciones le notifica al usuario que aspectos debe atender.

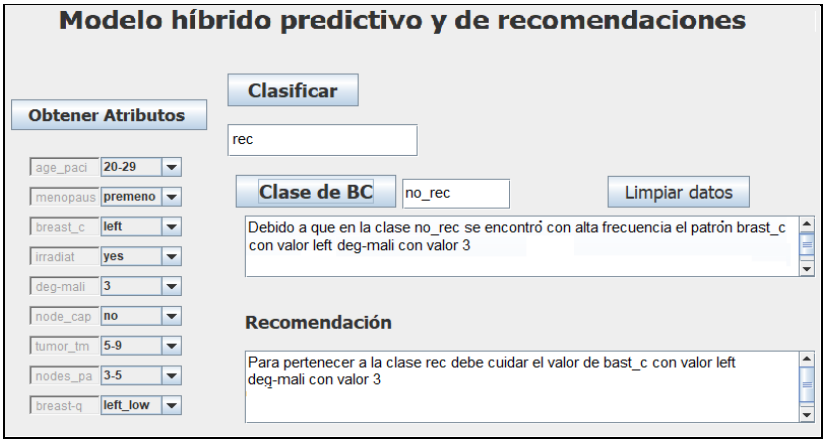


Figura 7.9 Caso donde no coincide la categoría en predicción de cáncer de mujer

7.1.3.3 Tercer caso: Predicción de tipo de auto

En este caso se utiliza el modelo para predecir el tipo de auto a comprar. Para ello, se utiliza la base datos *Car Evaluation* del repositorio UCI que contiene información sobre 6 atributos y consta de 1728 instancias, estructurados en cuatro categorías. Los datos de los atributos se muestran en la Tabla 7.7.

Tabla 7.7 Atributos del conjunto de datos car

Atributo	Descripción	Valores
buying	Precio de compra	vhigh, high, med, low
maint	Precio de mantenimiento	vhigh, high, med, low
doors	Número de puertas	2, 3, 4, 5more
persons	Número de personas	2, 4, more
lug_boot	Tamaño de cajuela	small, med, big
safety	Tipo de seguro	low, med, high
clase	Indica tipo compra	unacc, acc, good, vgood

En la primera tarea se generaron las reglas que se muestran en el Cuadro 7.4.

Cuadro 7.4 Reglas generadas para predicción de tipo de auto

```
buying_c=Vhigh maint_ca=Vhigh  ==> class=Unacc
buying_c=High maint_ca=High  ==> class=Acc
Safety_c=Cuatro maint_ca=Med  ==> class=Good
buying_c=Low  ==> class=Good
buying_c=Vhigh maint_ca=Vhigh  ==> class=Unacc
maint_ca=High  ==> class=Acc
Safety_c=Cuatro maint_ca=Med  ==> class=Good
Safety_c=Cuatro buying_c=Low  ==> class=Good
buying_c=High ==> class=Acc
Safety_c=Cuatro maint_ca=Med  buying_c=Low  ==> class=Good
```

En la etapa de predicción se realizaron pruebas con 26 instancias, cuyos resultados se muestran en la Tabla 7.8.

Por ejemplo, en la Figura 7.10 se muestran los resultados que se obtuvieron con la primera instancia de la Tabla 7.8, donde se detectaron en la Base de Conocimientos únicamente axiomas correspondientes a la categoría *Acc* y se le notifica al usuario el patrón que encontró para que dicha instancia pertenezca a esa clase.

Tabla 7.8 Casos para predecir tipo de auto donde se obtuvo la misma categoría

#	Instancias probadas	Categoría del clasificador	Detectó axiomas de clase distinta a la del clasificador
1	High,High,Dos,CinMore,Low,CinMore	Acc	No
2	High,High,Dos,Cuatro,Low,CinMore	Acc	No
3	High,Vhigh,Cuatro,Dos,Low,Dos	Vgood	Si
4	Vhigh,Vhigh,CinMore,Cuatro,Good,Cuatro	Unacc	No
5	Vhigh,High,Dos,Cuatro,Med,Cuatro	Unacc	No
6	Vhigh,Vhigh,Cuatro,CinMore,Med,CinMore	Unacc	No
7	Vhigh,Vhigh,CinMore,CinMore,Good,CinMore	Unacc	No
8	High,High,Dos,Cuatro,Low,Dos	Vgood	Si
9	Med,Med,CinMore,Cuatro,Med,Dos	Good	No
11	Med,Med,CinMore,Cuatro,Good,Cuatro	Good	No
12	Med,Med,CinMore,Cuatro,Good,Cuatro	Good	No
13	High,High,Dos,Cuatro,Good,CinMore	Acc	No
14	Med,High,Dos,Dos,Med,CinMore	Acc	No
15	Low,High,Dos,Cuatro,Low,CinMore	Acc	Si
16	High,High,Cuatro,Dos,Good,Cuatro	Unacc	No
17	Vhigh,Vhigh,CinMore,Cuatro,Good,Cuatro	Unacc	No
18	Vhigh,Vhigh,Dos,Cuatro,Med,Cuatro	Unacc	No
19	Vhigh,Vhigh,Cuatro,Dos,Med,CinMore	Unacc	Si
20	High,Vhigh,CinMore,CinMore,Good,CinMore	Acc	No
21	High,High,Cuatro,Dos,Low,Cuatro	Good	No
22	Med,Med,CinMore,Cuatro,Med,Dos	Good	No
23	Med,Med,CinMore,Cuatro,Good,Cuatro	Good	Si
24	Med,Vhigh,Cuatro,CinMore,Med,CinMore	Unacc	No
25	Low,Vhigh,CinMore,CinMore,Good,CinMore	Acc	Si
26	High,Med,Cuatro,Dos,Low,Cuatro	Good	No

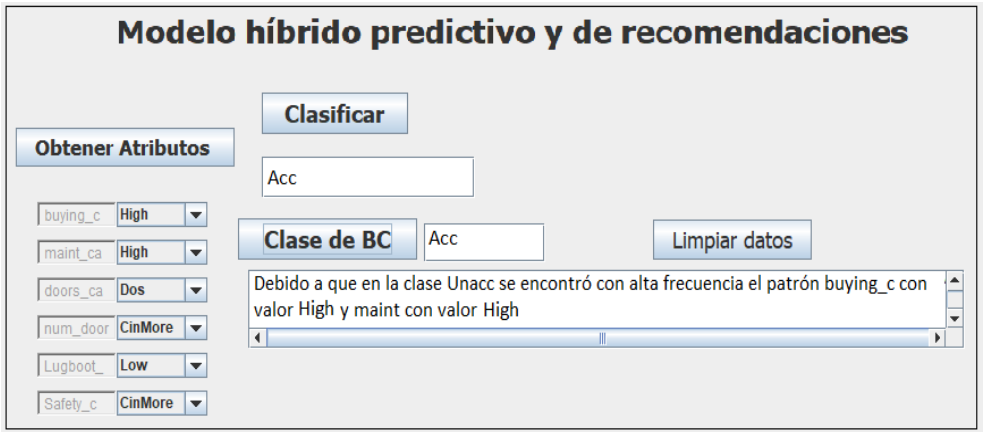


Figura 7.10 Caso donde coincide la categoría en predicción de tipo de auto

Por otra parte, las pruebas realizadas con las instancias 3, 8, 15, 19, 23 y 25 corresponden al segundo tipo de predicción, donde se detectaron axiomas que corresponden a una categoría distinta a la que proporcionó el clasificador. Por ejemplo,

en la Figura 7.11 se presentan los resultados que se obtuvieron con la instancia 3 de la Tabla 7.6, el clasificador da como resultado la categoría *Vgood* y notifica el axioma encontrado perteneciente a la categoría *Unacc*, por ello en la parte de recomendaciones le notifica al usuario que aspectos debe atender.

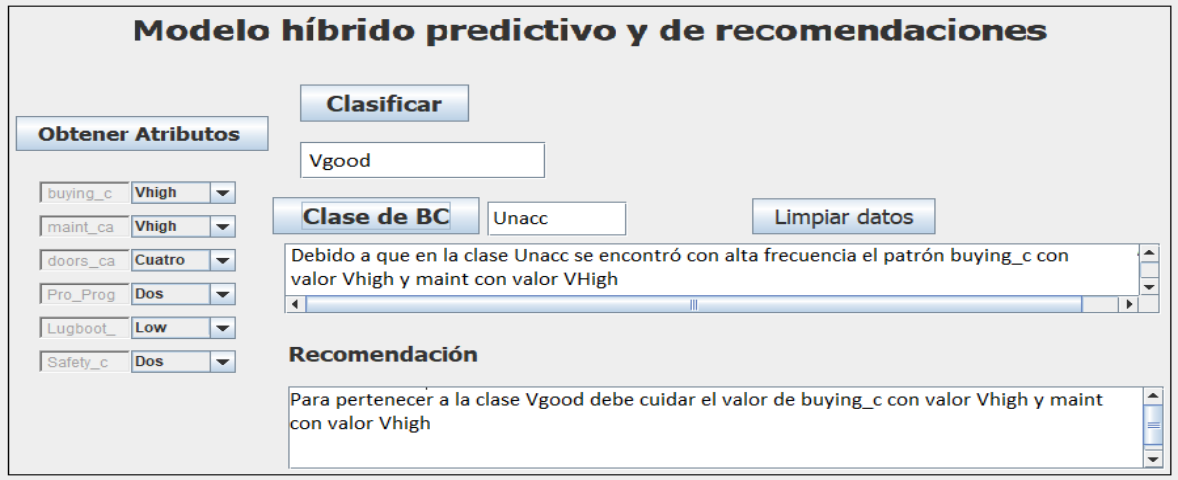


Figura 7.11 Caso donde no coincide la categoría en predicción de tipo de auto

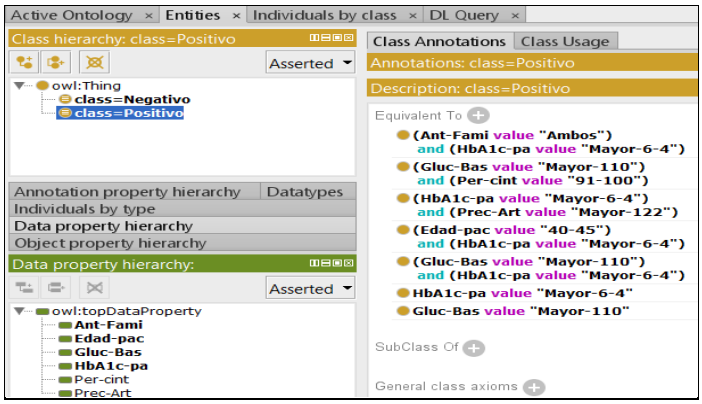
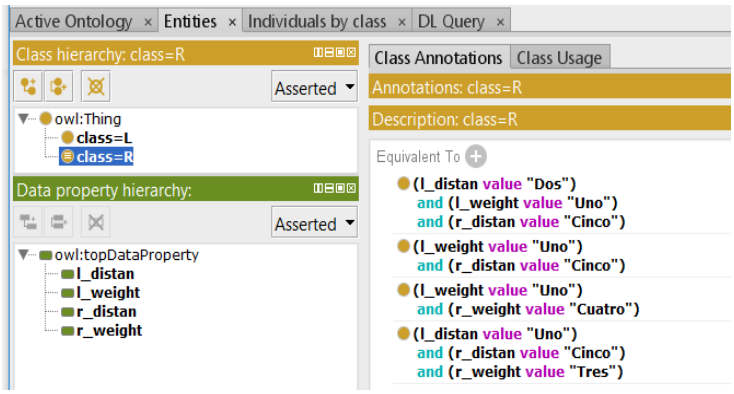
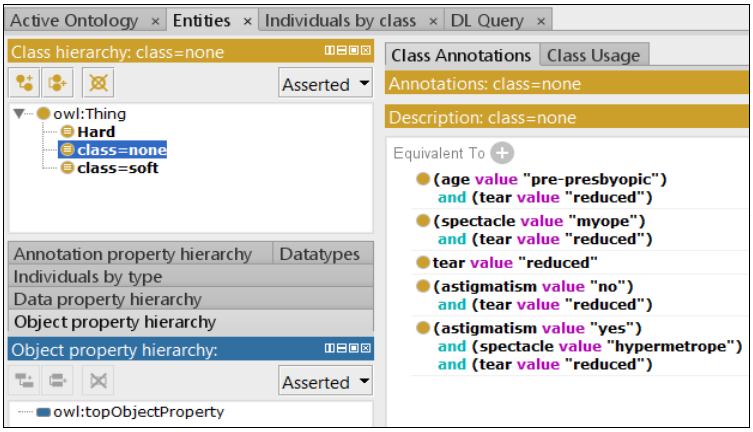
7.1.3.4 Casos de estudio para los conjuntos de datos Diabetes, Balance Scale y Lenses

En estos tres casos se realizó la predicción sobre tipo de diabetes, tipo de desarrollo cognitivo y tipo de lentes. En las Tablas 7.9 y 7.10 se muestran las reglas y ontologías generadas para cada caso.

Tabla 7.9 Reglas generadas con las bases de datos Diabetes, Balance y Lenses

Base de datos	Reglas Generadas
Pime Diabetes	HbA1c-pa=Menor-5-7 ==> class=Negativo Gluc-Bas=Mayor-110 ==> class=Positivo Ant-Fami=Ambos HbA1c-pa=Mayor-6-4 ==> class=Positivo Gluc-Bas=Mayor-110 ==> class=Positivo Gluc-Bas=80-95 ==> class=Negativo Prec-Art=110-116 Per-cint=80-90 ==> class=Negativo HbA1c-pa=Menor-5-7 ==> class=Negativo Per-cint=80-90 ==> class=Negativo
Balance	l_weight =Uno r_distan =Cinco ==> class=R l_weight =Uno r_weight=Cinco l_distan=Dos ==> class=R l_distan=Uno r_distan=Cinco r_weight =Tres ==> class=R r_distan =Uno r_weight=Dos l_weigh=Tres ==> class=L l_weight =Uno r_weight =Cuatro ==> class=R l_distan =Uno b=Cuatro l_weigh=Cuatro ==> class=L
Lenses	age = pre-presbyopic tear = reduced ==> class=none astigmatism=no tear=normal ==> class=soft spectacle=myope tear=reduced ==> class=none tear=reduced astigmatism=no ==> class=none spectacle=myope tear=normal astigmatism=yes ==> class=hard astigmatism=no spectacle=hyperm tear=reduced ==> class=none tear=normal astigmatism=yes ==> class=hard astigmatism=no age = presbyopic ==> class=soft

Tabla 7.10 Ontologías generadas con las bases de datos Diabetes, Balance y Lenses

Base de datos	Ontología Generada
Pime Diabetes	
Balance	
Lenses	

7.2 Discusión de los resultados

En la tarea de modelación de reglas de asociación de clase en esquema ontológico, en los tres casos de estudio se comprobó la consistencia de los axiomas generados, automatizando con ello la emisión de recomendaciones. Por su parte, para la tarea de

predicción, el modelo ayudó a elegir el clasificador, en el primer caso sugirió Naive Bayes, en el segundo KNN y en el tercero MLP.

Considerando los resultados obtenidos con las pruebas realizadas en los tres casos de estudio, respecto al primer tipo de predicción donde se encontraron axiomas pertenecientes a la misma categoría que proporcionó el clasificador, en el primer caso se presentó en un 77% de las instancias, en el segundo en un 83% y en el tercero en un 72%. Lo anterior indica que el modelo logró detectar patrones de comportamiento de otra clase en un 23%, 17% y 29% para el primero, segundo y tercer caso respectivamente. Siendo estos, buenos porcentajes de instancias sobre los que el modelo fue capaz de avisar con anticipación qué patrones debe atender el usuario para mejorar su situación.

En el tercer caso de estudio se comprobó que el modelo propuesto funciona adecuadamente con conjuntos de datos de más de dos clases, se constató que en los casos donde se detectaron axiomas pertenecientes a una clase distinta al clasificador se emitieron correctamente las respectivas recomendaciones.

Con los resultados obtenidos en la validación del modelo se comprobaron las siguientes ventajas que ofrece:

- En los casos donde sólo se detectan axiomas relacionados con la misma categoría proporcionada por el clasificador, se dan a conocer los factores que influyeron para que la nueva instancia pertenezca a dicha categoría, a diferencia del resultado que proporciona un modelo predictivo convencional donde sólo se proporciona el nombre de la categoría. Este conocimiento extra le ayudará al usuario a tener mayores recursos para la toma de decisiones debido a que le permite saber qué indicadores debe atender.
- Los casos en los que el framework detectó axiomas pertenecientes a una categoría diferente a la que obtuvo el clasificador, se le proporcionan al usuario las recomendaciones que debe tomar en cuenta. Por ejemplo, en el segundo caso se sugiere que debe hacer el paciente para dejar de pertenecer a la clase padecer cáncer.

En diversos casos, el modelo propuesto logró encontrar axiomas correspondientes a una categoría distinta a la que proporcionó el clasificador debido a que la base de conocimientos contienen los patrones extraídos mediante la generación de reglas de asociación de clase, la cual permite encontrar todas las correlaciones que se presentan con mayor frecuencia entre los distintos valores de los atributos.

Los resultados obtenidos demuestran las ventajas que proporciona el modelo, debido a su capacidad que tiene para detectar los comportamientos más frecuentes que ocasionan que una instancia pertenezca a una clase y en algunos casos comportamientos que pueden afectar a futuro a una instancia para pertenecer o dejar de pertenecer a determinada clase.

Conclusiones

En el presente trabajo, se propuso un modelo híbrido de predicción. Se demostró que el modelo propuesto, descrito en el capítulo tres, resuelve algunos inconvenientes que presenta un modelo predictivo típico. La combinación de técnicas de Minería de Datos e Inteligencia Artificial en este nuevo modelo, permitió brindarle más elementos al usuario para la toma de decisiones.

Como otra ventaja del modelo, la incorporación en el modelo de un método de generación de reglas de asociación ayudó a identificar comportamientos particulares en los datos, para posteriormente mediante el uso de técnicas de recomendación, proporcionárselos al usuario en caso de presentarse.

Teniendo en cuenta que los métodos de generación de reglas de asociación abordan de manera diferente el problema de redundancia, se efectuó la experimentación, descrita en el capítulo cinco, donde se demostró que el algoritmo TNR se desempeña mejor frente a este problema, brindando mejores resultados sobre los métodos que utilizan FCI y sobre los que recurren a un post-procesamiento. La incorporación de TNR en el modelo propuesto, permitió obtener los resultados de manera óptima ya que la representación de los axiomas se efectúa a partir del conocimiento integrado por un conjunto de reglas que se generaron de manera previa.

Para la tarea de predicción, se obtuvo un conjunto de patrones que le sirven de ayuda al usuario como conocimiento de apoyo para tomar la decisión sobre qué clasificador aplicar tomando en consideración las características del conjunto de datos en tratamiento. Estos criterios se obtuvieron después de evaluar el desempeño respecto a la precisión que proporcionaron los algoritmos C4.5, Naive Bayes, MLP y KNN frente a diversos conjuntos de datos.

Por otra parte, en el presente trabajo se demostraron las ventajas que proporciona la representación del conocimiento extraído con técnicas de Minería de Datos utilizando técnicas de Inteligencia Artificial. Esto mediante el modelado del conocimiento extraído de una base de datos con la generación de RAC, representándolo en un lenguaje formal para ser utilizado en la toma de decisiones.

Se aportó un método para la modelación de las RAC en un sistema basado en conocimiento bajo un esquema ontológico, lo cual permitió proporcionarle al usuario las recomendaciones pertinentes de manera natural. Todas las componentes de la ontología, modeladas a partir de las RAC, permitieron la integración del conocimiento contenido en una Base de Conocimientos**.** Sin embargo, la incorporación de los axiomas brindaron una aportación especial, debido a que son los elementos que permiten inferir conocimiento que no está indicado explícitamente en la taxonomía de conceptos.

Para validar el modelo fue necesario crear una aplicación de computadora (framework) que permitiera utilizar las técnicas de Minería de Datos e Inteligencia Artificial en un mismo ambiente, de tal forma que la interacción entre los algoritmos fuese transparente para el usuario. Los resultados que proporciona el framework en las tareas de generación de reglas, generación de ontología y clasificación fueron validados y contrastados con las herramientas Weka, Protegé y SPMF, respectivamente.

Debido a que el proceso de validación se llevó a cabo utilizando bases de datos de diferentes características, esto permitió constatar que el modelo propuesto se desempeña de forma correcta con conjuntos de datos que contienen más de dos clases. En estos casos, la predicción de instancias donde la base de conocimientos contenía axiomas relacionados con una clase distinta al clasificador, se emitieron correctamente las respectivas recomendaciones.

Considerando las conclusiones mencionadas anteriormente, la contribución principal de este trabajo se resume en los siguientes puntos:

- Se comprobó que el uso del modelo en la tarea de predicción, le brinda al usuario resultados más precisos.
- El uso de técnicas de Inteligencia Artificial en el modelo permite proporcionar los resultados al usuario de una forma más comprensible.
- El modelo propuesto le proporciona al usuario más elementos para la toma de decisiones.
- En la validación, se pudo detectar que el modelo es capaz de personalizar las recomendaciones proporcionadas al usuario, debido a que los resultados emitidos se basan en la correlación de los datos introducidos de la nueva instancia con los patrones existentes en la base de datos histórica.

Aún con los esfuerzos que se han realizado para la mejora de los modelos de predicción y los avances alcanzados hasta ahora, todavía se presentan algunos desafíos relacionados con dichos modelos. De esta manera, existen retos no atendidos en esta Tesis, que permitirían reducir aún más las limitaciones presentes en los sistemas predictivos.

Una de las áreas por madurar es la predicción en ambientes Web. Debido a que el número de usuarios en los sistemas Web crece de manera considerablemente por su constante popularización, se requieren adaptaciones y extensiones en este ambiente. La metodología propuesta podría ser extendida para hacer frente a problemas recurrentes en áreas del ámbito de la minería Web y a nuevas limitaciones que puedan surgir en un futuro que pudieran tener su origen en la dispersión de datos.

Por otra parte, la migración del conocimiento extraído con técnicas de minería de datos en un esquema de representación del conocimiento, es una de las áreas que aún tiene trabajo por realizar, aunque en el modelo propuesto en este trabajo se brinda un enfoque de solución ontológico mediante el uso de clases de equivalencia y axiomas, se pueden desarrollar otros enfoques que resuelvan el problema.

Finalmente, aunque se atendió la migración de conocimiento extraído mediante generación de reglas de asociación. Sin embargo, este mismo principio puede aplicarse a conocimiento extraído con otro tipo de técnica de minería de datos como el agrupamiento.

Trabajo futuro

El modelo propuesto contiene conocimiento obtenido con técnicas de Minería de Datos, sin embargo, puede enriquecerse aún mediante otras técnicas, por lo que se sugiere incorporar al modelo un módulo donde se pueda agregar a la base de conocimientos el conocimiento del experto para contar con más recursos para la emisión de recomendaciones.

Otra tarea que permitirá obtener mayores beneficios en el modelo es la incorporación de técnicas de agrupamiento, lo cual permitirá detectar en el conjunto de datos original, las instancias atípicas existentes.

Respecto a la representación del conocimiento en una base de conocimientos, aunque para ello el modelo sólo utiliza OWL, puede ser traducida y representada en otros lenguajes como RDF o XML para permitir mayor compatibilidad con otras herramientas.

Por otra parte, es importante mencionar que las reglas de asociación pueden ser fácilmente combinadas con métodos de aprendizaje automático. Por lo tanto, el algoritmo TNR, podría ser combinado con algún método basado en contenido con el objeto de mejorar la calidad de las recomendaciones.

Glosario de términos

API (Application Programming Interface)

Interface de lenguaje de programación que relaciona o permite extender el programa. Efectúa una serie de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos.

CART (Classification And Regression Trees)

Método de clasificación basado en árboles de decisión, se caracteriza por ser un algoritmo de los más utilizados en estadística para predecir valores de variables continuas debido a su fácil interpretación.

CLIPS (C Language Integrated Production System)

Es una herramienta que provee un entorno de desarrollo para la producción y ejecución de sistemas expertos.

ECLAT (Equivalence CLAss Transformation)

Método de generación de reglas basado en obtención de itemsets frecuentes mediante el uso de identificadores de transacciones.

FCI (Frequent Close Itemset)

Itemset frecuente que se caracteriza por abarcar otros itemsets que contienen menos pares atributo-valor.

FP-TREE (Frequent Pattern Tree)

Método de generación de reglas basado en obtención de itemsets frecuentes mediante el uso de una estructura de árbol.

Item

Es el valor que tiene un atributo dentro de un conjunto de datos.

Itemset

Está conformado por un conjunto de ítems o pares atributo-valor.

KDD (Knowledge Data Discovery)

Es un proceso que permite identificar patrones válidos, novedosos y potencialmente útiles a partir de un conjunto de datos.

MLP (Multi Layer Perceptrón)

Método de clasificación basado en redes neuronales, se basa en una red neuronal artificial formada por múltiples capas, esto le permite resolver problemas que no son linealmente separables.

Outlier

Es 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 que deberían ser examinados detenidamente.

OWL (Ontology Web Languaje)

Ontology Web Languaje, lenguaje para representar ontologías basado en marcas.

PROLOG (PROgrammation en LOGique)

Es un lenguaje de programación lógica que permite crear programas que se componen de cláusulas de Horn que forman reglas del tipo "ponendo ponens".

RAC (Rule Association Clase)

Es un tipo de regla de asociación que se caracteriza por contener un solo item en el sucedente.

RDF (Resource Description Framework)

Es un lenguaje de ontologías que proporciona los elementos básicos para la descripción de vocabularios orientado al ambiente de la web.

Transaction IDentifier (TID)

Es el número con el que se identifica una transacción dentro de un conjunto de datos.

Top-K Rules Non Redundant (TNR)

Algoritmo de generación de reglas de asociación que trata el problema de redundancia en reglas.

SVM (Support Vector Machine)

Método de clasificación basado en kernels o núcleos que representa a los puntos de muestra en el espacio, separando las clases a dos espacios.

XML (eXtensible Markup Language)

Es un lenguaje de marcas utilizado para guardar datos de forma legible, permite utilizar una definición gramática similar a la de HTML.

Referencias

Agrawal, R. (1993). Mining Associations beetwen sets of items in massive databases. *In ACM-SIGMOD International Conference on Data*, 207-216.

Agrawal, R. (1994). Fast algorithms for mining association rules in large databases. *Proceedings of the 20th International Conference on Very Large Data Bases*, 487-499.

Akinola, S., & Oyabugbe, O. (2015). Accuracies and Training Times of Data Mining Classification Algo-rithms: An Empirical Comparative Study. *Indian Journal of Science and Technology, Vol 8, No. 15*, pp. 440-447.

Anbuselvan, S., & Balamuralithara, B. (2014). Holistic Prediction of Student Attrition in Higher Learning Institutions in Malaysia Using Support Vector Machine Model. *International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)*, 29-35.

Ashari, A., Paryudi, & Tjoa, A. (2013). Performance Comparison between Naive Bayes, Decision Tree and k-Nearest Neighbor in Searching Alternative Design in an Energy Simulation Tool. *International Journal of Advanced Computer Science and Applications (IJACSA), 4*(11), 33-39.

Baader, F. (2010). *The description Lógic Handbook, Theory, Implementation and Applications* (Vol. 2). Cambridge Inglaterra: Cambridge University Press.

Balasubamanian, M. (2014). Churn prediction in mobile Telecom System using Data Mining techniques. *International Journal of Scientific and Research Publications, 1*(4), 1-6.

Beverton, R., & Holt, S. (1957). On the Dynamics of exploited fish populations. *UK Ministry of Agriculture ans Fisheries, 2*(1), 533-541.

Bichen, Z., Jinghe, Z., & Sang, H. (2015). Predictive modeling of hospital readmissions using metaheuristics and data mining. *Expert Systems with Applications, 42*(20), 7110-7120.

Bonde, A., & Gore, D. (2014). Comparative Study of Associa-tion Rule Mining Algorithms. *International Journal of Computer Science and Information Technologies (IJCSIT)*, Vol. 5, pp. 6153-6157.

Brachman, R. J. (1983). What IS-A Is and Isn't: An analysis of Taxonomic Links in Semantic Networks. *IEEE Computer, 16*(10), 30-36.

Cabena, P. (1998). *Discovering Data Mining From Concept to Implementation* (Vol. 1). Prentice Hall.

Clifford, L. (1991). *Neural networks: theoretical foundations and analysis* (1 ed., Vol. 1). NJ, USA: IEEE Press.

Cohen, W. (1995). Fast Effective Induction. *International conference Twelfth on Machine Learning, 1*, 115-123.

Coomans, D., & Massart, D. (1982). Alternative k-nearest neighbour rules in supervised pattern recognition: k-Nearest neighbour classification by using alternative voting rules. *Analytica Chimica Acta, 136*, pp.15-27.

Cuadras, C. (2007). *Nuevos métodos Estadísticos* (Vol. 1). España: CMS Editions.

Cuello, G. (2006). *Técnicas de Minería de Datos dentro de contextos metodológicos y de empresa.* Argentina.: Instituto Tecnologico de Buenos Aires.

Devroye, L. (1996). A Probabilistic Theory of Pattern Recognition. *Springer-Verlang*.

Deza, E., & Deza, M. (2009). Encyclopedia of Distances. *Springer, 1*, 94-115.

Divya, R., & Kumar, L. (2012). Survey on AIS, Apriori and FP-Tree algorithm. *International Journal of Computer Sci-ence and Management Research, 1*(2), 130-136.

Diwani, S., & Sam, A. (2014). *Diabetes Forecasting Using Supervised Learning Techniques, ACSIJ Advances in Computer Science: an International Journal, Vol. 3, Issue 5, No.11 , ISSN : 2322-5157*.

Doreswamy, L., & Hemanth, D. (2011). Hybrid Data Mining Technique for Knowledge Discovery from Discovery from Engineering Materials Datasets. *Computer Science, 3*(1), 166-177.

Eibé, F., & Witten, I. (1998). Generating Accurate Rule Sets Without Global Optimization. *Fifteenth International Conference on Machine Learning, 1*, 144-151.

Entezari, R., Rezaei, A., & Minaei, B. (2009). Comparison of Classification Methods Based on the Type of Attributes and Sample Size. *Journal of Convergence Information Technology, 4*(3), 94-102.

Espín, V., Hurtado, M., & Noguera, M. (2016). Nutrition for Elder Care: a nutritional semantic recommender system for the elderly. *Expert Systems*, pp. 201-210.

Fayyad, U. (2009). *Data Mining To Knowledge Discovery: An Overview. In Advances In Knowledge Discovery And Data Mining* (Vol. 1). Menlo Park, CA: AAAI Press / The MIT Press.

Finlay, S. (2014). *Predictive Analytics, Data Mining and Big Data. Myths, Misconceptions and Methods.* (1st edición). Basingstoke: Palgrave Macmillan.

Fournie, V., Gomariz, P., & Soltani, A. (2014). SPMF: a Java Open-Source Pattern Mining Library. . *Journal of Machine Learning Research (JMLR), 15: 3389-3393.*

Frawley, W., Shapiro, G., & Matheus, C. (1992). Knowledge Discovery in Databases: An Overview. *AI Magazine AAAI, 13*(3), 57-70.

Genkin, A. (2004). Large-Scale bayesian logistic for text categorization. *Learning Machine, 1*, 291-304.

Gerben. (2009). Predicting Students Drop Out: A Case Study. *Educational Data Mining, 1*, 41-50.

Ghezaiel, L., Latiri, C., & Ahmed, C. (2012). Ontology Enrichment based on Generic Basis of Association Rules for Conceptual Document Indexing. *International Conference on Knowledge Engineering and Ontology Development, 1*, 53-65.

Girotra, M., Nagpal, K., & Se Minocha and Ne Sharma. (2013). Comparative Survey on Association Rule Mining Algo-rithms. *International Journal of Computer Applications, 84*(10), 18-23.

Gopalachari, V., & Sammulal, P. (2014). Personalized Web Page Recommender System using integrated Usage and Content Knowledge. *IEEE Advanced Communication Control and Computing Technologies, 1*, 1066-1071.

Gruber, T. (1993). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human-Computer Studies, 43*(5), 907-928.

Guida, G., & Tasso, C. (1994). *Design and Development of Knowledge-Based Systems. From Life Cycle to Methodology* (Vol. 1). England: John Wiley and Sons Ltd.

Hall, M. F. (2009). The WEKA Data Mining Software, [http://www.cs.waikato.ac.nz /ml/weka].

Hall, M., Frank, E., Holmes, G., & Pfahringer, B. (2009). The WEKA Data Mining Software, [http://www.cs.waikato.ac.nz /ml/weka], 2009.

Hand. (2001). *Principles of Datamining.* Massachussets Institute of Technology: MA:The MIT Press.

Harmelen, F. (2008). The Knowledge Representation, Semantic Web. Holanda: University Amsterdam.

Hernández, J., Ramírez, M., & Ferri, C. (2013). *Introducción a la Minería de Datos.* México: Pearson Prentice Hall.

Holte, R. (1993). Very Simple classification rules perform well on most commonly used dataset. *Machine Learning, 11*, 63-91.

Hudaib, A., Dannoun, R., & Harfoushi, O. (Mayo de 2015). Hybrid Data Mining Models for Predicting Customer Churn. *Communications, Network and System Sciences, 1*, 91-96.

Jayavani, K., & Nawaz, M. (2014). Surveillance of Efficient Algorithms for Mining Frequent Itemsets and Closed Frequent Itemsets. *International Journal of Informative & Futuristic Research, Multidisciplinary Research, 1*(11), 174-183.

Jiawei, H. (2004). Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Aproach. *Springer, Data Mining and Knowledge Discovery, 29*(2), 1-12.

Kendall, J. (2008). *Designing and assessing educational objectives: Applying the new taxonomy.* California, EE.UU.: Corwnin Press.

Kononenko, I. (1990). Comparison of inductive and naive Bayesian learning approaches. *Proceedings of the 6th European Working Session on Learning, 1*, 206–219.

Li, Y., & Roy, U. (2015). Integrating rule-based systems and data analytics tools using open standard PMML. *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 1-12.

Lichman. (2013). UCI Machine Learning Repository. *School of Information and Computer Science, Acceso Febrero 2014, Irvine, CA: University of California*.

Light, R., & Margolin, B. (1971). An analysis of variance for categorical gorical data. *Statistical Association*, 534-544.

Liu, B. (1999). Association Rules and Sequential Patterns,. *Data-Centric Systems and Applications*, 2-7.

López, J. (2005). Comparación de modelos de clasifficación automática de patrones texturales de minerales presentes en los carbones colombianos. *Dyna, 2*, 115-124.

Lu, H., & Setiono, R. (1997). Effective Data Mining Using Neural Networks. *IEEE Transactions on Knowledge and Data Engineering, Vol. 1*, 957-961.

Marinica, C., & Guillet, F. (2009). Improving Post-mining of Association Rules with Ontologies. *The XIII International Conference Applied Stochastic Models and Data Analysis, 1*, 76-81.

Márquez, C. (2013). Predicting of school failure using data mining techniques. *IEEE-RITA, 7*(3), 7-14.

Minsky, M. (1975). *A Framework for Representing Knowledge .* The Psychology of Computer Vision: McGraw-Hill.

Miquelez, T., Bengoetxea, E., & Larranaga, P. (2004). Evolutionary Computation based on Bayesian Classifier. *nternational Journal Application Mathematics Computation, 3*, 335–349.

Moran, S., He, Y., & Liu, K. (2009). Choosing the Best Bayesian Classifier: An Empirical Study. *IAENG International Journal of Computer Science (IJCS), 36*(4), 25-34.

Musen, M. (2014). The Protégé project: A look back and a look forward. AI Matters. *Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4).*

Nachev, A. (2016). Using Multi-Layer Perceptrons for Analysis of Labour. *International Conference Artificial Intelligence, Cairnes School of Business & Economics, NUI, Galway, Ireland*, 223-229.

Nardi, D., & Brachman, J. (2002). *An introduction to description logics. In Description logic Handbook.* Cambridge: University Press.

Nirenburg, S. (2004). *Ontologicaal Semaantics.* Baltimore County: University of Maryland.

Paiva, L., Figueiras, P., & Lima, C. (2014). Discovering Semantic Relations from Unstructured Data for Ontology Enrichment Asssociation rules based approach. *Iberian Information Systems and Technologies*, Vol. 1, pp. 1-6.

Pajares, G. (2010). *Inteligencia Artificial e Ingeniería del Conocimiento.* España: Alfa-Omega.

Palma, J., & Marín, R. (2008). *Inteligencia Artificial. Técnicas, métodos y aplicaciones.* Madrid, España: McGRAW-HILL.

Pasquier, N. (2004). Discovering Frequent Closed Itemsets for Association Rules. *Complexe Scientific, Dedex Francia*.

Pérez, L., & Santín, G. (2007). *Mineria e Datos. Técnicas y Herramientas.* Madrid: Thomson.

Presman, R. (2010). *Ingeniería del Software: Un enfoque practico* (Vols. 1, Septima Edición). Madrid España: Mc-Graw-Hill.

Prithiviraj, P., & Porkodi, R. (2015). A Comparative Analysis of Association Rule Mining Algorithms in Data Mining: A Study. *American Journal of Computer Science and Engineering Survey, 1*, 98-105.

Quinlan. (1996). Improved Use of Continuous Attributes in C4.5. *Journal of Artificial Intelligence Research, 4*, 77-90.

Russell, S., & Norvig, P. (2004). *Inteligencia Artificial, Un enfoque moderno.* Salamanca: Pearson, Prentice Hall.

Saheb., R., Solaiman, B., Hamrouni, K., & Mnif, N. (2016). Association rules-based Ontology Enrichment. *International Journal Web Applications, 8*(1), 16-25.

Saxena, A., & Gadhiya, S. (2014). A Survey on Frequent Pat-tern Mining Methods Apriori, Eclat, FP growth. *International Journal of Engineering Development and Research, 2*, 92-96.

Schalkoff, J. (1990). *Artificiaal Inteligence.* EU: Springer.

Shweta, K., & Sunita, S. (2016). Weighted Naive Bayes Classifier: A Predictive Model for. *International Journal of Computer Applications* , 33-37.

Sinha, G., & Ghosh, S. (2014). Identification of Best Algo-rithm in Association Rule Mining Based on Performance. *Journal of Computer Science and Information Technology - IJCSMC, 3*(11), 38-45.

Solanki, S., & Soni, N. (2013). A Survey on Frequent Pattern Min-ing Methods Apriori, Eclat, FP growth. *International Journal of Computer Techniques, Computer Engineering Department*, 86-89.

Tatsiopoulos, C., & Boutsinas, B. (2009). Ontology Mapping based on Association Rule Mining. *International Conference on Enterprise Information Systems, 1*, 34-40.

Thompson, W., & Bell, F. (1934). Biological Statistics of Pacific Halibut Fishery. *Pacific Halibut*, 89-95.

Timarán, R. (2012). Applied data mining patterns to discovery survival in women with invasive cervical cancer. *SCIELO, Universidad y Salud, ISSN 0124-7107*, 117-129.

Uschold, M. (1996). Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 93–155.

Vani, K. (2015). Comparative Analysis of Association Rule Mining Algorithms Based on Performance Survey. *International Journal of Computer Science and Information Technologies – IJCSIT, 6*, 3980-3985.

Vapnik, V. (1995). Support vector networks. *Machine Learning, vol. 20*, 273-297.

Venu, M., & Sammula, P. (2014). Personalized Web Page Recommender System. *IEEE Advanced Communication Control and Computing Technologies*, 1066-1071.

Viger, F., & Philippe. (2012). Mining Top-K Association Rules. *Dept. of Computer Science, University of Moncton, Canada*.

Vivekananth, P. (2012). Different Data Mining Algorithms: A Performance Analysis. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), 1*(3), 75-84.

Zaki. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*.

Zaki, M., & Hsiao, C. (2002). *An Efficient Algorithm for Closed Itemser Mining, Computer Science Departament, Institute Troy NY*.

Zeigler , B. P. (1986). System Knowledge: A Definition ans its mplications. *In: Modelling and Simulation Methodology in the Artificial Intelligence*, 15-17.

Zhang, Y., & Deng, A. (2015). Redundancy Rules Re-duction in Rule-Based Knowledge Bases. *12th International Conference on Fuzzy Sys-tems and Knowledge Discovery (FSKD)*, 639-643.

Zhao, Z. (2014). A research of optimal Rejection Thresholds based on ROC curve. *School of Air and missile defense, Air force Engineering University*.

Zwirk, U., & Wigury, I. (2013). Automated Tests, Repeatable Experiments, Meaningful Results, http//dataset.org/repo/PROMISE/EffortPrediction.