Universidad Autónoma del Estado de México

Centro Universitario UAEM Zumpango

Ingeniería en Computación

Análisis Comparativo de Árboles de Decisión y Máquina de Vectores Soporte para conjuntos de datos de Diabetes y Hepatitis

Tesis

Que para obtener el título de

Ingeniero en Computación

presenta:

Rodolfo Iván Flores de la Torre

Asesor:

Dr. Asdrúbal López Chau

Enero 2014

Dedicatoria

A la mujer que amo, María Nazaria Vargas Flores, que con su ejemplo, consejos y su apoyo incondicional fue posible alcanzar la culminación de este trabajo, gracias mi amor.

A mi mamá, que sin su apoyo no hubiera alcanzado el sueño anhelado, este es el trabajo en donde está puesto todo lo invertido y la confianza que depositaste en mi. Sin duda alguna eres mi ejemplo a seguir en la vida con tus buenos sentimientos, valores y la perseverancia para lograr algo. Espero no decepcionarte, gracias mamá.

A mi papá, que ha estado conmigo a pesar de la distancia, y me impulsó a terminar la carrera y me ha brindado su apoyo cuando lo he necesitado, gracias papá.

A mis hermanos Dulce, Tony y Dany que sin sus consejos de seguir adelante, y de la vida cotidiana, me han permitido aprender de cada uno ya que poseen diferentes formas de pensar, gracias.

A mi asesor de tesis, Dr. Asdrúbal López Chau, que sin su ayuda este trabajo no hubiera sido posible, gracias maestro.

De igual forma, dedico esta tesis al Ing. Jesús Flores Maya, que ha sido mi ejemplo a seguir de forma profesional y de llevar la vida con gran entusiasmo, gracias maestro.

Zumpango Estado de México, a 19 de noviembre de 2013.

Dr. Raymundo Ocaña Delgado

Subdirector académico del CU UAEM Zumpango

PRESENTE

Sirva este medio para saludarlo y para informarle que el alumno **Rodolfo Iván Flores de la Torre** ha **concluido** su **trabajo de tesis**, bajo mi dirección.

Los datos completos son los mostrados a continuación.

Título de la tesis: "Análisis Comparativo de Árboles de Decisión y Máquina de Vectores Soporte para Conjuntos de Datos de Diabetes y Hepatitis"

Alumno: Rodolfo Iván Flores de la Torre

Carrera: Ingeniero en Computación

Número de cuenta: 0720489

Generación: 2007-2012

Sin más por el momento, me despido agradeciendo su atención.

ATENTAMENTE

Dr. Asdrúbal López Chau

Profesor de Tiempo Completo del

Centro Universitario UAEM Zumpango

Universidad Autónoma del Estado de México
UAEM

Centro Universitario UAEM Zumpango

Martes, 19 de noviembre del 2013
Oficio No. TIT/662/13

C. RODOLFO IVÁN FLORES DE LA TORRE
PASANTE DEL PE DE ICO
DEL CENTRO UNIVERSITARIO UAEM ZUMPANGO
P R E S E N T E

Por este conducto, la Subdirección Académica del Centro Universitario UAEM Zumpango informa a usted que, los siguientes profesores han sido designados como **REVISORES** del trabajo de titulación bajo la modalidad de tesis denominada: **"ANÁLISIS COMPARATIVO DE ÁRBOLES DE DECISIÓN Y MÁQUINA DE VECTORES SOPORTE PARA CONJUNTOS DE DATOS DE DIABETES Y HEPATITIS."**

NOMBRE	DE ACUERDO	APROBADO
DRA. MARÍA DE LOURDES LÓPEZ GARCÍA		
M. EN C. VALENTÍN TRUJILLO MORA		

Quien deberá emitir su dictamen por escrito en **diez días hábiles.**

A T E N T A M E N T E
PATRIA, CIENCIA Y TRABAJO
"2013, 50 Aniversario Luctuoso del Poeta Heriberto Enríquez"

DR. EN ED. RAYMUNDO OCAÑA DELGADO
SUBDIRECTOR ACADÉMICO DEL CU UAEM ZUMPANGO

CENTRO UNIVERSITARIO UAEM ZUMPANGO
SUBDIRECCIÓN ACADÉMICA

c.c.p. Jefatura de Control Escolar
c.c.p. Minutario
ROD/yjms*

UAEM | Universidad Autónoma del Estado de México

CUZ

Martes, 14 de enero del 2014
Oficio No. **TIT/014/14**

C. RODOLFO IVÁN FLORES DE LA TORRE
PASANTE DEL PE DE ICO
DEL CENTRO UNIVERSITARIO UAEM ZUMPANGO
P R E S E N T E

Por este conducto, la Subdirección Académica del Centro Universitario UAEM Zumpango informa a Usted que, una vez concluidas las etapas de desarrollo y revisión del trabajo de titulación bajo la modalidad de tesis denominado: **"ANÁLISIS COMPARATIVO DE ÁRBOLES DE DECISIÓN Y MÁQUINA DE VECTORES SOPORTE PARA CONJUNTOS DE DATOS DE DIABETES Y HEPATITIS"** se autoriza su **IMPRESIÓN** para posteriormente ser sustentado en evaluación profesional.

Sin otro particular, se despide de Usted.

A T E N T A M E N T E
PATRIA, CIENCIA Y TRABAJO
"2014, 70 Aniversario de la Autonomía ICLA-UAEM"

DR. EN ED. RAYMUNDO OCAÑA DELGADO
SUBDIRECTOR ACADÉMICO DEL CU UAEM ZUMPANGO

CENTRO UNIVERSITARIO UAEM ZUMPANGO
SUBDIRECCIÓN ACADÉMICA

c.c.p. Departamento de titulación del CU.
c.c.p. Minutario
ROD/yjms*

Resumen

En este trabajo se presenta la comparativa de desempeño de dos tipos de algoritmos de clasificación, que fueron aplicados a conjuntos de datos de las enfermedades Diabetes y Hepatitis. Los conjuntos de datos relacionados con las enfermedades por lo general son desbalanceados, es decir, contienen muchos más objetos de un tipo que de otro, por lo que la mayoría de los métodos de clasificación presentan problemas de desempeño, ya que intentan generalizar el modelo subyacente en los datos y no fueron desarrollados para conjuntos de datos con características particulares.

Los tipos de algoritmos comparados son los árboles de decisión y las máquinas de soporte vectorial. El primero produce un modelo que puede ser interpretado por un experto humano, tiene un tiempo de entrenamiento pequeño y puede obtener resultados con valores altos de precisión en la clasificación. El segundo, produce modelos compactos, que alcanzan buena precisión de clasificación, tienen un poder de generalización mayor respecto a otros algoritmos, pero su tiempo de entrenamiento es computacionalmente alto. Los algoritmos comparados fueron árboles de decisión (*ADTree* y C4.5) y máquina de soporte vectorial (SVM), entrenada con el algoritmo de optimización mínima secuencial (SMO). Este último es un método para resolver los problemas de programación cuadrática, y es ampliamente utilizado para acelerar el entrenamiento de las máquinas de soporte vectorial.

Los resultados presentados incluyen mediciones y comparaciones de precisión de clasificación, errores absolutos y otras medidas como la Kappa estadística. En los experimentos realizados utilizando la plataforma Weka, se variaron los parámetros de los árboles de decisión y de la máquina de vectores soporte, para observar su efecto en el desempeño. Los resultados mostrados pueden servir de guía para la aplicación de los algoritmos *ADTree*, C4.5 y máquinas de soporte vectorial en el área médica.

Abstract

In this work, a comparison of the performance of two types classification of algorithms is shown. The algorithms were applied to Diabetes and Hepatitis data sets. In general, the data sets that contain information about illness are skewed, i.e.; the number of elements of a class is considerably larger than the other classes. Most classification methods are unsuitable for these data sets.

The type of algorithms are decision trees and support vector machines. The former produces a model that can be intepreted by a human expert. That algorithm is trained quickly, and it achieves a high classificacion accuracy. The second method, support vector machine, produces a compact model, it achieves an elevated classification accuracy; it has a good generalization capability. However, support vector machines usually have a large training time. The algorithms compared are ADTree, C4.5 (both of them are decision trees) and the support vector machine using Squential Minimal Optimization (SMO), which is a method widely used to train SVM.

The results presented throughout this work include comparative of classification accuracies, absolute errors and other statistical measures, such as Kappa. The experiments were executed using Weka.

Contenido

Lista de Tablas

Lista de Figuras

Capítulo 1

Introducción

En México, dos de las enfermedades que provocan más muertes en el país son la Diabetes y la Hepatitis [1, 2]. En el área de computación, el diagnóstico automático de enfermedades se realiza mediante métodos de extracción de conocimiento conocidos como clasificadores. Los clasificadores son métodos de aprendizaje supervisado, esto significa que dichos métodos requieren de conjuntos de datos etiquetados para poder construir un modelo matemático.

Este trabajo realiza una comparativa del desempeño entre dos de los tipos de algoritmos de clasificación más importantes actualmente, los árboles de decisión y las máquinas de soporte vectorial (SVM), aplicados a conjuntos de datos reales de las enfermedades Diabetes y Hepatitis. Debido a que fue imposible obtener datos de pacientes en hospitales de la región, se usaron datos disponibles públicamente en el repositorio Web UCI (Universidad de California Irvine).

A lo largo de este trabajo, se describe la patología de las enfermedades, los síntomas, los órganos más afectados y las posibles causas. También se describen las herramientas utilizadas, las características y formato de los conjuntos de datos, y se proporciona una explicación de los clasificadores utilizados. Dentro de los algoritmos de árboles de decisión se comparan los algoritmos *ADTree* y C4.5, los desempeños son diferentes ya que cada uno posee distintos parámetros y genera los modelos usando estrategias distintas.

El algoritmo *ADTree* fue realizado por Freund Y. y Mason L. en 1999. Este algoritmo construye árboles de decisión alternos y elige el que produce la mejor precisión de clasificación. Por cada iteración construye 3 nodos uno de división y dos de

predicción utilizando medidas de entropía. El segundo árbol de decisión utilizado para la comparación es el C4.5, probablemente este sea el más utilizado en la literatura. C4.5 fue desarrollado por Ross Quinlan en 1993, realiza técnicas de poda para eliminar datos que causan ruido o que no son valiosos para realizar la clasificación. En cada iteración, el algoritmo calcula el atributo más significativo y se elige como parámetro de decisión para dividir en dos partes el conjunto de datos, su criterio se basa en medidas de entropía.

Las SVM fueron propuestas por Vladimir Vapnik y sus colaboradores en 1992. Este método utiliza un hiperplano de margen máximo para dividir el espacio de entrada en dos regiones, cada una de las cuales corresponde a una clase de objetos. Las SVM en general transforman el espacio de entrada en otro espacio de una dimensión más alta para poder resolver los problemas de clasificación no linealmente separables. Este método además de ser aplicable a problemas de clasificación puede usarse también para problemas de regresión.

Para realizar la clasificación por SVM se utilizó el algoritmo de entrenamiento llamado optimización mínima secuencial (SMO). Este fue desarrollado por John Platt en 1998, y actualmente es ampliamente utilizado. Su funcionamiento consiste en resolver subproblemas de optimización cuadrática de manera analítica.

Los algoritmos de clasificación fueron comparados usando la herramienta Weka[1], que es un software de código abierto para realizar tareas de minería de datos. El desempeño de cada algoritmo aplicado a cada conjunto de datos fue llevado a cabo variando los parámetros más importantes, la intensión fue observar como varía la precisión de clasificación. Entre las mediciones más importantes que se realizaron en cada algoritmo fueron la precisión de clasificación, promedio del error absoluto, tiempo de entrenamiento, y otras medidas estadísticas. Para el caso de los árboles de decisión, se tomó en cuenta el número hojas y el tamaño del árbol.

[1] Weka: Waikato Environment for Knowledge Analysis, es una herramienta para realizar minería de datos, desarrollada por la Universidad de Waikato en Nueva Zelanda en el año de 1999. Descargable gratuitamente en http://www.cs.waikato.ac.nz/ml/weka.

1.1. Motivación

La Diabetes y la Hepatitis son dos enfermedades que ocupan uno de los primeros lugares de muerte en el país [1, 2]. La Diabetes ocupa el primer lugar de mortalidad en México y la Hepatitis el tercer lugar. Por otro lado, la minería de datos es un área de la inteligencia artificial y que ha sido aplicada en muchas áreas, incluyendo la médica. Dentro de los algoritmos de minería de datos existe la clasificación, que en el área médica pueden ser de gran ayuda, por ejemplo, para realizar diagnósticos de enfermedades para apoyar las decisiones de los médicos.

La motivación principal de este trabajo es realizar una comparativa en el desempeño de algoritmos de clasificación aplicados en el área de medicina, con la finalidad de ofrecer un panorama sobre cúal algoritmo proporciona mejores resultados, y cúal es el efecto de sus parámetros.

1.2. Planteamiento del Problema

La problemática de esta investigación, se centra en comparar el desempeño de dos de los algoritmos más utilizados en el área de minería de datos en conjuntos de datos relacionados con la salud: Árboles de decisión y máquinas de soporte vectorial. ¿Cuál de estos algoritmos de clasificación es el más adecuado para este tipo de conjunto de datos?, en este trabajo se realizan pruebas con conjuntos de datos reales y se mide su precisión de clasificación, desviaciones estándar y el efecto de los principales parámetros de cada algoritmo.

1.3. Justificación

La comparación de algoritmos de clasificación con respecto a su tiempo de entrenamiento, precisión de clasificación, desviación estándar de precisión, error absoluto, resulta importante para el usuario de este tipo de algoritmos, debido a que le permite tener un panorama amplio del desempeño y de esta manera decidir cuál utilizar en una aplicación específica. En este trabajo, la aplicación se centra en el área médica, por lo que se utilizan conjuntos de datos de dos de las enfermedades que producen un mayor número de muertes en el país, la Diabetes y la Hepatitis.

1.4. Alcance del trabajo

Debido a que no se pudieron conseguir datos de personas en México que padecen Diabetes o Hepatitis, se decidió utilizar los conjuntos de datos de dichas enfermedades del repositorio UCI de la Universidad de California. Estos y otros conjuntos de datos se encuentran publicados en la página Web de esa universidad y también en varios sitios Web. Para analizar los conjuntos de datos, se utilizó la versión de desarrollador de Weka, una herramienta de código abierto. También se desarrolló una interfaz de usuario gráfica en la que se agregaron tres algoritmos de clasificación incluidos en Weka.

Se presentan resultados de desempeño de cada uno de los algoritmos como la precisión de clasificación, el tiempo de entrenamiento y el promedio del error absoluto. También se hace una breve discusión sobre cuál sería el algoritmo más adecuado para cada conjunto de datos.

1.5. Metodología

En este trabajo se usarán conjuntos de datos disponibles públicamente en el repositorio UCI *Machine Learning*[2] para el análisis empírico de los algoritmos de aprendizaje automático, debido a que son utilizados ampliamente en la literatura y además por la limitación de conseguir datos de pacientes reales. Para realizar la comparativa, primero se investigaron los algoritmos de clasificación *ADTree*, C4.5 y SVM. También se realizó una investigación sobre el uso de la plataforma de código abierto Weka para poder realizar las comparaciones.

La comparación de desempeño de algoritmos es realizada en condiciones idénticas para cada uno de ellos, el método para validar los resultados es el de validación cruzada, el más utilizado en el área de minería de datos, reconocimiento de patrones y aprendizaje automático.

1.6. Objetivo

El objetivo general de este trabajo es realizar una comparativa de desempeño de dos tipos de algoritmos para clasificación automática de patrones, aplicados a conjuntos datos de las dos principales enfermedades que causan la muerte en México; Diabetes

[2] http://courses.soe.ucsc.edu/

y Hepatitis. Los algoritmos a comparar son los árboles de decisión *ADTree*, C4.5 y la máquina de soporte vectorial (SVM).

1.6.1. Objetivos específicos

Los objetivos específicos para este trabajo son los siguientes:

1. Realizar una investigación documental sobre las enfermedades de Diabetes y Hepatitis, para describir los principales órganos afectados, causas que las originan y los síntomas principales de estas enfermedades.

2. Explicar las principales características de los conjuntos de datos en formato .arff para las enfermedades del objetivo específico anterior.

3. Investigar el funcionamiento de los algoritmos de clasificación *ADTree*, C4.5 y Máquinas de Soporte Vectorial, con la finalidad de comprender su funcionamiento.

4. Describir las principales características de la herramienta para extracción automática de conocimiento llamada Weka, con la finalidad de utilizarla en los experimentos con los conjuntos de datos elegidos.

5. Comparar el desempeño de los algoritmos de clasificación *ADTree*, C4.5 y SVM, aplicados sobre los conjuntos de datos referentes a la Diabetes y la Hepatitis, con el objetivo de realizar un análisis sobre el desempeño de cada uno de los algoritmos.

1.7. Organizacion del documento

El resto de este trabajo está organizado de la siguiente manera:

En el capítulo 2 se presentan los fundamentos con una breve historia del almacenamiento de datos con las primeras computadoras creadas por el hombre, definiendo que es una base de datos, los principales sistemas gestores de bases de datos para la manipulación de los datos almacenados. Después, se aborda la manera en la que se lleva acabo la extracción de conocimientos de conjuntos de datos de manera digital con las principales tareas de minería de datos, describiendo algunos de los algoritmos de clasificación y por último, se hace una descripción de las principales características de la herramienta Weka que es el software que se utilizará en este trabajo para el

análisis de datos.

En el capítulo 3 se hace una descripción general de las enfermedades de Diabetes y Hepatitis. Se dan a conocer los orígenes de estas enfermedades, los principales órganos afectados, las posibles causas que las originan y los principales síntomas de cada enfermedad. También se describen los conjuntos de datos con formato .arff de Diabetes y Hepatitis, mostrando los atributos y explicando el significado de cada uno.

En el capítulo 4 se muestran las configuraciones de los algoritmos de clasificación utilizados, que son árboles de decisión y máquina de soporte vectorial. Dentro de árboles de decisión, se presentan dos algoritmos diferentes: *ADTree* y C4.5. Se describe su funcionamiento y los principales parámetros para configurar en la herramienta Weka. De igual manera, se realiza una revisión de las máquinas de soporte vectorial. Esto sirve de apoyo para realizar los experimentos y posteriormente comparar el desempeño de los algoritmos. Esto permite dar una idea de cuál sería el más adecuado para cada conjunto de datos. Se presenta una implementación propia de una interfaz gráfica de usuario desarrollada para usar estos clasificadores, y, por último, se presentan las conclusiones generales y los trabajos futuros.

Capítulo 2

Fundamentos

En este capítulo se realiza una breve reseña histórica del almacenamiento de datos, continuando con una descripción de las base de datos, tipos y sistemas gestores. También, se hace una revisión de la minería de datos, que consiste en extraer información oculta de conjuntos de datos almacenados digitalmente. Las principales tareas de la minería de datos son descritas de manera resumida.

En este trabajo se utiliza Weka como herramienta para análisis de datos, por lo que se da una descripción de las principales características de este software.

2.1. Almacenamiento de datos

Los primeros signos que demuestran un almacenamiento de conocimiento o de información guardados intencionalmente por el hombre, se remontan a la época de la prehistoria, cuando los datos eran plasmados por medio de pinturas rupestres en cuevas o cavernas y donde mostraban sus hazañas de caza, recolección de alimentos y animales para que alguien pudiera verlas, o simplemente para poder expresar ideas. A medida que el hombre fue evolucionando en el manejo de herramientas, le fue posible labrar en piedras algunos glifos o jeroglíficos que almacenaban información interesante de viajes, historias, profecías en el caso de los Mayas e historias de faraones en el caso de Egipto.

Uno de los 4 grandes inventos de los chinos junto con la brújula, la pólvora, y la imprenta, fue el papel, por siglos de años se ha conocido el nombre del inventor quien es Cai Lun (?-121), de la dinastía Han [3], ya que este invento permitió almacenar de manera más fácil la información de científicos, poetas, pensadores, etc. al

almacenar varias hojas se formaron libros con información más completa sobre un tema.

Actualmente la información que se tiene es extensa y variada, proveniente de diferentes fuentes como científicos, artistas, médicos, ingenieros, etc. el a cúmulo de información dio pie a la construcción de bibliotecas, hemerotecas, filmotecas, fototecas, etc. que son lugares con acceso al público en general para la consulta y proporción de información clasificada en secciones.

Con el paso del tiempo, los avances en la ciencia y tecnología, permitieron al hombre llegar a desarrollar dispositivos de almacenamiento. Las primeras computadoras creadas en 1946, fueron la ENIAC [4] (por sus siglas en inglés, *Electronic Numerical Integrator And Computer*), por el ejército de los Estados Unidos de América y la Colossus [5] por el Reino Unido para descifrar códigos Alemanes en la segunda guerra mundial. Estas computadoras no tenían capacidad de almacenamiento temporal, es decir no contaban con una memoria de acceso aleatorio.

La ENIAC fue sustituida por la EDVAC [6] (*Electronic Discrete Variable Automatic Computer*), que utilizó el primer programa con instrucciones para almacenar datos, esto fue fundamental para la evolución de las computadoras. Uno de los primeros dispositivos de almacenamiento en perfeccionar fue el tubo *selectron* o el tubo Williams [7], que almacenaba información de dos estados, mismos que podían escribirse y leerse en cualquier momento. Estos estados eran almacenados en un conjunto de posiciones que contenían bits denominado código binario. Al principio se permitía almacenar un sólo dígito, conforme el paso del tiempo se pudo llegar a capacidades de almacenamiento de 2048 bits, usadas como una memoria de acceso aleatorio. Esto fue el antecedente para las computadoras actuales, que contienen un tipo de memoria similar, conocida comúnmente como RAM (*Random Access Memory*).

La evolución de los dispositivos de almacenamiento fueron avanzando en el desarrollo de grandes cantidades de medios magnéticos, cintas, medios ópticos etc. con una gran aportación de almacenamiento o espacio para guardar datos. Después del tubo de Williams se desarrollaron algunos dispositivos como son las tarjetas perforadas y las cintas perforadas [8] que consisten en el almacenamiento de datos en Tablas y en tiras de papel muy largas con agujeros codificadas y decodificadas por computadoras programadas en la década de 1970, la cinta magnética que fue patentada por el ingeniero alemán Fritz Pfieumer [9] en el año 1928, pero no fue hasta el año 1951 que

fue utilizada para la grabación de datos. La capacidad de la cinta era de varios cientos de Kilo Bytes (KB) y fue utilizada para la UNIVAC I (*Universal Automatic Computer I*) [10].

Un medio de almacenamiento relacionado con las cintas fueron los discos de acetato, que junto con los discos de vinilo fueron utilizados para grabar música inventado por Peter Golmark, [11] el 21 de junio de 1948. El disquete que fue elaborado por los laboratorios de la empresa IBM *(International Business Machines)* en San José California, por David Noble [12] en el año de 1967 con capacidades de 100 KB de capacidad llamado *floppy* por su flexibilidad en aquel entonces este era un *floppy* de 8", con el paso del tiempo se desarrolló el disco flexible 5 $\frac{1}{4}$" en el año de 1978 con capacidades desde 360KB hasta 1.2 Mega Bytes (MB), a mediados del año de 1979 se creó ST-506 una unidad de disco duro de 6MB de capacidad y no fue hasta el año de 1981 que la empresa Sony introduce al mercado los discos flexibles de 3.5 pulgadas con capacidades desde 720KB hasta los 2.88MB.

Con la invención del circuito integrado en el verano de 1958, desarrollado por Jack Kilby[13], se desarrollaron memorias USB (*Universal Serial Bus*), con gran capacidad de almacenamiento, superando los 100GB de memoria y desarrollada en 1998 por IBM [14]. Estas sustituyeron a los disquetes y permitieron el desarrollo de lectores ópticos como los CD's y DVD's diseñados para leer datos por medio de un láser.

El sector empresarial se percató que era necesario almacenar sus datos de manera que pudieran ser fácilmente recuperados, por lo que rápidamente se comenzaron a utilizar sistemas electrónicos para este fin, sustituyendo los archivos en papel. Estos últimos tenían muchas desventajas, por ejemplo:

1. Ocupan una gran cantidad de espacio físico. En las oficinas, algunas áreas generan mucha información impresa, por lo que las pilas de papel se convierten en un estorbo para las personas que ahí trabajan. Ejemplos de esta situación son los historiales clínicos de personas que están afiliados en una institución de salud, como hospitales. Otro ejemplo son las instituciones de educación superior, donde se almacenan expedientes de cada alumno, de diferentes generaciones y licenciaturas.

2. Daño al medio ambiente. Como bien se sabe, el papel es extraído de los árboles, tan sólo para producir una tonelada de papel se necesitan 15 árboles, 7800 kilowatts

(kw) de energía eléctrica y 100,000 (cien mil) litros de agua [15]. Además, al quemar el papel, se contamina el medio ambiente.

3. Pérdida de tiempo. Los historiales o los expedientes que se requieran para ser consultados deben buscarse manualmente dentro de los archivos divididos por bloques y ordenados por alfabéticamente en la mayoría de los casos. El extravio de la información es común, lo que ocasiona pérdida de tiempo.

4. Falta de respaldo de información. Debido a que el almacenamiento físico consume mucho espacio, el respaldo o duplicación de documentos no es fácil.

El aporte que ha tenido la informática, ha permitido que se generen documentos que son almacenados en forma de archivos digitales. Las empresas, al ver las desventajas que ocasionaba el papeleo decidieron utilizar estos tipos de medios informáticos para la protección de la información y disminuir los problemas que se tenían. Sin embargo, no basta solamente con registrar o almacenar los datos en un procesador de texto o en hojas de cálculo electrónicos, sino que es necesario darle un orden para la manipulación de estos, de lo contrario estarían desordenados dentro de su medio de almacenamiento.

2.2. base de datos

Para tener un control de la información almacenada y darle una mejor organización, se crearon las base de datos, que han eliminado, si no en su totalidad pero si en gran cantidad, las desventajas del almacenamiento de documentos físicos y la recuperación de información o de datos importantes de empresas, instituciones educativas, negocios y datos personales, entre otros. Así, los datos son de los bienes más valiosos de las empresas, por lo que es necesaria la protección de los mismos. Las base de datos pueden almacenar, recuperar, organizar, cifrar y manejar información en forma rápida y precisa gracias a las tecnologías actuales.

Una base de datos es un conjunto estructurado de datos que representa entidades y sus interrelaciones. La información contenida debe ser verídica ya que puede ser utilizada de forma compartida por varios usuarios de diferentes tipos [16]. Otros autores, como [17], la definen como un fondo común de información almacenada en una computadora, para que cualquier persona o programa autorizado pueda acceder a ella, independientemente de su procedencia y del uso que haga.

En resumen, se puede decir que una base de datos proporciona a los usuarios el acceso de la información de manera pública o privada, es decir, existen personas o programas que pueden acceder a ella para sólo visualizar, ingresar datos y otras con los derechos de eliminar, actualizar, ingresar, dependiendo para lo que fue creada. La información que se presenta puede ser local para que se pueda ver en un sólo equipo o de manera distribuida que puede acceder a ella a través de la red.

La mayoría de las empresas tienen diferentes tipos de información de las cuales existen archivos privados y de valiosa protección y que hay que centralizar, es decir, existe una persona identificable dentro de la empresa que tendrá esta responsabilidad central sobre los datos. A esta persona se le conoce como administrador de datos cuyas funciones son determinar que datos almacenar, mantener políticas para indicar quien puede realizar operaciones y manejar esos datos una vez almacenados para lograr una gran seguridad en ellas [18].

Dentro de las base de datos existe el concepto de rol. Un rol está relacionado con los permisos que un usuario tiene sobre una base de datos. Dos roles importantes son el de administrador de datos y el administrador de base de datos. El primero proporciona las políticas y los permisos de los usuarios, mientras que el segundo tiene como función crear la base de datos, conforme a las políticas del administrador de datos, de una manera sencilla de entender, y contener controles técnicos para poder llevar a cabo las decisiones del administrador de datos y que el sistema opere de manera segura.

Una de las ventajas de las base de datos es que varios usuarios pueden acceder a ellas simultáneamente, a través de la Internet, que es una red de redes mundial que tiene almacenada en distintos servidores grandes cantidades de información por medio de diferentes programas que permiten, definir, construir, y manipular base de datos y posteriormente acceder a los datos de forma rápida y estructurada. A estos programas se les denomina sistemas gestores de base de datos (SGBD). Estos sistemas son muy utilizados en las empresas para gestionar los datos más importantes de las actividades realizadas diariamente. Prácticamente en la actualidad la mayoría de empresas e instituciones tienen base de datos para el mejor control de la información.

Los SGBD no manipulan información sino registros. Existen diferentes tipos de SGBD según el criterio de uso. El criterio más específico es según el modelo de base de datos como relacional, orientado a objetos, jerárquico y en red, estos tipos de

programas ofrecen facilidades en el manejo de grandes cantidades de datos por medio de interfaces entre la base de datos y el usuario conjuntado con las aplicaciones de uso [19].

Los SGBD más utilizados son: SQL (*Structured Query Language*) [20]. Fue desarrollado por IBM en 1970 y en 1986 fue estandarizado por ANSI (*American National Standards Institute*) como ANSI SQL [21], el cual es un estándar para acceder y manipular base de datos, así como, hacer consultas, recuperar datos, insertar, actualizar, eliminar registros, crear nuevas base de datos, crear tablas y otorgarles permisos de cualquier tipo, crear vistas, etc. Para este tipo de SGBD es necesario utilizar comandos como *SELECT, UPDATE, DELETE, INSERT, WHERE.*

Para la utilización de un SGBD de una base de datos en red son necesarios algunos sistemas gestores de base de datos relacionales como MS Access, SQL Server, My SQL, que puedan comunicar varias computadoras dentro de una red y tener un lenguaje de script del lado del servidor.

SQL fue la base en el desarrollo de sistemas de base de datos más modernos como MS SQL Server, IBM DB2, Oracle, My SQL y Microsoft Access.

Las ventajas de utilizar los SGBD son [19]:

- Ser de fácil manejo para el usuario: La información que se requiera por los usuarios debe ser de manera amigable de modo que sea muy fácil de usar y entender para no confundir al usuario.
- Evitar redundancia: Eliminar datos repetidos para no almacenar y ocupar espacios de memoria, que puede ocupar otra información, el SGBD gestiona los datos concurridos para resolver ese asunto.
- Proporcionar la integridad de los datos: La información no debe ser accedida por usuarios que no tienen autorización, para que los datos no sean manipulados y pierdan su veracidad, pero mantener la integridad no es posible al cien por ciento ya que existen errores humanos que la persona autorizada se le pase un dato o un error de dedo, o en un fallo del sistema hay que tener también una recuperación de fallos.
- Dar seguridad: Brindar mecanismos de identificación para que la persona autorizada sea la única de manipular la información y los usuarios que no tienen ciertos privilegios cumplan su función que les corresponde.

- Mantener la fiabilidad del sistema: Tener un respaldo de la información en diferentes servidores o copias de seguridad para tener un control y evitar el extravío de datos o influir en la corrupción de la información ya que hoy en día las filtraciones de información son más notables.

Pero como cualquier sistema, los SGBD tienen desventajas, por ejemplo:

- Tamaño: En algunos tipos de SGBD requieren para su instalación mucho espacio en disco y más cuando se trata de grandes volúmenes de información y de memoria para su buen funcionamiento.
- Costo: Los costos son variados dependiendo de la complejidad y el entorno que ofrece, además, hay que considerar las licencias para poder ocupar dicho sistema.
- Vulnerable a fallos: No existe un sistema perfecto, algunos sistemas pueden tener errores ya que fue programado por el ser humano, pero se van dando cuenta de las fallas cuando se van utilizando.

A continuación se explican los principales paradigmas de arquitecturas de base de datos.

2.2.1. Modelos de base de datos

Existen diferentes tipos de base de datos pero dependerá de diferentes modelos de datos, en la actualidad el modelo relacional es el más utilizado [22] aunque existen otros que se explicarán en las siguientes subsecciones.

2.2.1.1. Modelo relacional

El modelo relacional es un modelo de datos que tiene en cuenta una estructura que permita representar la información del mundo real, la manipulación mediante operadores de actualización y de consulta de los datos y la integridad que por medio de reglas se deben especificar las condiciones que los datos deben cumplir [23].

Este tipo de modelo fue establecido por E.F Codd en los años 1969 y 1970, el objetivo principal del modelo relacional es que la base de datos sea vista por el usuario como una estructura lógica que consiste en relaciones de registros y objetos y no en una estructura física de implementación como métodos de acceso, estructura de archivos, etc. La representación lógica de las entidades y sus relaciones se representan en tablas bidimensionales.

Se llamará registro o tupla a cada fila de la tabla y campo o atributo a cada columna de la tabla. Uno de los requisitos de las tablas es que no deben haber tuplas repetidas. Una clave será un atributo o conjunto de atributos que identifique de forma única a una tabla como en la Figura 2.1.

Para manejar dichas tablas se deben utilizar operaciones clásicas de la teoría de conjuntos como unión, intersección, diferencia, y producto cartesiano así como operaciones específicas del modelo relacional como selección, proyección, unión y división.

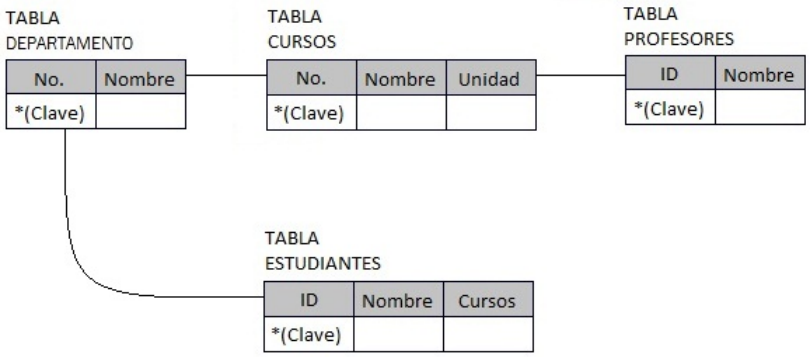

Figura 2.1: Esquema del modelo relacional

2.2.1.2. Modelo orientado a objetos

Los métodos orientados a objetos, se enfocan principalmente en el modelado de un sistema en términos de objetos, que básicamente consiste en el encapsulamiento de código de leguajes de programación orientado a objetos como Java, C/C++ y Visual Basic.NET, un objeto poseé almacenado atributos o datos sobre sí mismo. Durante las actividades de desarrollo se utilizan diferentes herramientas de modelado [24].

- Diagramas de clases: Sirven para describir los componentes necesarios de un sistema, mostrando sus clases, atributos y las relaciones de asociación entre ellas como se aprecia en la Figura 2.2.

- Diagramas de casos de uso: Sirve para representar el comportamiento de los diferentes casos que pueden presentarse en cada nivel del sistema creado llevando al usuario a seguir diferentes pasos de usos para realizar una tarea específica.

- Diagramas de transición de estado: Describen los cambios de estado en los objetos y su ciclo de vida.

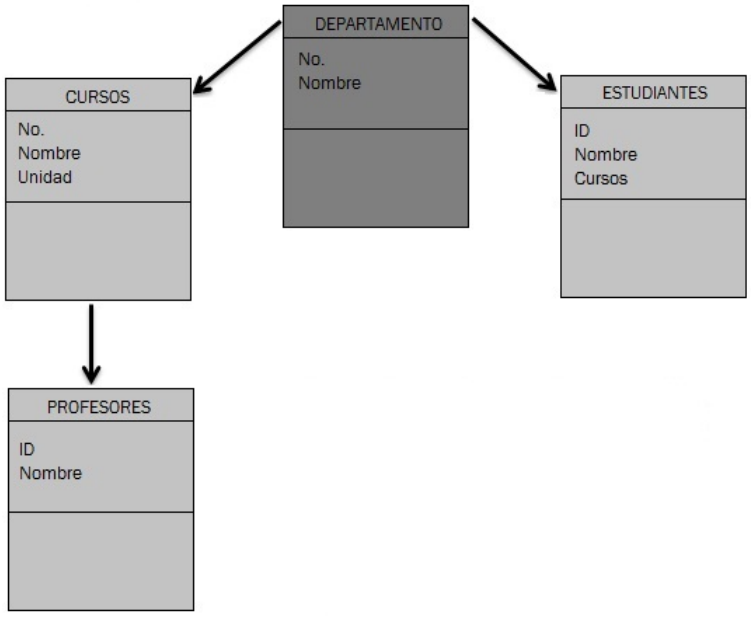

Figura 2.2: Esquema del modelo orientado a objetos

2.2.1.3. Modelo jerárquico

El modelo de base de datos jerárquico está basado en una estructura de árbol compuesto de un segmento raíz, segmento padre y segmentos hijos. El segmento es el equivalente a un tipo de registro de archivos, además ilustra un conjunto de relaciones uno a uno (1:1) y uno a muchos (1:M) entre un padre y sus hijos. Como se puede ver en la Figura 2.3, este tipo de modelo utiliza una secuencia jerárquica o ruta preordenada para navegar por sus estructuras, misma que siempre inicia por el lado izquierdo del árbol [25]. Los registros están dispuestos en forma de árbol y no pueden existir ciclos, cuando se borra un registro padre se borran todos sus hijos.

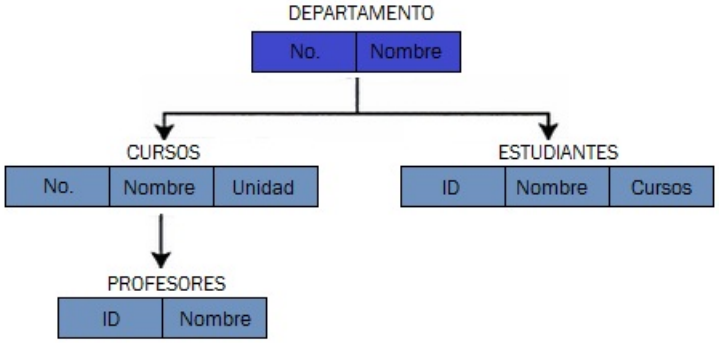

Figura 2.3: Esquema del modelo jerárquico

2.2.1.4. Modelo en red

En este tipo de modelo, las entidades se organizan en un grafo, donde se puede tener acceso a algunas entidades a través de varios caminos. No existe una jerarquía ni siquiera un análisis posterior [26].

Tiene dos características principales [22]:

1. La estructura principal consiste en dos tipos de registros (propietario del conjunto y miembro) en el que el propietario de la clase es el padre y el miembro es el hijo, sólo existe un propietario y uno o varios miembros, como se observa en la Figura 2.4.

2. Un registro miembro se puede asociar con uno o varios propietarios.

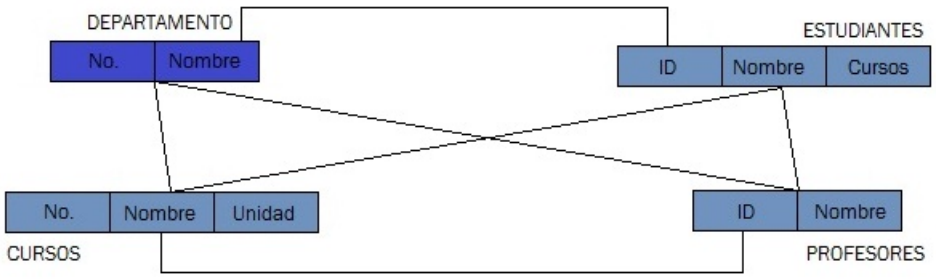

Figura 2.4: Esquema del modelo en red

2.2.1.5. base de datos distribuidas

Está basado en el modelo relacional pero los datos están almacenados en varias computadoras que se comunican a través de Internet o en una red de área amplia. Cada computadora o nodo contiene una parte o toda la base de datos. Quizás, esté fragmentada con parte de la información en cierto punto de la red o duplicado en cada computadora, como se observa en la Figura 2.5.

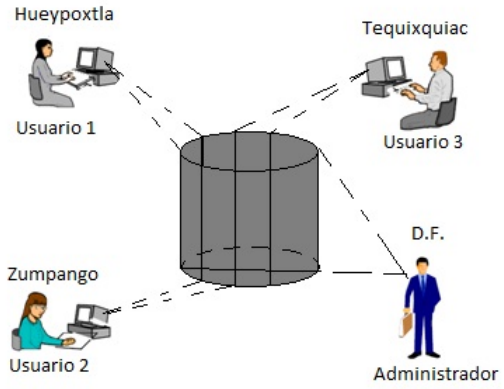

Figura 2.5: Esquema de las base de datos distribuidas

2.3. Recuperación de información en las base de datos

Las bases de datos son ampliamente utilizadas para almacenamiento masivo de registros, sin embargo, los datos por sí mismos no poseen un valor agregado si no se les da un significado, o si no les utiliza para obtener información valiosa de ellos. Una base de datos a la que no se le emplea más que para almacenamiento, se convierte simplemente en un cementerio de datos.

Como ya se mencionó anteriormente, existen diferentes tipos de SGBD que hacen posible la iteración entre el usuario y la base de datos. Esta interacción es un proceso complicado, que involucra la comunicación de diversas partes del SGBD con la interfaz de usuario, y la utilización de un lenguaje estándar para consultar y/o actualizar los datos. Un resumen de este proceso se presenta a continuación.

Para el diseño de una arquitectura de un sistema de base de datos, se empieza por entender que hay una separación entre los programas y los datos, además, deben existir múltiples vistas de usuarios para la manipulación de los mismos. Para esto se creó una arquitectura estandarizada por ANSI/SPARC por sus siglas en inglés (*American National Standards Institute*)/(*Standards Planning And Requirements Committee*) propuesta en 1975 [27], que establece tres niveles de abstracción[22]:

1. Interno: Este nivel es el más abstracto porque es donde se encuentra toda la información almacenada, y posee un esquema interno en donde se describe en detalle las estructuras de datos a nivel físico como por ejemplo: organización física de los archivos, modo de acceso a los registros que lo componen, tipos de

registros, campos, etc.

2. Conceptual: Este nivel establece una conexión entre el nivel interno y el externo, describiendo que datos están almacenados en la base de datos y las relaciones que existen entre ellos describiéndolos en un esquema conceptual por ejemplo: entidades, tipos de datos, relaciones, operaciones de los usuarios y restricciones.

3. Externo: Este nivel es el más alto de abstracción ya que establece la relación de la base de datos con el usuario, utilizando varios esquemas externos o de vistas, mostrando sólo una parte de la base de datos, es decir, muestra una porción que le interese a un grupo determinado de usuarios ocultándoles el resto de la base de datos.

Visto de otra manera, los niveles anteriores se pueden entender como se muestra en la Figura 2.6.

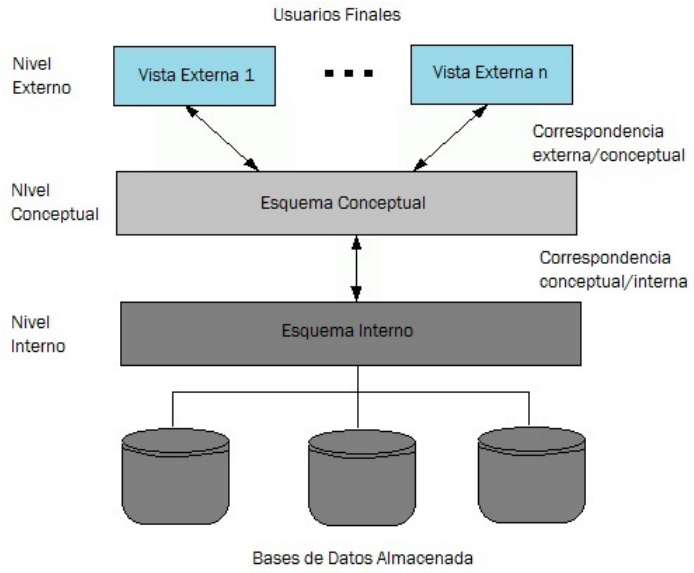

Figura 2.6: Esquema de niveles de abstracción de un SGBD

Los componentes principales que conforman un SGBD y las tareas que realizan cada uno de ellos son los siguientes [28, 22]:

1. Gestor de Archivos: Se encarga de mantener libre un espacio dentro de la unidad de almacenamiento para las estructuras de datos para representar la información en la base de datos.

2. Gestor de la base de datos: Se encarga de la manipulación de los datos almacenados en las base de datos y los programas de aplicación por medio de una interfaz.

3. Procesador de consultas: El procesador se encarga de traducir las sentencias inscritas en un lenguaje de bajo nivel para que el SGBD las entienda y las ejecute.

4. Compilador de DDL (*Data Definition Languaje*): Convierte las definiciones escritas en DDL en un conjunto de Tablas que contienen metadatos, estas Tablas se almacenan después en el diccionario de datos, además genera índices, relaciones, disparadores (*triggers*), etc.

5. Pre compilador de DML (*Data Manipulation Languaje*): Convierte las sentencias en DML incrustadas en un programa de aplicación a un lenguaje principal para que el procesador de consultas genere el código apropiado para insertar, borrar, modificar y consultar los datos de una base de datos.

6. Archivos de datos en disco: Almacena físicamente la base de datos en si.

7. Diccionario de datos: Almacena metadatos de estructuras de la base de datos.

8. Archivos índices: Permiten el acceso rápido a los datos. Al igual que el diccionario, están incluidos físicamente en los archivos de datos.

2.4. Minería de datos o extracción de nuevo conocimiento a partir de datos

Las base de datos son utilizadas en todo el mundo para la recopilación y la organización de los datos. El almacenamiento de datos ha ido evolucionando muy rápido, desde los años 1960's, cuando se desarrollaron las tecnologías como los sistemas de procesamiento y de almacenamiento para el control de datos, hasta el desarrollo de base de datos más sofisticadas y poderosas de la actualidad.

Los SGBD tienen como funciones las de gestionar datos repetidos, brindar seguridad en la base de datos, mantener la integridad de los datos y mantener la comunicación con el usuario. Sin embargo, los SGBD solamente son capaces de presentar información que se encuentra almacenada, y no permiten extraer conocimiento oculto en los datos. A cualquier usuario de una base de datos que requiera hacer un análisis de los registros, le resultaría difícil realizarlo de manera manual, esto debido al rápido crecimiento del número de datos. Actualmente, la cantidad de información supera la capacidad de comprensión humana.

Para la automatización del análisis de datos, se han creado métodos para la extracción del conocimiento, que utilizan grandes cantidades de datos. Algunos ejemplos específicos son los sistemas expertos y la minería de datos (MD), que engloba varias técnicas. La MD es el resultado de la evolución natural de la tecnología de base de datos.

La MD realiza análisis de datos para descubrir patrones o relaciones entre ellos, con la finalidad de apoyar a la toma de decisiones, estrategias de negocios, investigación científica y médica, entre otros. Es utilizada para descubrir patrones ocultos en grandes conjuntos de datos, con la finalidad de comprenderlos y aplicarlos en tareas tales como la detección de fraudes, análisis de mercado, control de producción, descubrimiento de eventos anómalos o predicción. En resumen, la minería de datos se refiere a la extracción de conocimiento de grandes cantidades de datos.

Para llevar acabo la extracción del conocimiento de datos, se lleva acabo un proceso de pasos de forma secuencial como son [29]:

1. Eliminación datos. Donde se elimina el ruido y datos inconsistentes, es decir, eliminar datos que no están compuestos por otros elementos que no son coherentes.

2. Integración de datos. Donde grandes cantidades de datos pueden ser combinados después de eliminar los datos innecesarios.

3. Selección de datos. Donde los datos relevantes, proceden para hacer un análisis y se recuperan de la base de datos.

4. Transformación de datos. Donde los datos se transforman para darle una firmeza, solidez y estabilidad de una forma apropiada para la minería mediante operaciones de adición o agregación.

5. Minería de datos. Donde se le da un proceso esencial, los métodos inteligentes se aplican con el fin de extraer patrones de datos.

6. Evaluación de patrones. Donde se identifican los patrones realmente interesantes que representan el conocimiento.

7. Presentación del conocimiento. Donde las técnicas de visualización y representación del conocimiento se utilizan para presentar el conocimiento extraído para el usuario.

2.4.1. Tareas principales de la minería de datos

El objetivo principal de la MD es extraer nuevos conocimientos a partir de datos. Existen diversos tipos de métodos para extraer el conocimiento, estos métodos se agrupan de acuerdo al tipo de tarea que realizan. Las principales tareas son: clasificación, regresión, agrupación y asociación. A continuación se explican de manera resumida cada una de ellas.

2.4.1.1. Clasificación

Este tipo de tarea sirve para predecir la categoría a la que pertenece un objeto dado. Un conjunto de datos están etiquetados si todos o la mayoría de sus registros contienen un atributo especial llamado "atributo de clase". El objetivo de un atributo de clase es clasificar la muestra para que pertenezca a una categoría. Esta asociación se realiza con base en las características o propiedades de los objetos mediante un patrón frecuente.

La clasificación funciona mediante la descripción del manejo de datos de alguna área determinada. Llamados conjuntos de datos, y los elementos que los componen son conocidos como objetos, registros o muestras. A esta parte se le conoce como fase de entrenamiento, donde el clasificador mediante un análisis va aprender de un entrenamiento de un conjunto de tuplas de base de datos y los atributos asociados de la clase.

Una tupla para el aprendizaje de las máquinas se le conoce comúnmente como "muestras de entrenamiento". Una muestra, X , está representada por un vector de atributo de n-dimensiones, $X = (x_1, x_2, x_3, ..., x_n)$, que representa n mediciones hechas sobre los datos de la muestra de n atributos, respectivamente, $A_1, A_2, A_3, ..., A_n$. Cada atributo representa una característica de X, que pertenece a determinada clase según por el atributo de clase. Hay que mencionar que contiene valores discretos y desordenados, cada valor sirve para una determinada categoría [29]. Para poder entender esta forma matemática se explicará usando la Tabla 2.1, que muestra un conjunto de datos de personas con cancer. Cada fila de la Tabla representa una instancia o muestra, cada columna es una propiedad o atributo. La última columna es el atributo de clase.

Tabla 2.1: Ejemplo de un conjunto de datos con un atributo de clase de personas con cáncer

Nombre	Edad	Sexo	Tamaño de tumor	Benigno/Maligno
Karmen	45	Femenino	2 cm	Maligno
Roxana	30	Femenino	2 cm	Benigno
...	...	...	...	...
Roberto	12	Masculino	1.5 cm	Benigno
Juan	22	Masculino	7.5 cm	Maligno
n	n	n	n	n

En general, la información que se tiene dentro de una base de datos no es suficiente para tener una clara relación entre las entradas y salidas de valores. Por lo que se necesita buscar un modelo que describa el atributo clase como una función de los atributos de salida, es decir, la tarea de un clasificador es construir una función de decisión que pueda usarse para hacer buenas predicciones de etiquetas de los datos que entran.

2.4.1.2. Regresión

La regresión es similar al método de clasificación, pero con el objetivo de buscar patrones para determinar un valor numérico. Es decir, es una metodología estadística que se utiliza para la predicción numérica. Una regresión lineal consiste en encontrar la mejor línea de ajustar dos atributos, por lo que un atributo se puede utilizar para predecir la otra línea y una regresión lineal múltiple es una extensión de la regresión lineal, en donde más de dos atributos están involucrados y los datos se ajustan a una superficie multidimensional.

2.4.1.3. Agrupación

También conocida como segmentación (en inglés se le conoce como *clustering*) en la que se identifican grupos naturales de objetos, basándose en la similitud de atributos. Se puede utilizar para discretizar un atributo numérico. Las agrupaciones están formadas por varios grupos vecinos con el fin de formar conceptos de alto nivel. Una agrupación es una colección de objetos que son similares entre si, dentro del mismo grupo, y no son similares a los objetos de otros grupos. La ventaja de un proceso de agrupamiento es que es adaptable a los cambios y ayuda a sincronizar características útiles que distinguen a los otros grupos.

2.4.1.4. Asociación

También conocida como análisis de cesta de la compra. Donde un cliente representa una variable, en esta regla de asociación se tiene un sólo atributo por ejemplo la compra o producto, también contienen un sólo predicado este tipo de reglas se conocen como reglas de asociación de una sola dimensión.

2.5. Algoritmos de clasificación

En esta parte se explican de manera general algunos métodos de clasificación.

2.5.0.5. Árbol de decisión

En general, un árbol de decisión ayuda a tomar decisiones en cualquier área específica. Plantea de forma esquemática algunas alternativas disponibles y las posibles consecuencias que se puedan encontrar. Su nombre proviene de la forma que tiene un árbol natural, pero invertido. Es decir, la raíz de los árboles en computación se encuentran arriba, mientras que las hojas se encuentran abajo.

El modelo de lo árboles de decisión está constituido por múltiples nodos, donde cada nodo denota una prueba en un valor de atributo. Cada rama del árbol representa un resultado de la prueba anterior y cada hoja representa una clase o la distribución de la clase [29]. En síntesis, un árbol tiene un nodo raíz en la parte más alta del árbol, en el que se desglosan nodos internos que representan una prueba en un atributo, el árbol contiene nodos terminales u hojas que mantienen una etiqueta de clase. En cada nodo que no es hoja, hay bordes entrantes y salientes que son las ramas, cada rama en un nodo interno representa un resultado. La Figura 2.7 muestra un ejemplo de un árbol de decisión.

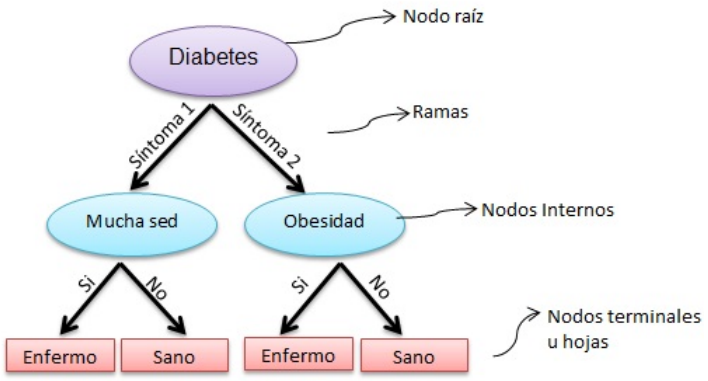

Figura 2.7: Ejemplo de un árbol de decisión

Al construir una partición en cada nodo, se pretende aumentar la pureza de los nodos. Las reglas de partición de un nodo dependen exclusivamente de los atributos, así que se debe elegir a la que mejor contribuya al aumento de la homogeneidad de sus hijos. Esto se logra definiendo una medida de impureza sobre la variable de respuesta, con valores discretosX^jde números continuos, se plantea la siguiente regla $X^j \leq c$, *con* $c \epsilon \mathbb{R}$

Las medidas de impureza más comunes de un nodo, según los criterios de clasificación de árboles de decisión son [30]:

- Medida de entropía: Se concibe como medida de desorden a la peculiaridad de ciertas combinaciones. La entropía puede ser considerada como una medida de la incertidumbre y de la información necesaria para que en cualquier proceso pueda acotar, reducir o eliminar la incertidumbre como se ve en la ecuación 2.1.

$$i_{ent}(t) = -\sum_{j=1}^{J} p_j(t)\, log\, p_j(t) \tag{2.1}$$

donde:

i = impureza.

t = cada nodo del árbol

$p_j(t)$ = probabilidad condicional de que un elemento pertenesca a la clase j

- Índice de Gini: Es una medida de desigualdad bajo la ecuación 2.2.

$$i_{Gini}(t) = \sum_{i,j=1;i\neq j}^{J} p_i(t)\, p_j(t) = 1 - \sum_{j=1}^{J} [p_j(t)]^2 \tag{2.2}$$

La elección de la medida de impureza depende del problema y del predictor construido. Por ejemplo un nodo t es partido en dos t_I (nodo izquierdo) y t_D(nodo derecho). Sea p_I la proporción de elementos del nodo t que caen en el hijo izquierdo y p_D la del derecho, como se ve en la Figura 2.8.

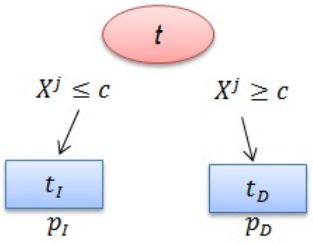

Figura 2.8: Partición de un nodo, si el atributo X^j supera o no al umbral c

Una vez construido completamente el árbol de decisión, las reglas de poda intentan eliminar los subárboles que no contribuyen significativamente a la precisión de la clasificación. De hecho, el método recursivo de construcción de árboles de decisión continúa dividiendo el conjunto de casos de entrenamiento, hasta que encuentra un nodo puro. El resultado suele ser un árbol muy complejo, más de lo deseable, que "sobre ajusta" los datos del conjunto de entrenamiento [31]. El sobreaprendizaje es un problema bastante importante ya que limita considerablemente la aplicabilidad del modelo de clasificación aprendido.

2.5.0.6. Redes Neuronales Artificiales

Las Redes Neuronales Artificiales (RNA) fueron inspiradas en las neuronas biológicas. Actualmente, se considera que las neuronas biológicas son unidades de procesamiento conectadas unas con otras, y que responden a un estimulo eléctrico de otras neuronas a través de su árbol dendrítico. Este estimulo eléctrico, al pasar de un cierto umbral, hace que una neurona active su salida, y se comunique con otra por medio de su axón. A una unión funcional entre dos neuronas se le llama sinapsis. Este modelo inspiró a McCulloch y Pitts en 1943, para crear un modelo artificial, con el que las computadoras pudieran aprender. Hebb, en 1949, propuso un modelo para aprendizaje de las neuronas. Dos de los algoritmos de aprendizaje más importantes actualmente son retro propagación (*backpropagation*) y propagación rápida (*quick propagation*). En 1957, Rosenblatt creó un modelo de un perceptrón simple [32].

El perceptrón, en su forma más básica, consiste en representar a una neurona capaz de aprender una función discriminante lineal $fd(X)$. Esta función debe ser capaz de separar perfectamente a cualquier conjunto de entrenamiento linealmente separable. La función corresponde a una combinación lineal, que para el caso de una neurona es la

suma ponderada de sus entradas. Ver ecuación 2.3.

$$fd(X) = \sum_{i=1}^{n} w_i x_i + w_{n+1} \tag{2.3}$$

La ecuación 2.3 representa hiperplano en el espacio de n dimensiones. En la Figura 2.9 se muestra la representación de un perceptrón simple.

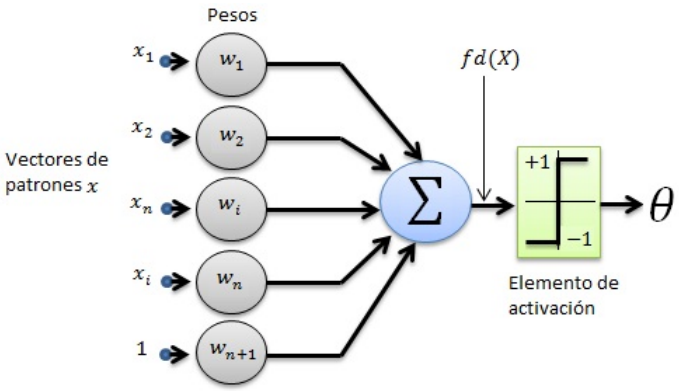

Figura 2.9: Ejemplo de un perceptrón simple

donde:

x_i son los valores de ciertos patrones que entran por las dendritas. Estos valores pertenecen a la variable independiente x.

w_i son valores constantes, llamados pesos sinápticos.

w_{n+1} es el incremento unitario de los pesos sinápticos

$fd(X)$ es la función discriminante, y arroja un resultado positivo o negativo.

θ es el elemento que determina si se activa o no la neurona bajo un umbral, bajo la siguiente regla: Si $fd(X) > 0\, entonces\ = 1$ y si $fd(X) < 0\, entonces\ - 1$. La salida se ve como un escalón.

El perceptrón puede clasificar con precisión las funciones booleanas estándar, como AND, OR, NAND y NOR. Estas funciones pueden ser linealmente separadas por el hiperplano. En el caso de la función XOR, que es linealmente inseparable, una neurona es incapaz de realizar una clasificación perfecta. En la Figura 2.10, se muestra un ejemplo de clasificación con las funciones lógicas AND y XOR.

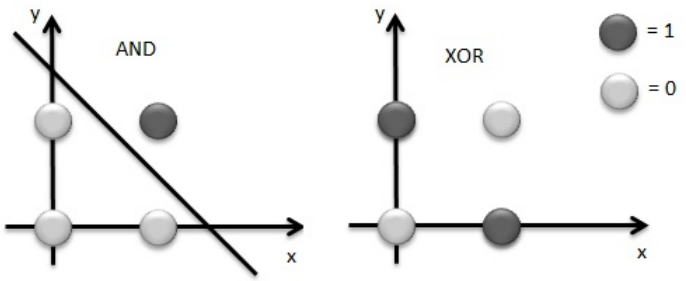

Figura 2.10: Ejemplo de clasificación de un perceptrón

Como se puede apreciar, en la función AND, las respuestas si se pueden dividir o clasificar correctamente usando un hiperplano. A esto se le llama conjunto de datos linealmente separable. En el caso de la función XOR, es imposible encontrar un hiperplano que separe correctamente todas las respuestas. Se necesitan al menos dos hiperplanos para resolver este problema. Se descubrió que si se incluyen nuevas capas de perceptrones, se puede resolver el problema de la función XOR. A esta arquitectura se le conoce como Red Neuronal Artificial.

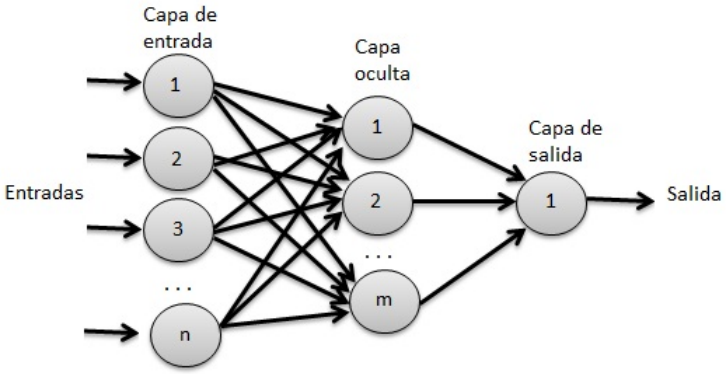

Figura 2.11: Ejemplo de un perceptrón multicapa

2.5.0.7. Máquina de soporte vectorial

Las máquinas de soporte vectorial (SVMs, por sus siglas en inglés: *Support Vector Machines*), son un tipo de clasificador de patrones basado en técnicas estadísticas de aprendizaje. Estas fueron propuestas por Vladimir Vapnik y sus colaboradores en 1992. Las SVM permiten resolver problemas de clasificación y de regresión.

Para la resolución de este tipo de problema, se toma un conjunto de vectores de entradas $X = \{x_i \mid x_i = (x_{i1}, x_{i2}, ..., x_{in})\}$ que representan patrones de n-dimensiones. Se considera un conjunto de vectores $Y=\{y_i \in \{+1, -1\}\}$

que representa las etiquetas o clases de objetos. El conjunto de datos de entrenamiento para las SVMs aplicadas como clasificadores es entonces definido como $\{(x_i, y_i)\ donde\, x_i \in R^d,\ y_i \in \{+1, -1\}, i = 1, ..., N\}$.

Cuando los conjuntos de datos no son linealmente separables, las SVMs usan funciones llamadas *kernel* o núcleo. Estas son usadas para medir la influencia de un patrón sobre otros patrones cercanos. El efecto del uso de kernels es transformar los vectores de entrada en vectores de mayor dimensión, para conseguir que las clases sean linealmente separables.

Las SVM clasifican patrones utilizando la evaluación de una función similar a la ecuación 2.4 [32]:

$$f(x) = \sum_{vectores\ soporte} \alpha_i y_i (\Phi(x) \cdot \Phi(x_i)) + b = \sum_{vectores\ soporte} \alpha_i y_i K(x, x_i) + b \qquad (2.4)$$

- Clasificación lineal: Las SVM generan un hiperplano que separa los patrones con clase positiva (y_i =+1) de los patrones con clase negativa (y_i =-1). Los puntos x_i que están en el hiperplano satisfacen $w^T x + b = 0$. La idea es encontrar el hiperplano óptimo, en términos de margen máximo. El margen se define como la distancia entre el objeto de cada clase más cercano al hyperplano. Los vectores de soporte son los puntos que tocan el límite del margen, como se ve en la Figura 2.12.

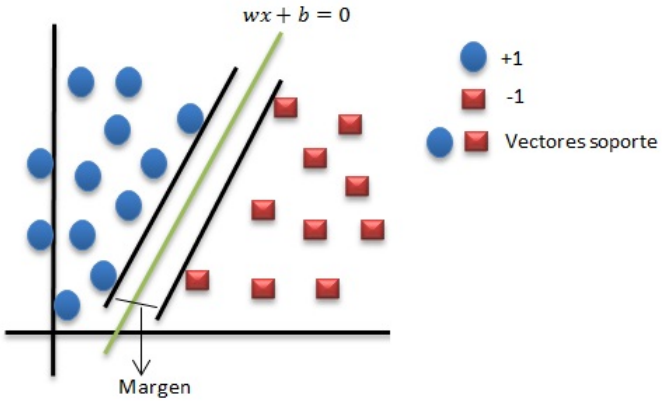

Figura 2.12: Ejemplo de clasificación lineal en SVM

- Clasificación no lineal: Se realiza una transformación del espacio de entrada a otra

dimensión más alta, en el que los datos sean linealmente separables. Para ello se usa una función *kernel*.

Los tipos de *kernel* más comunes se muestran la Tabla 2.2.

Tabla 2.2: Tipos de funciones *kernel*

Tipo de kernel	función
Gausiano RBF	$K(x_i, x_j) = \exp^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$
Polinomial	$K(x_i, x_j) = (x_i \cdot x_j + 1)^d$
Lineal	$K(x_i, x_j) = x_i \cdot x_j$

Las SVM se han utilizado en diversos problemas de reconocimientos de patrones como detección de fraudes, análisis de texto, tratamiento de imágenes, en áreas de medicina, bioinformática, etc.

2.6. Weka

La herramienta Weka fue desarrollada en la Universidad de Waikato en Nueva Zelanda. El nombre fue elegido para representar a una ave con una naturaleza inquisitiva (examinar, averiguar o indagar cuidadosamente algo) llamada Weka, que sólo se encuentra en las islas de Nueva Zelanda. Weka fue desarrollada en lenguaje Java y se distribuye bajo los términos de la Licencia Pública General (GNU), lo que significa que el programa es de libre distribución. Este software funciona casi en cualquier plataforma. Ha sido probado en Linux, Windows y Macintosh [33].

Weka proporciona implementaciones de algoritmos de aprendizaje para tareas de minería de datos. Incluye una variedad de herramientas para la transformación de conjuntos de datos en varios formatos. Weka puede realizar las tareas de regresión, clasificación, agrupamiento (*clustering*), reglas de asociación y la selección de atributos.

Weka se puede descargar de http://www.cs.waikato.ac.nz/ml/weka. Los sistemas operativos soportados son Windows x86, Windows x64, Mac OS X y Linux en versiones de 32 y 64 bits. Weka está desarrollado en lenguaje Java, por lo consiguiente se requiere tener instalada una máquina virtual de Java para poder ejecutarse. En la página mencionada anteriormente, hay versiones para descargar que ya incluyen la máquina virtual.

La versión usada en este trabajo es la versión de desarrollador. Una vez instalada la herramienta, la interfaz de usuario es como se muestra en la Figura 2.13. Esta ventana proporciona acceso a todas las opciones de Weka.

Figura 2.13: Interfaz de Weka 3.7.9

Weka trabaja con un formato denominado arff (*Attribute-Relation File Format*) este formato está compuesto por una estructura diferenciada por tres partes [34, 35]:

1. **Cabecera**: Se define el nombre de la relación. @relation <nombreRelacion> , si contiene un espacio en el nombre de la relación hay que marcarlo entre comillas simples.

2. **Declaración de atributos**. Se declaran los atributos que contiene el conjunto de datos. La sintaxis es la siguiente: @attribute <nombreAtributo> <tipo> donde <nombreAtributo> es de tipo *string* y en <tipo> acepta números reales de tipo *real*, números enteros de tipo *Integer*, fechas de forma entre comillas "dd-MM-yyyy" de tipo *date*, cadenas de texto de tipo *string*, valores entre llaves y comas {uno, dos, tres} de tipo enumerado *Numeric*.

3. **Declaración de datos.** Se declaran los datos que componen la relación, separando entre comas atributos y saltos de líneas las relaciones. En el caso de que algún dato sea desconocido se expresará con un signo de interrogación que cierra "?", para introducir comentarios se usa el símbolo " %" como se describe en la Figura 2.14.

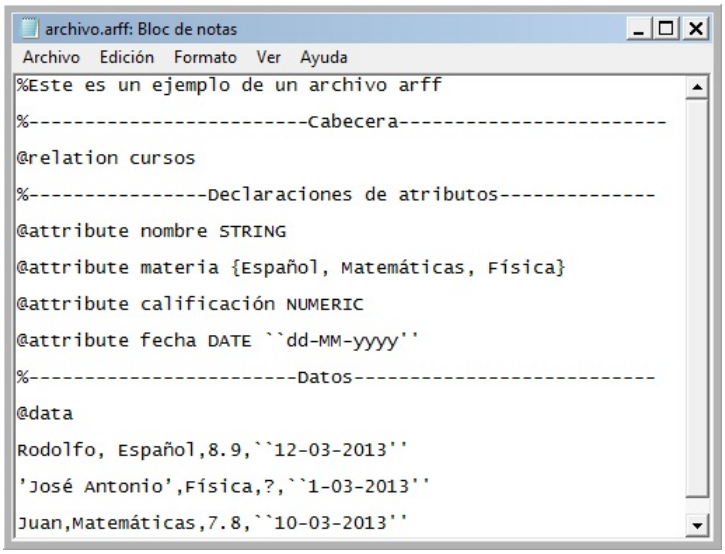

Figura 2.14: Ejemplo de la estructura de un archivo arff

La interfaz *Explorer,* permite visualizar y aplicar distintos algoritmos de aprendizaje a un conjunto de datos como clasificación, agrupamiento, búsqueda de asociaciones, selección de atributos y visualización de datos. También permite abrir archivos con el formato .arff o directamente de la red introduciendo la URL (*Uniform Resourse Locator*) como se ve en la Figura 2.15.

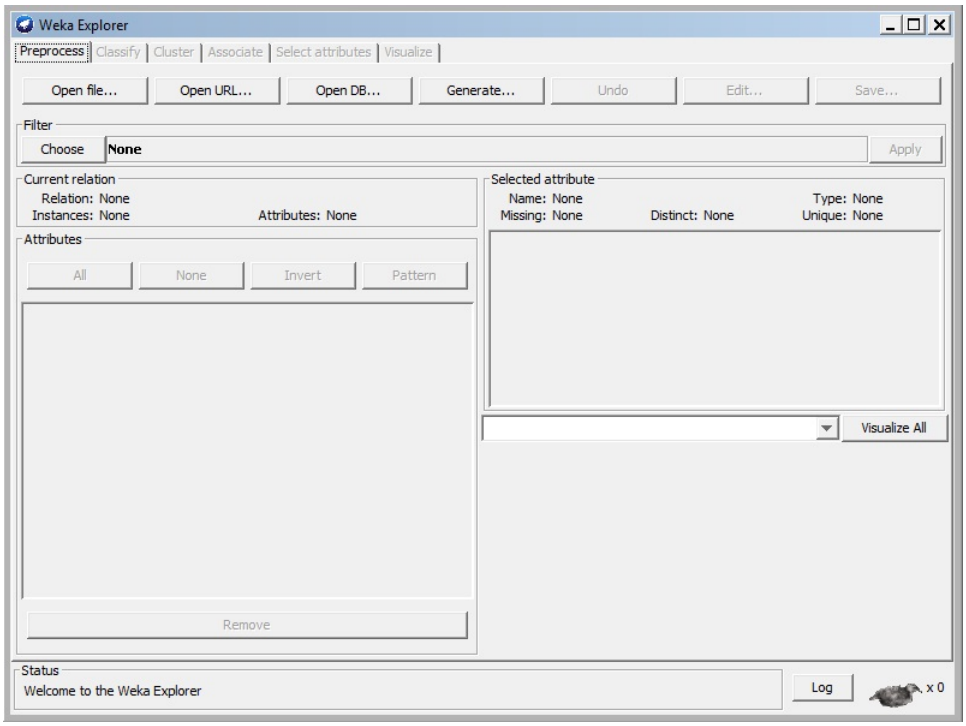

Figura 2.15: Ventana inicial de la interfaz *Explorer*

Al entrar directamente en modo *Explorer,* de manera predeterminada se elige la pestaña de *Preprocess,* que sirve para procesar los datos. Este preprocesamiento incluye la eliminación de patrones con atributos incompletos. El primer paso para empezar un análisis de datos es definir el origen de los mismos. En nuestro caso se trata de un archivo arff almacenado localmente.

Weka ya tiene algunos archivos arff que se pueden utilizar para probar y observar como funciona el software. Para acceder a ellos hay que posicionarse en la carpeta donde está instalado Weka y abrir un archivo, en este caso se abrirá el archivo iris.arff. Al abrirlo, se muestran 4 atributos de tipo numérico y un atributo de clase, que son:

- Longitud del sépalo (el sépalo son las hojas verdes que se encuentran debajo de los pétalos) en centímetros (cm).
- Anchura del sépalo en cm.
- Longitud del pétalo en cm.
- Anchura del pétalo.
- Clase {Iris-setosa, Iris-versicolor, Iris- Virginica} que son tres tipos de flores distintas con diferentes características cada una. Hay muchos tipos de iris, pero en este conjunto de datos se clasifican estas tres especies con cincuenta casos, como se puede ver en la Figura 2.16

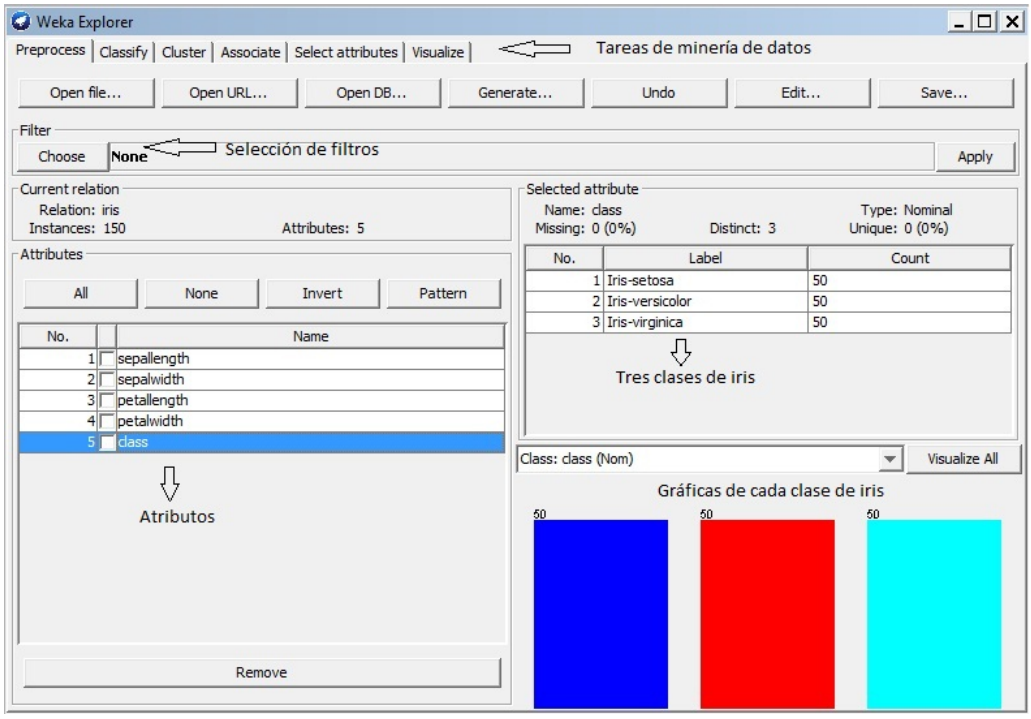

Figura 2.16: Conjunto de datos del archivo Iris

Se pueden aplicar diferentes tipos de filtros sobre los datos, permitiendo alterar los datos como agregar o eliminar atributos, introducir ruido a los datos para tener otros resultados. La segunda pestaña en el *Explorer* es la que corresponde a la tarea de clasificación (*Classify*) . Este tipo de tarea clasifica los datos por varios tipos de métodos, la interfaz de clasificación se puede ver en la Figura 2.17.

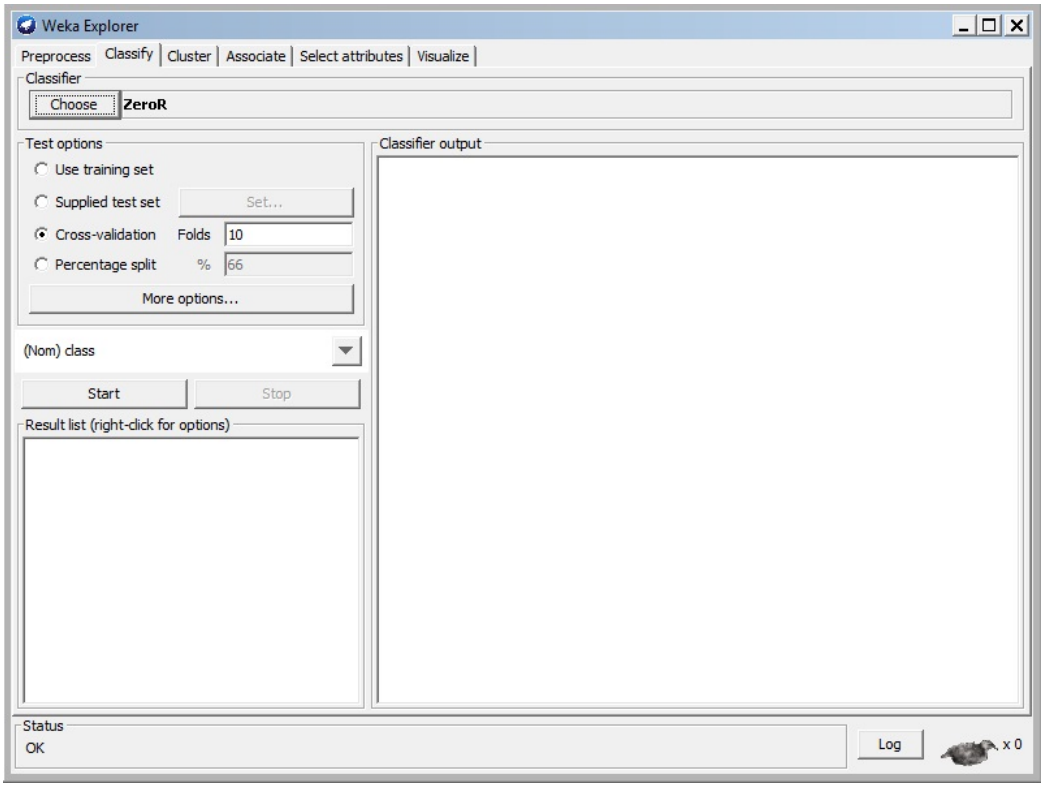

Figura 2.17: Interfaz *Classify*

Los métodos de clasificación se pueden elegir en la opción *choose.* Al dar clic se despliega un árbol que permite elegir el método que se quiera aplicar. Para empezar el entrenamiento se da un clic en el boton *Start*. Al término del entrenamiento y prueba se pueden visualizar los errores, el porcentaje de aciertos en *test*, y las estadísticas evaluadas con el mismo método. También se puede elegir el tipo de *test* para ejecutar un clasificador con una mejor precisión, la que viene por default es validación cruzada (*Cross-validation*) este tipo de *test* es dividida en partes para revolver conjuntos de datos y evaluarlos para hacer el clasificador con cada una de las partes revueltas o cruzadas.

Para presentar un ejemplo de clasificacion, se eligió un árbol de decisión de tipo *NBTree* (*Naive Bayes*), en el que los clasificadores NB están en las hojas. En la Figura 2.18, se muestran los resultados que genera Weka.

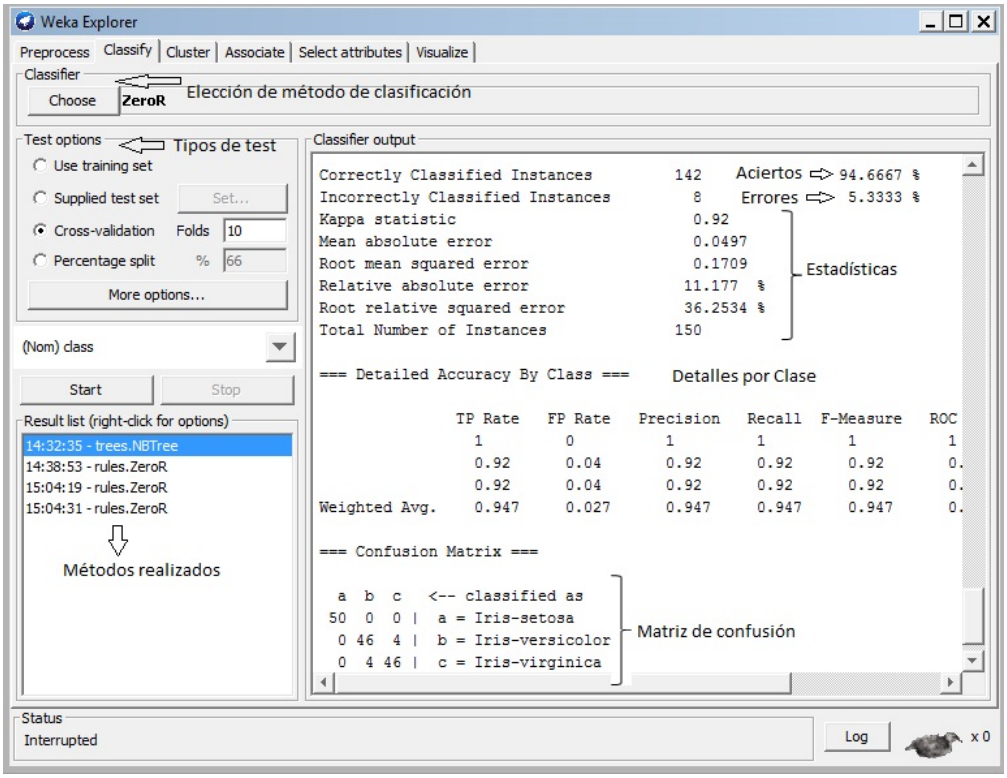

Figura 2.18: Algunos métodos de clasificación

La Figura 2.19 muestra el árbol generado. Esta es otra de las funcionalidades de Weka.

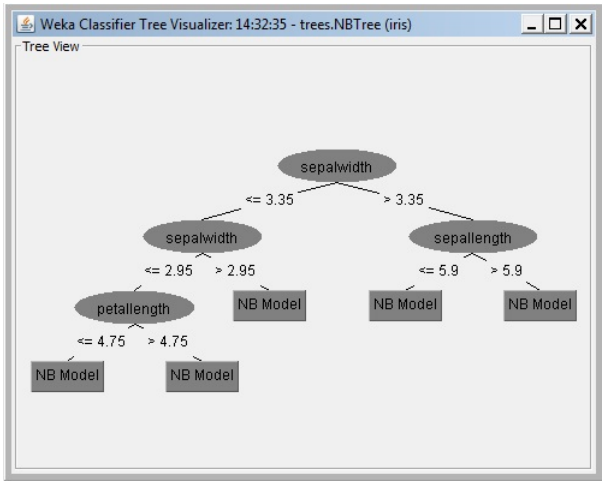

Figura 2.19: Árbol de decisión con *Naive Bayes*

Otra tarea de MD aplicada en este software es la agrupación (*Cluster*). La forma de acceder a los algoritmos de este tipo es muy similar a la de clasificación, para ello se elije la pestaña de *Cluster* en el *Explorer*. La Figura 2.20 muestra un ejemplo de

los resultados obtenidos con Weka para la tarea de agrupación. De manera similar al ejemplo anterior, para visualizar cada uno de los *clusters,* se posiciona el usuario en el tipo de método que se eligió en la parte inferior izquierda y con un clic derecho muestra una lista de opciones que se pueden ver gráficamente.

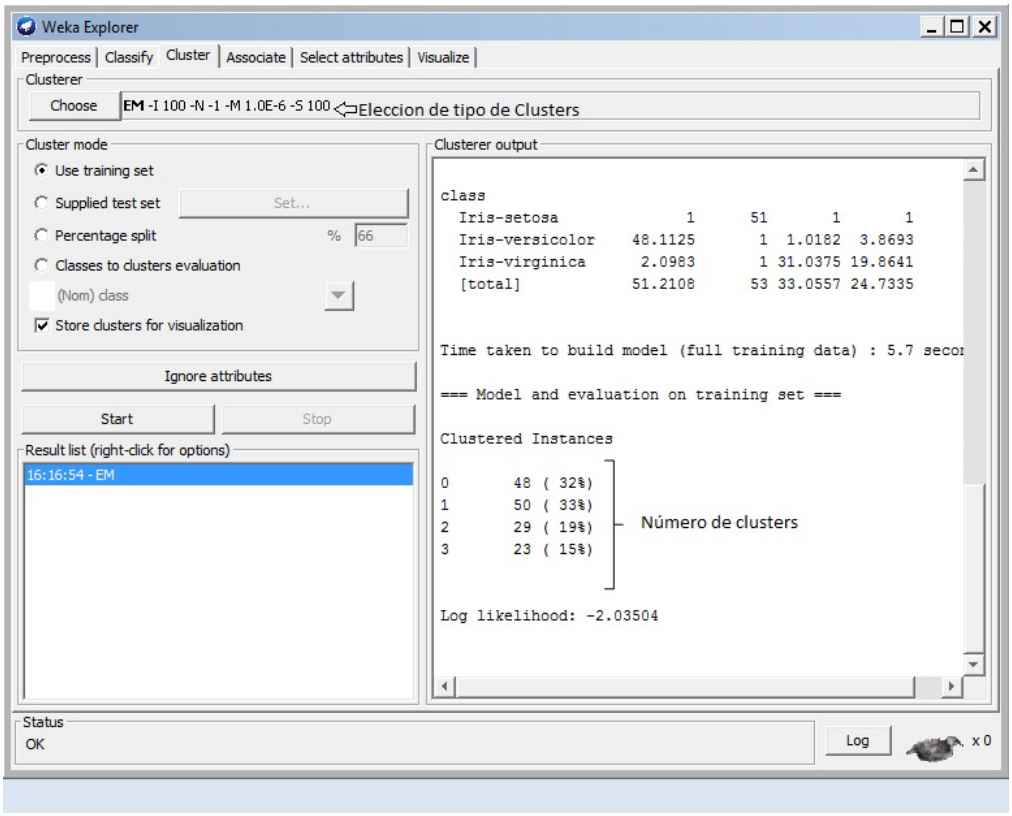

Figura 2.20: Ejecución de un método de Cluster

En la cuarta pestaña se encuentra la tarea de asociación (*Associate*), que es un método que permite buscar asociaciones entre los datos. La pestaña de *Select attributes* pernite sleccionar los atributos que contienen más influencia en asociaciones. La última pestaña que es de visualización (*Visualize*), que muestra la representación gráfica de todos los atributos. Permite mostrar los posibles pares de combinaciones de atributos, para poder apreciar la relación que existe entre ellos en una gráfica de dos dimensiones. La Figura 2.21 muestra un ejemplo de visualización.

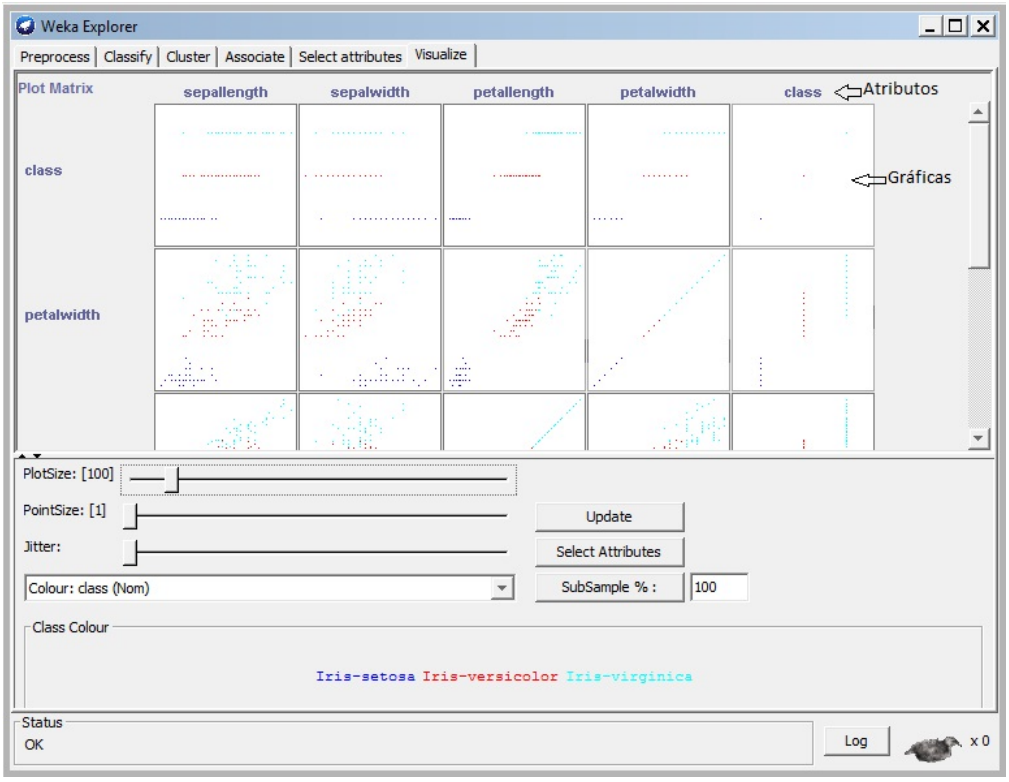

Figura 2.21: Visualización de gráficos de atributos

La interfaz *Experimenter* de Weka está diseñada para responder a la pregunta básica en la aplicación de técnicas de clasificación y regresión: ¿Qué métodos son más eficaces para un problema dado?. Esta interfaz permite recopilar estadísticas de rendimiento y realizar pruebas a los algoritmos. En *KnowledgeFlow* se permite diseñar configuraciones de *streaming* para el procesamiento de datos, tambien permite arrastrar cajas que representan el aprendizaje. *SimpleCli*, es una abreviación de *Simple Client*, es útil como una herramienta de ayuda a la fase de pruebas y proporciona una interfaz para poder introducir comandos.

Weka posee varias técnicas que pueden ser utilizadas en áreas como: marketing, manufactura, salud, energía, finanzas, medicina, etc. al definir la tarea que se quiera realizar. Además de Weka, existen en el mercado herramientas para la explotación de minería de datos. Básicamente, cumplen con la misma finalidad a continuación se comparan algunas herramientas con Weka como se ve en la Tabla 2.3 según [36].

Tabla 2.3: Cuadro comparativo de herramientas para minería de datos

Característica	Clementine	SAS Enterprise Miner	Tariykdd	Weka
Licencia libre	No	No	Si	Si
Conocimientos avanzados	No	No	No	No
Acceso a SQL	Si	No	Si	Si
Multiplataforma	No	Si	Si	Si
Bases de datos especializadas	No	—	No	No
Métodos de máquina de vector soporte	Si	Si	No	Si
Métodos bayesianos	Si	—	No	Si
Combinación de modelos	Si	Si	No	Si (no muy eficiente)
Modelos de clasificación	Si	Si	Si	Si
Árboles de decisión	Si	Si	Si	Si
Modelos de regresión	Si	Si	No	Si
Agrupamiento (Clustering)	Si	Si	No	Si
Interfaz amigable	Si	Si	Si	Si
Visualización de datos	Si	Si	Si	Si

Como se puede observar, las herramientas Clementine y SAS Enterprise Miner no son de licencia libre, se debe que pagar para poder trabajar con ellos. Además, SAS Enterprise Miner no cuenta con acceso a SQL. Clementine no es multiplataforma. Tariykdd y Weka son de licencia libre. La principal diferencia que existe entre estas dos últimas herramientas es que Tariykdd no cuenta con métodos de máquina de soporte vectorial. Estas son las razones por las que Weka fue elegida para este trabajo.

Capítulo 3

Diabetes y hepatitis

La minería de datos ha sido utilizada en muchas áreas tales como la medicina, comercio electrónico y detección de fraudes. En esta investigación, se aplican técnicas de clasificación a datos sobre las enfermedades de Diabetes y Hepatitis. En este capítulo se describen de manera general ambas enfermedades, se muestran los conjuntos de datos a utilizar, y presenta el significado de cada uno de sus atributos.

3.1. Diabetes mellitus

Las primeras descripciones de la Diabetes tratan desde 1553 A.C, con evidencias descritas por los egipcios en el papiro encontrado en Tebas por el arqueólogo alemán George Ebers, en 1862. Tebas es conocida actualmente como la ciudad de Luxor. Otras culturas también conocían la Diabetes, como las culturas orientales. El dato más antiguo que se tiene pertenece a un médico hindú llamado Súsruta en el siglo V A.C. Este médico observó los síntomas de esta enfermedad, a la que llamó enfermedad de ricos, porque afectaba a personas obesas y consumidores de grandes cantidades de dulces y de arroz [37].

El nombre médico de esta enfermedad conocida mundialmente en lengua castellana, catalana y románica o neolatina es "Diabetes mellitus". La palabra Diabetes tiene su origen de un vocablo griego (*diabeinenen*) que significa "Pasar a través con fuerza". Pues uno de los principales síntomas más llamativos por la que es conocida esta enfermedad es por eliminar de manera exagerada el agua por el riñón. Refiriéndose a este síntoma, Areteo de Capadocia [38], en el siglo II D.C, lo denominó como "sifón", queriendo expresar que el agua entraba al organismo del diabético y de manera rápida salía del mismo, sin tener un control propio de almacenamiento.

A mediados del siglo XVII, Thomas Willis hizo una descripción científica al hacer un experimento, al probar la orina de un paciente con Diabetes y su sorpresa fue que la orina tenía un sabor muy dulce como la miel, ya que la orina de un diabético contiene demasiada azúcar y así surge el vocablo latino mellitus (sabor a miel). En el siglo XIX, Claude Bernard hace los primeros trabajos experimentales observando que el azúcar que aparece en la orina de los diabéticos había estado almacenado en el Hígado en forma de glucógeno intuyendo que la Diabetes era ocasionada por la incapacidad del Hígado.

En la segunda década del siglo XX, se llegó de manera asertiva a la conclusión de que el páncreas es el causante de la Diabetes, esto se logró mediante varios experimentos que se realizaron con extirpaciones de pedazos de este órgano, para hacer un análisis de lo que sucede durante la digestión. En los experimentos, los jugos fermentados que pasaban al intestino para la digestión de los alimentos, demostraban que también se digería insulina, esta substancia es segregada por el páncreas. Los jóvenes canadienses Frederik Grant Banting y Charles Best [39] consiguieron aislar la insulina del páncreas y demostrar su efecto hipoglucemiante en 1921, esto permitió que se desarrollaran medicamentos para la disminución del azúcar en la sangre. Actualmente esta es la forma más generalizada para tratamiento de pacientes diabéticos.

3.1.1. El páncreas, órgano afectado por la diabetes

El páncreas es una glándula que mide alrededor de 15 a 23cm de largo, 4cm de ancho, 5cm de grosor y se ubica en el abdomen. Está rodeada por el estómago, el intestino delgado, el hígado y la vesícula biliar, el páncreas tiene la forma de una pera plana. El extremo ancho del páncreas se llama cabeza, las secciones medias son el cuello y el cuerpo y el extremo delgado es la cola como se ve en la Figura 3.1 [40].

El páncreas tiene dos funciones principales: la función exocrina y la endocrina. La descripción de la función exocrina se puede describir como sigue:

En el páncreas existen células llamadas exocrinas, que producen enzimas para la digestión de los alimentos. Cuando los alimentos llegan al estómago, el conducto pancreático libera enzimas en la primera parte del intestino delgado para la digestión de grasas, carbohidratos y las proteínas de los alimentos. La segunda función del páncreas es la endocrina, usada para la producción de hormonas que circulan en el torrente

sanguíneo. Las hormonas pancreáticas principales son la insulina y el glucagón. La insulina sirve para bajar el nivel de azúcar en la sangre, mientras el glucagón lo aumenta, juntas estas dos hormonas trabajan para mantener el nivel adecuado de glucosa en la sangre [40].

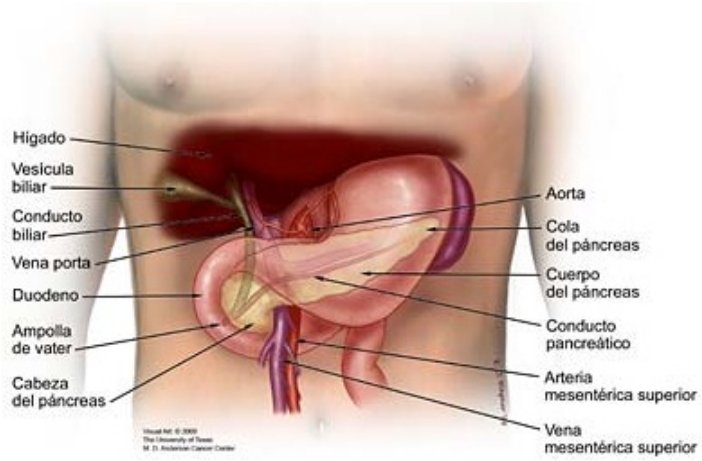

Figura 3.1: Anatomía del páncreas en el cuerpo humano
Imagen obtenida de
http://www.pancan.org/section_en_espanol/learn_about_pan_cancer/what_is_the_pancreas.php

3.2. Diabetes

La Diabetes se mantiene como la primera causa de muerte en México y representa el 13.67 % de la mortalidad total [1] y afecta a más del 4 % de la población mundial, por lo que se considera de alta frecuencia. Para poder metabolizar el azúcar se necesita de la hormona insulina que permite controlar el azúcar que proviene de los alimentos en la corriente sanguínea cuando se sufre un déficit de esta hormona pueden aumentar peligrosamente el nivel de glucosa en la sangre [41]. Es decir, cuando el páncreas deja de producir el nivel necesario de insulina aumenta el nivel de riesgo para contraer esta enfermedad.

Los síntomas más característicos de la Diabetes es tener una sed insaciable y una excesiva necesidad de orinar, que puede provocar que el cuerpo pierda fluidos esenciales. Existen dos tipos de Diabetes la de tipo I y la de tipo II. A continuación se describen ambos tipos.

3.2.1. Tipos de diabetes

3.2.1.1. Diabetes tipo I

En la gran mayoría de los casos se presenta en personas jóvenes y en niños. Los síntomas más comunes de la Diabetes tipo I son: cansancio o debilidad, apetito excesivo, mucha sed, necesidad frecuente de orinar, presencia de azúcar en la orina y sangre, visión borrosa, dolor en las piernas, cambios en la piel y deficiencia en cicatrización. En este tipo de Diabetes, la persona enferma no produce insulina, por cuestiones de que el páncreas no la pueda producir de manera eficiente, o si la produce es muy poca. Es necesario recurrir a inyectarse insulina cada día, ya que si se administra de forma vía oral, los ácidos del estómago la vuelven ineficiente [1] .

3.2.1.2. Diabetes tipo II

Es la más común y puede presentarse en personas jóvenes, pero en la mayoría de los casos se da en personas mayores de 40 años. Los pacientes que desarrollan este tipo de Diabetes no necesitan inyectarse insulina ya que su cuerpo la produce, pero no la utiliza de forma correcta, porque las células se resisten a ella. Es producto de la obesidad, ya que la grasa bloquea los receptores de insulina en la célula y ocaciona que el azúcar aumente en la sangre. La mejor manera de tratar esta enfermedad es haciendo un seguimiento en dietas, ejercicio y control de peso [1] .

3.3. Conjunto de datos Diabetes

El conjunto de datos Diabetes.arff fue proporcionado por Vincent Sigillito en mayo de 1990. Sin embargo, el lugar en donde se obtuvieron los datos originalmente fue en el Instituto Nacional de Diabetes y Enfermedades Digestivas y Renales [42]. Este conjunto de datos fue practicado a 768 personas para su evaluación y predicción de la enfermedad. El conjunto de datos describe nueve atributos incluyendo el atributo de clase. Los atributos que se mencionan son los números de embarazos que han tenido las mujeres, algunos diagnósticos para evaluar los niveles de azúcar, la presión arterial, 2 horas de insulina en suero, edad y el atributo de clase donde el cero indica examen negativo y el uno indica examen positivo, etc. Como se ve en la Tabla 3.1, estos atributos están ligados con la enfermedad para su pronóstico.

Tabla 3.1: Atributos del conjunto de datos Diabetes

No.	Atributo	Tipo de dato
1.-	Número de embarazos:	Real
2.-	La concentración plasmática de glucosa de 2 horas en una prueba de tolerancia de glucosa oral:	Real
3.-	La presión arterial diastólica (mm Hg):	Real
4.-	Grosor del pliegue cutáneo del tríceps (mm):	Real
5.-	2 Horas de insulina en suero (mc U / ml):	Real
6.-	Índice de masa corporal $IMC = {}^{peso}/_{altura^2}(\mathrm{kg}/m^2)$	Real
7.-	Función de diabetes pedigrí:	Real
8.-	Edad:	Real
9.-	Clase:	Enumerado (0, 1)

La mayoría de los pacientes que se sometieron a esta prueba son mujeres de 21 años de edad de herencia indígena. Las personas que tienen altos niveles de azúcar en la sangre se deben hacer pruebas de tolerancia de glucosa que es una herramienta para diagnosticar si se padece Diabetes. La prueba más común de tolerancia a la glucosa consiste en administrar una dosis única oral de glucosa. Después de ingerir 75 gramos de glucosa se debe tomar una muestra de sangre a las 2 horas del suministro. Anteriormente a la prueba, se debe de someter al paciente a dietas adecuadas, para mantener un nivel de glucosa diaria de al menos 150g de carbohidratos diarios, el ayuno previo debe ser de 8 a 14 horas.

En condiciones normales, la sangre en ayunas debe tener un nivel de glucosa inferior a 100 miligramos por decilitro (mg/dl), en una hora después de la prueba menos de 200 mg/dl, en dos horas menos de 140 mg/dl. Los valores entre 140 y 199 se consideran de intolerancia a la glucosa y hay mayor riesgo de contraer Diabetes y por encima de los 200 ml/dl indican que el paciente tiene Diabetes [43].

Aproximadamente el 73 % de los adultos con diabetes tienen presión sanguínea mayor o igual a 130/80 milímetros de mercurio (mm Hg), uno de los métodos para medir la presión alta en la sangre que es la fuerza que la sangre ejerce en contra de las paredes de las arterias es la presión diastólica, que es la presión dentro de la arteria cuando el corazón está descansando y se está llenando de sangre [44] los valores de presión diastólica en adultos con presión alta es de 90 mm Hg o mayor.

Para calcular la cantidad de grasa subcutánea del individuo se hace por medio de un método llamado pliegue cutáneo del tríceps, consiste en la inserción de un pliegue de piel entre las pinzas de un compas para ser medida la cantidad de grasa del individuo de 0 a 65mm. Los niveles normales de concentraciones de insulina en suero son de 0 a 20 micro unidad por mililitro (mcU/ml) en ayunas y después de los alimentos 50-200 mcU/ml. Se cree que el sobrepeso puede elevar los niveles de colesterol y la presión arterial. La obesidad aumenta las probabilidades de tener otros factores a parte de desarrollar Diabetes como riesgo cardiovascular, presión arterial alta y colesterol elevado. Actualmente la medición de la obesidad se hace por medio de una formula llamada índice de masa corporal que se calcula dividiendo el peso en kilogramos (Kg) entre el cuadrado de la altura en metros se habla de sobrepeso cuando se tiene un resultado de más de 25 y de obesidad con valores superiores a 30 [45].

A veces en las familias hay descendencia en la enfermedad de la Diabetes, en algunos casos varían en la edad de cada familia. Para saberlo, la función de Diabetes Pedigrí se obtiene cuando la métrica tiene valores menores o iguales a 0.546 pero cuando se superan esos valores o menores a 94.5 se puede decir que se tiene una gran concentración de glucosa en la sangre y en efecto el individuo heredó la enfermedad de uno de sus familiares [46]. Por último, se tiene el atributo edad de tipo real y el atributo de clase de tipo enumerado donde el cero indica que no tiene la enfermedad y el uno que si tiene la enfermedad.

3.4. Hepatitis viral

La historia de la Hepatitis se remota desde la antigüedad, cuando Hipócrates (450 A.C - 380 A.C) descubre la Ictericia que es la coloración amarilla de la piel, pero no fue hasta el siglo VIII cuando empezó la Hepatitis a brotar como enfermedad infecciosa. En 1889, en Alemania A. Lurman reportó brotes de Hepatitis, transmitida, por medio de transfusiones de sangre.

En 1942, en la segunda guerra mundial, se detectaron brotes de una nueva Hepatitis a soldados americanos después de que se vacunaran contra el sarampión y la fiebre amarilla. En 1946, Mc Callum [47] clasifica a la Hepatitis viral en dos tipos 1. Hepatitis viral A o hepatitis infecciosa y 2. Hepatitis viral B o hepatitis sérica. En 1965, Barry Blumberg [48] gana el premio novel por descubrir HBsAg (por su siglas en inglés) Antígeno de Superficie de la Hepatitis B, inicialmente identificado de un aborigen

australiano. Dane descubrió la partícula completa del virus de la Hepatitis B (VHB) a la que denominó partícula de Dane en 1970. En 1973, Steve Feinstone y Robert Purcell describieron partículas virales en la deposición de voluntarios infectados con Hepatitis, encontrando el virus de la Hepatitis A (VHA).

En 1977, M. Rizetto describió un virus RNA (virus que usa ácido ribonucleico) pequeño, que requiere de un hepadnavirus (virus que daña al Hígado de humanos y animales) para replicarse y causar infección, al que denominó desde un principio agente delta y virus delta después (VHD). En 1983, Balayan identificó por inmuno-electromicroscopía el virus de la Hepatitis E (VHE) en pacientes con Hepatitis, cuya estructura molecular se conoció en 1990. En 1989, Michael Houghton y el grupo de laboratorios Chiron usando técnicas de biología molecular descubrieron el virus de la Hepatitis C (VHC). Estos hallazgos permitieron conocer mejor estas enfermedades, ideando mejores técnicas de diagnóstico para prevención como producir vacunas y mejores curas para la actualidad [49].

3.5. El Hígado, órgano afectado por la Hepatitis

El Hígado es uno de los órganos más importantes de nuestro cuerpo, se ubica en la parte abdominal superior derecha, debajo del diafragma y por encima del estómago, el riñón derecho y los intestinos es de color marrón rojizo oscuro como se ve en la Figura 3.2 [50], el Hígado mide aproximadamente 26cm x15cm de la parte delantera a la trasera y pesa alrededor de 2 Kg. El hígado regula los niveles sanguíneos de la mayoría de los compuestos químicos y excreta una sustancia llamada, bilis. Algunas de las funciones vitales que cumple el hígado para el correcto funcionamiento de nuestro cuerpo son [51]:

- Producción de bilis: Esta función es necesaria para eliminar los desechos del Hígado como eliminar sustancias dañinas que alteran el organismo como drogas, alcohol y otras suatancias tóxicas para la sangre limpia y descomponer las grasas para desarrollar la digestión de los alimentos.
- Producción de proteínas: Convirtiendo la glucosa en glucógeno como fuente de energía, además regula los niveles de sangre de aminoácidos que son los encargados en la producción de proteínas.
- Procesamiento de la hemoglobina: Este proceso regula los niveles de ph de la sangre, además de almacenar el hierro que contiene y brindar la tarea de oxigenación de los tejidos del cuerpo humano.

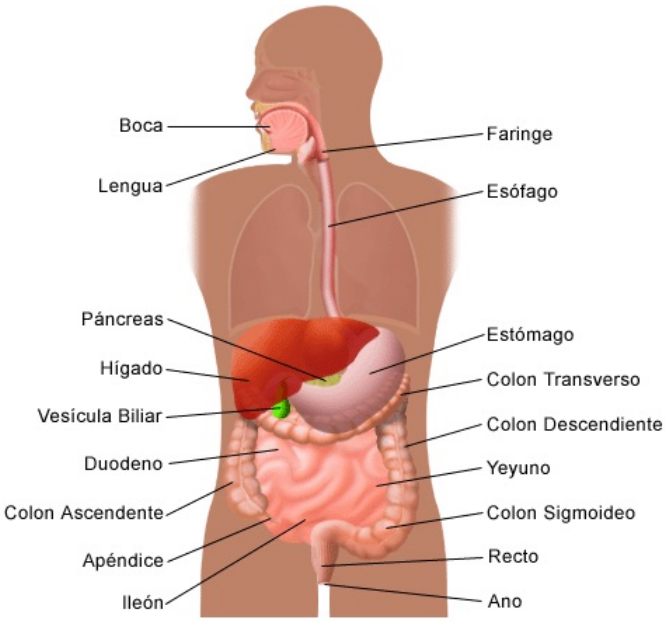

Figura 3.2: Anatomía del hígado en el cuerpo humano
Imagen obtenida de http://www.sharpenespanol.com/healthinfo/content.cfm?pageid=P06162

3.6. Tipos de Hepatitis

3.6.1. Hepatitis viral

La Hepatitis viral es una enfermedad infecciosa que se centra principalmente en el órgano vital llamado Hígado, que es causada por distintos virus que se han designado por letras del abecedario. Esta enfermedad está caracterizada por una inflamación del Hígado [52] provocando lesiones histológicas producidas por los distintos agentes virales. Existen diferencias en el mecanismo de transmisión ya que se lleva a cabo por etapas como el periodo de incubación y la de su evolución. Los virus que son causa de la Hepatitis viral son: VHA, VHB, VHC, entre otros. Existen otros virus en el ambiente que pueden causar daño al Hígado como citomegalovirus, virus del dengue, virus de la fiebre amarilla, etc.

La hepatitis viral se manifiesta en tres fases diferentes en el proceso de la enfermedad como son [53]:

1. **Hepatitis aguda**: Cuando se habla de una hepatitis aguda se refiere a una inflamación del Hígado temporal, esto puede producirse por bacterias, tóxicos, virus, etc. Se divide en cuatro cuadros clínicos: periodo de incubación, fase

preictérica, fase ictérica y período de convalecencia. Pero no siempre se cumplen estas etapas. En el periodo de incubación los pacientes permanecen asintomáticos, es decir, que no presentan síntomas durante 15 o 30 días, casi al término de incubación el paciente puede contagiar a otro individuo de la enfermedad. En la fase preictérica, el paciente presenta síntomas como malestar general, anorexia, náuseas, vómitos y dolor abdominal en la parte superior derecha, otros síntomas según [54] son: acolia (no secreción de bilis), hepatomegalia (aumento del tamaño del hígado), fatiga, pérdida de peso y prurito (picazón), generalmente duran entre 3 y 10 días. En la fase ictérica se presenta coloración amarilla en la piel y coluria que es una coloración oscura en la orina la ictericia se observa en un 20-50 % de los casos de todas las Hepatitis. El periodo de convalecencia es la finalización de la enfermedad hasta alcanzar la recuperación completa de la salud del paciente.

2. **Hepatitis crónica**: También se puede producir por tóxicos o por adicción, autoinmunitaria que es cuando el sistema inmunitario ataca las células del propio organismo en lugar de protegerlo. Otra forma de padecer hepatitis crónica es por acciones en el desequilibrio de los virus y reacciones hereditarias. Se dice que cuando un paciente tiene Hepatitis crónica es por que tiene un aumento de los enzimas del tipo aminoácidos alanina aminotransferasa (ALT) y aspartato aminotransferasa (AST) o de ambas, por más de 6 meses en el Hígado. Cuando se tiene un nivel sérico normal de estas enzimas no se corre el riesgo de complicaciones a largo plazo, de otra manera se producirá en un futuro algunas enfermedades como cirrosis o cáncer.

3. **Hepatitis fulminante**: Es la más peligrosa en la manifestación de la Hepatitis, se define como una insuficiencia hepática aguda, que pone en riesgo la vida y afecta al Hígado para mantener su funcionamiento normal en el transcurso de las 8 semanas siguientes después de la fase ictérica. Se sabe que la hepatitis fulminante puede presentarse en cualquier fase de la enfermedad provocando confusión, transtornos de la memoria, coma, etc.

3.6.2. Virus de la Hepatitis A

También llamada Hepatitis infecciosa, constituye un problema de salud pública mundial ya que es transmitida por vía fecal/oral por la ingesta de alimentos o agua contaminados siendo la más frecuente de hepatitis virales tanto en países desarrollados, como en países en desarrollo. Es una enfermedad que en la mayoría de las veces se mantiene en una Hepatitis aguda, y en muy pocos pacientes puede conllevar a la muerte

en casos de Hepatitis fulminante. Para dar una idea de la forma del VHA se observa en la Figura 3.3 [52]. Las personas que cuentan con un sistema inmunológico resistente pueden combatir y hacer frente a esta enfermedad y posibilita la erradicación del agente infeccioso. El virus de la hepatitis A se elimina en las heces una semana antes del inicio de los síntomas pudiendo infectar a personas que esten en contacto y dos semanas después también es neutralizado tanto por los IgG y los IgM que son anti-VHA que se encuentran dentro del organismo del individuo infectado. El mayor índice de quienes la aportan son los niños de 9 a 12 años de edad.

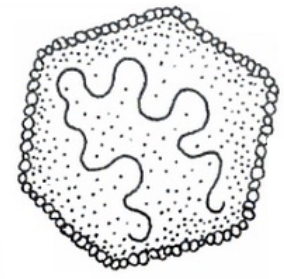

Figura 3.3: Virus de la hepatitis A

Imagen obtenida de [52]

3.6.3. Virus de la Hepatitis B

El VHB tiene una gran difusión en el mundo entero con un elevado porcentaje de personas infectadas, en diferentes zonas geográficas, presenta hepatitis aguda y crónica. Los adolescentes constituyen un grupo de riesgo, principalmente por la posibilidad de transmisión por contacto sexual, adictos que conllevan a inyectarse por vía endovenosa, homosexuales con múltiples parejas, convivencia con personas con el VHB, hemofílicos y por personas que estén expuestos a fluidos corporales, sangre y derivados. Un ejemplo de poder ver a este virus se muestra en la Figura 3.4 [52].

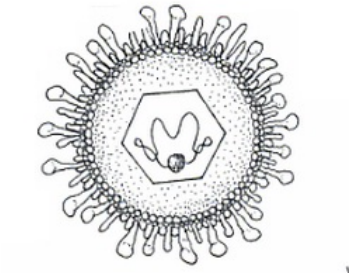

Figura 3.4: Virus de la hepatitis B

Imagen obtenida de [52]

3.6.4. Virus de la Hepatitis C

La Hepatitis C puede presentar hepatitis aguda, pero la más frecuente en esta enfermedad es hepatitis crónica y actualmente se considera entre las principales causas de daño hepático de desarrollo crónico en México. La Hepatitis C es la tercera causa de muerte, debajo de la Diabetes [2], y representa el 1.4 % de la mortalidad total del país. A nivel mundial la hepatitis C es la causa de hepatitis crónica, 170 millones de personas han sido infectadas con este virus. Se transmite de persona a persona y los factores de riesgos son: transfusión de sangre, aplicación de drogas intravenosa, realización de tatuajes y de agujeros en el cuerpo es decir a cualquier exposición de sangre infectada. El virus opera de manera lenta por lo que los datos clínicos de daños hepáticos aparecen después de diez a veinte años después de la infección. Las personas infectadas pueden desarrollar cirrosis, insuficiencia hepática y cáncer de hígado. En la Figura 3.5 [52] se muestra la forma del VHC de manera física.

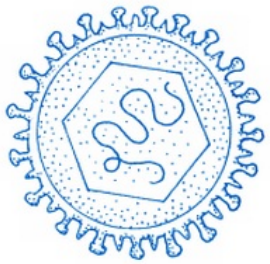

Figura 3.5: Virus de la hepatitis C

Imagen obtenida de [52]

3.7. Conjunto de datos Hepatitis

El conjunto de datos Hepatitis.arff es donado por Gail Gong, de la universidad de Carnegie Mellon, en Noviembre de 1988. Para realizar métodos intensivos de estadística para lograr extraer un conocimiento claro para el diagnóstico de la Hepatitis. Este conjunto de datos fue obtenido a través del diagnóstico de 155 personas para la evaluación y predicción de la enfermedad. El conjunto de datos describe veinte atributos incluyendo el atributo de clase. Los atributos corresponden a síntomas más comunes de la enfermedad, además algunos atributos de niveles de sustancias o sueros que puedan dañar el metabolismo del hígado; como la concentración de altos niveles de bilirrubina, fosfatasa alcalina, ascitis, albúmina etc. Como se puede ver en la Tabla 3.2 éstas concentraciones de sustancias afectan al hígado y se pueden saber realizando

exámenes de sangre en un laboratorio.

En el atributo edad se puede apreciar que es un tipo de dato de tipo entero, que representa la edad del paciente. El atributo Sexo se refiere al género de la persona, este atributo es enumerado. Los atributos Esteriodes, Antivirales, Fatiga, Malestar, Anorexia, Hígado grande, Hígado firme, Bazo palpable etc. hacen referencia a los síntomas comunes de la enfermedad.

Los atributos están relacionados con el padecimiento de la enfermedad y son la Ascitis que es la acumulación de líquido que hay entre los órganos abdominales y el abdomen, lo que puede causar esta acumulación de líquido es por tener Hepatitis del tipo C y B. [55]. Las arañas vasculares son basos sanguíneos cerca de la superficie de la piel que parecen delgadas líneas rojas en la cara o en el cuello. Estas se pueden apreciar en personas con hepatitis crónica o en niños y mujeres embarazadas [56]. El tiempo de protrombina es el tiempo en que la sangre tarda en coagularse y salga un líquido blanco o plasma el tiempo normal para producir este líquido es de 11 a 13 segundos [55].

La bilirrubina es un pigmento amarillo que se encuentra en la bilis y es producido por el hígado. La formación de la bilirrubina es producida en la eliminación de glóbulos rojos viejos por glóbulos nuevos. El hígado ayuda a descomponer la bilirrubina para que después se elimine en las heces y por la orina, las grandes cantidades de bilirrubina pueden ocasionar ictericia. Los niveles de bilirrubina en la sangre son de 0 a 0.3 mg/dl en bilirrubina directa y en bilirrubina total de 0.3 a 1.9 mg/dl, que son nombres alternativos de la bilirrubina.

La fosfatasa alcalina es una proteína que se encuentra en todos los tejidos corporales como hígado, vías biliares y huesos. Muchos medicamentos afectan el nivel de fosfatasa alcalina en la sangre es por eso que en los atributos se encuentran esteroides y antivirales. Los niveles normales de esta sustancia van desde 44 a 147 UI/L (Unidades internacionales por litro).

SGOT (*Serum Glutamic-Oxaloacetic Transaminase*) que es una aminotransferasa, también llamada transaminasa, es un examen de sangre, que arroja causas anomalas si se altera en caso de daño a las células hepáticas. El rango de evaluación de las aminotransferasas pueden ser de ayuda para la evaluación de las enfermedades del hígado como son: Las elevaciones leves: que van de una a 3 veces el valor máximo

normal orienta a hepatitis crónica por virus de la hepatitis B o C. Elevaciones moderadas: van de tres a 10 veces el valor máximo normal puede llevar a hepatitis alcohólica. Evaluaciones marcadas: sobre 10 veces el valor máximo de lo normal provocando hepatitis viral aguda como Hepatitis A o B. [57]

La Albúmina es una proteína producida por el hígado y se concentra en la sangre, los niveles altos de albúmina indican que el paciente sufre de una enfermedad hepática. El rango normal es de 3.4 a 5.4 g/dl. El bazo palpable se da cuando hay un aumento del bazo que es el encargado de filtrar la sangre para mantener los niveles adecuados de glóbulos rojos, blancos y plaquetas, se dice que es palpable cuando el bazo aumenta de 1.5 a 2 veces su tamaño. La histología estudia el tejido hepático y determina si es estable, si es regenerado en respuesta de estímulos externos como lesiones, si presenta procesos tumorales, o cirrosis. Y por último, el atributo de clase que es el que va a determinar de los 155 personas quienes murierón o vivierón teniendo el virus de Hepatitis.

Tabla 3.2: Atributos del conjunto de datos Hepatitis

No.	Atributo	Tipo de dato
1.-	Edad:	Entero (10,20,34,56,78)
2.-	Sexo:	Enumerado (Masculino/Femenino)
3.-	Esteroides:	Enumerado (si, no)
4.-	Antivirales:	Enumerado (si, no)
5.-	Fatiga:	Enumerado (si, no)
6.-	Malestar:	Enumerado (si, no)
7.-	Anorexia:	Enumerado (si, no)
8.-	Hígado grande:	Enumerado (si, no)
9.-	Hígado firme:	Enumerado (si, no)
10.-	Bazo palpable:	Enumerado (si, no)
11.-	Arañas vasculares:	Enumerado (si, no)
12.-	Ascitis:	Enumerado (si, no)
13.-	Varices:	Enumerado (si, no)
14.-	Bilirrubina:	Real (0.39, 0.80, 1.20, 2.00)
15.-	Fosfatasa alcalina:	Entero (33, 80, 120, 160, 200)
16.-	SGOT:	Entero (13, 100, 200, 300)
17.-	Albúmina:	Real (2.1, 3.0, 3.8, 4.5, 5.0, 6.0)
18.-	Tiempo de protrombina:	Entero (10,20,34,56,78)
19.-	Histología:	Enumerado (si, no)
20.-	Clase:	Enumerado (Vivió/ Murió)

Capítulo 4

Pruebas y análisis de desempeño de los algoritmos C4.5, ADTree y SVM para conjuntos de datos Diabetes y Hepatitis

En este capítulo se presentan las configuraciones necesarias para realizar las comparaciones de cada algoritmo de clasificación elegido. Los algoritmos comparados son árboles de decisión (en particular, *ADTree* y *C4.5*) y máquina de soporte vectorial (Entrenada con el método SMO). El entorno de experimentación fue Weka. Al final del capítulo se muestra una implementación propia de una interfaz gráfica de usuario, desarrollada para usar estos clasificadores.

4.1. Configuración del experimento

Las enfermedades Diabetes y Hepatitis presentan varios síntomas para su diagnóstico. En el caso de la Diabetes, ésta es producida por tener un desequilibrio en el cuerpo, en particular, un aumento de azúcar en la sangre. Esto se debe a que la cantidad de insulina que el paciente produce no es suficiente. Para el control de esta enfermedad, se necesita suministrar una hormona llamada insulina. La enfermedad Hepatitis se presenta por diversos virus que se clasifican por letras del abecedario. Cada tipo de virus ocasiona un tipo de Hepatitis. Algo característico de esta enfermedad es que todos los tipos presentan los mismos síntomas: ictericia, vómito, malestar, anorexia, etc. Estos virus dañan al hígado del paciente infectado, comenzando desde una hepatitis aguda hasta una hepatitis fulminante.

En los experimentos realizados, se utilizaron los conjuntos de datos Diabetes.arff y Hepatitis.arff. El primero contiene datos de 768 pacientes, con 9 atributos cada registro. Los registros contenidos son todos de tipo numérico. Este conjunto de datos fue elaborado en el Instituto Nacional de Diabetes y Enfermedades Digestivas y Renales. Puede ser descargado libremente en la página Web de la Universidad de Californa, Santa Cruz [1].

El conjunto de datos Hepatitis.arff está integrado por 155 registros, con 20 atributos cada registro. Los registros de este conjunto de datos son también de tipo numérico, este conjunto de datos fue proporcionado por Gail Gong, de la universidad de Carnegie Mellon, en Noviembre de 1988. Fue básicamente fue creado para la extracción del conocimiento para el diagnóstico de la enfermedad. Puede ser descargado libremente en el sitio Web de la Universidad de Mos, Francia [2].

Para la realización de los experimentos se utilizó Weka 3.7.9 de 32 bits, asignando una cantidad de 256 MB de RAM a la máquina virtual de Java ya que es una herramienta fundamental de Java. Las características de la computadora sobre la que se ejecutaron los experimentos se muestran en la tabla 4.1:

Tabla 4.1: Características del equipo utilizado

Características	**Propiedades**
Procesador:	Intel Celeron de 2.20 GHz
RAM:	1 GB
Sistema Operativo:	Windows 7 Ultimate de 32 bits

Una vista resumida de los conjuntos de datos utilizados se muestra en la Figura 4.1 y en la Figura 4.2. El formato de los datos en archivos arff es explicada en el capítulo dos, donde también se presentó la forma de usar la herramienta Weka para el análisis de los datos.

[1] http://classes.soe.ucsc.edu/cmps142/Winter10/handouts/diabetes.arff
[2] http://informatique.umons.ac.be/ssi/teaching/dwdm/hepatitis.arff

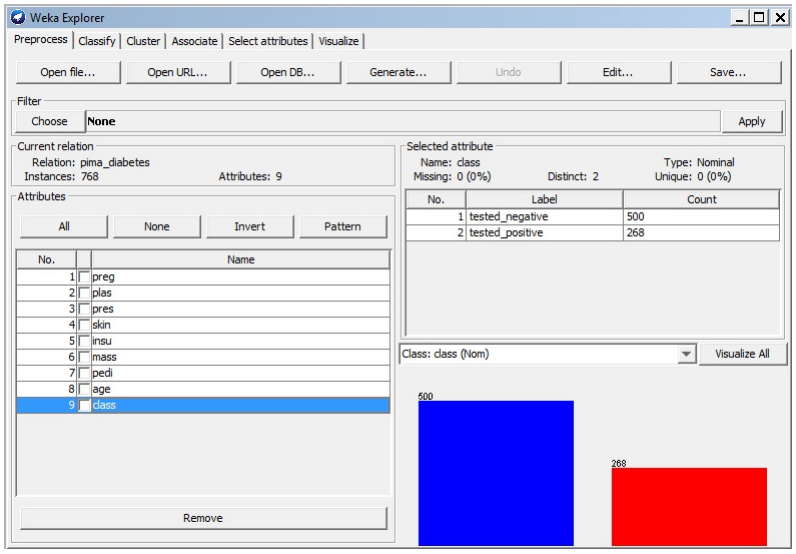

Figura 4.1: Conjunto de datos Diabetes.arff en Weka

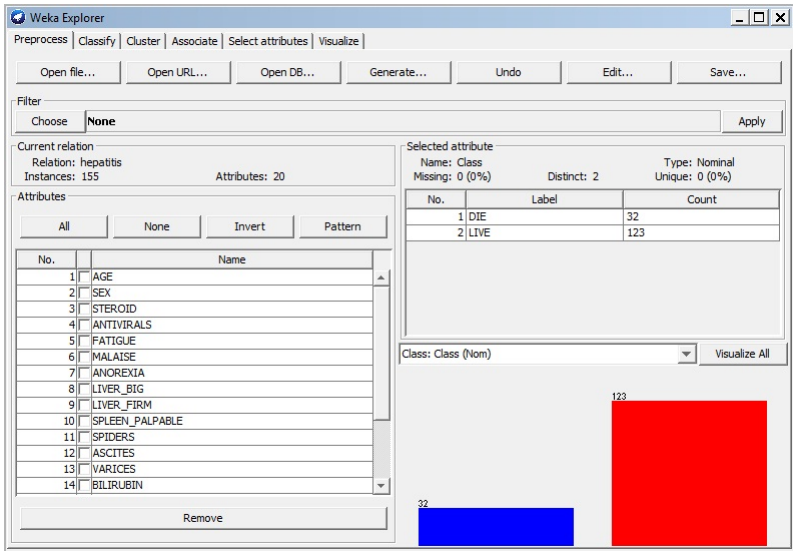

Figura 4.2: Conjunto de datos Hepatitis.arff en Weka

Puede observarse en las mismas Figuras 4.1 y 4.2, que el conjunto de datos diabetes tiene un desbalance moderado, es decir, contiene cantidades similares de datos de cada clase. Sin embargo, el conjunto de datos Hepatitis tiene un desbalance notable, ya que de un tipo de objetos se tiene aproximadamente un 20 % y del otro un 80 %. Para la mayoría de los clasificadores, un desbalance grande representa un problema, ya que la precisión de clasificación alcanzada para la clase con menos patrones es generalmente baja. En la sección correspondiente a experimentos, se mostrará el comportamiento de algoritmos de clasificación seleccionados frente a los dos conjuntos

de datos mencionados anteriormente.

El tipo de prueba de desempeño de clasificación que se aplicó a todos métodos usados en este trabajo es el de validación cruzada (*Cross-validation*). Esta prueba es ampliamente utilizada en inteligencia artificial, minería de datos, aprendizaje automático y reconocimiento de patrones. Se basa en un análisis estadístico, que garantiza la clasificación correcta de datos experimentales por medio de un entrenamiento y evaluación. Esta técnica consiste en hacer una separación en grupos de datos y evaluar un clasificador. Este proceso se realiza un número predeterminado de veces. El número de veces de evaluación de esta prueba en el presente trabajo es de 10, conforme a lo recomendado por la herramienta Weka. En cada una de las iteraciones este método se hace un cálculo de error. Para obtener el error final se obtiene del promedio del número de errores obtenidos, es decir, se realiza la suma de los valores de error y se divide entre el número de errores.

Una medida utilizada para evaluar los clasificadores es Kappa estadística, que fue introducida por Cohen, en 1960 [33]. índice kappa se usa para evaluar la concordancia o reproducibilidad de instrumentos de medida cuyo resultado es categórico.

De acuerdo al criterio de Landis and Koch [58], los valores negativos de esta medida indican un desacuerdo de clasificación, o en otras palabras, un mal clasificador. Los valores de Kappa estadística entre 0–0.20 son considerados bajos, aquellos entre 0.21–0.40 como regulares, 0.41–0.60 como moderados, 0.61–0.80 como substanciales y los valores entre 0.81–1 como clasificación casi perfecta.

A continuación se presentan las pruebas realizadas con estos conjuntos de datos.

4.2. Clasificación

La tarea de clasificación se utiliza para categorizar un objeto en una clase específica de un conjunto previamente conocido. Existen varios algoritmos para hacer la clasificación de datos, que permiten la extracción de conocimiento a partir de datos. A continuación se presentan los resultados aplicando dos tipos de algoritmos de clasificación diferentes, dichos algoritmos son árboles de decisión y máquina de soporte vectorial.

4.2.1. Clasificación con árboles de decisión

En ésta parte se hace una breve explicación del funcionamiento de los algoritmos de *ADTree* y C4.5, así, como los paramétros que utilizan para que sean modificados y puedan arrojar un resultado de presición de clasificación.

4.2.1.1. Clasificación con *ADTree*

ADTree es un algoritmo desarrollado en 1999 por Freund Y. y Mason L. cuando trabajaron sobre aprendizaje automático. El algoritmo realiza la construcción de un árbol de decisión alterno, y optimiza problemas de dos clases. El número de iteraciones debe ajustarse manualmente como uno de los parámetros del algoritmo. Este último controla el equilibrio entre la precisión de clasificación y la complejidad del modelo (tamaño del árbol). Cada iteración agrega 3 nodos en el árbol, un nodo de división y dos nodos de predicción. El método de búsqueda por defecto es la búsqueda exhaustiva, es decir, se expande el árbol por todos los caminos. Algunos métodos de búsqueda heurística se han introducido para acelerar el aprendizaje. En la figura 4.4 se muestra el árbol resultante del conjunto de datos Diabetes, para un valor de número de iteraciones igual a 5.

Para tener un mejor ajuste de precisión en el equilibrio del árbol generado por *ADTree* es importante manipular sus parámetros. En la Figura 4.3 se presentan los parámetros del *ADTree* como aparecen en la interfaz de Weka, mismos que se explican a continuación.

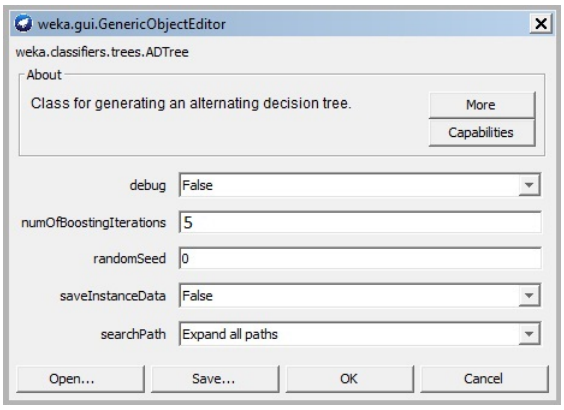

Figura 4.3: Parámetros de *ADTree*

debug: Incluye información para depuración en la consola de resultados si se selecciona su valor en *true*.

numOfBoostingIterations: Establece el número de iteraciones a realizar, mientras más iteraciones se elijan más grande será el árbol generado. (Este parámetro se manipulará para las pruebas realizadas).

randomSeed: Establece un número que sirve de semilla para el generador de números aleatorios, utilizado en la búsqueda que realiza el algoritmo.

saveInstanceData: Establece si el árbol debe guardar los datos en cada nodo.

searchPath: Establece el tipo de búsqueda al crear el árbol. La opción por omisión es expandir por todos los caminos (*Expand all paths*), en donde hará una búsqueda exhaustiva. Los otros métodos son búsqueda heurística, por lo que no garantizan encontrar una solución óptima, pero son mucho más rápidos.

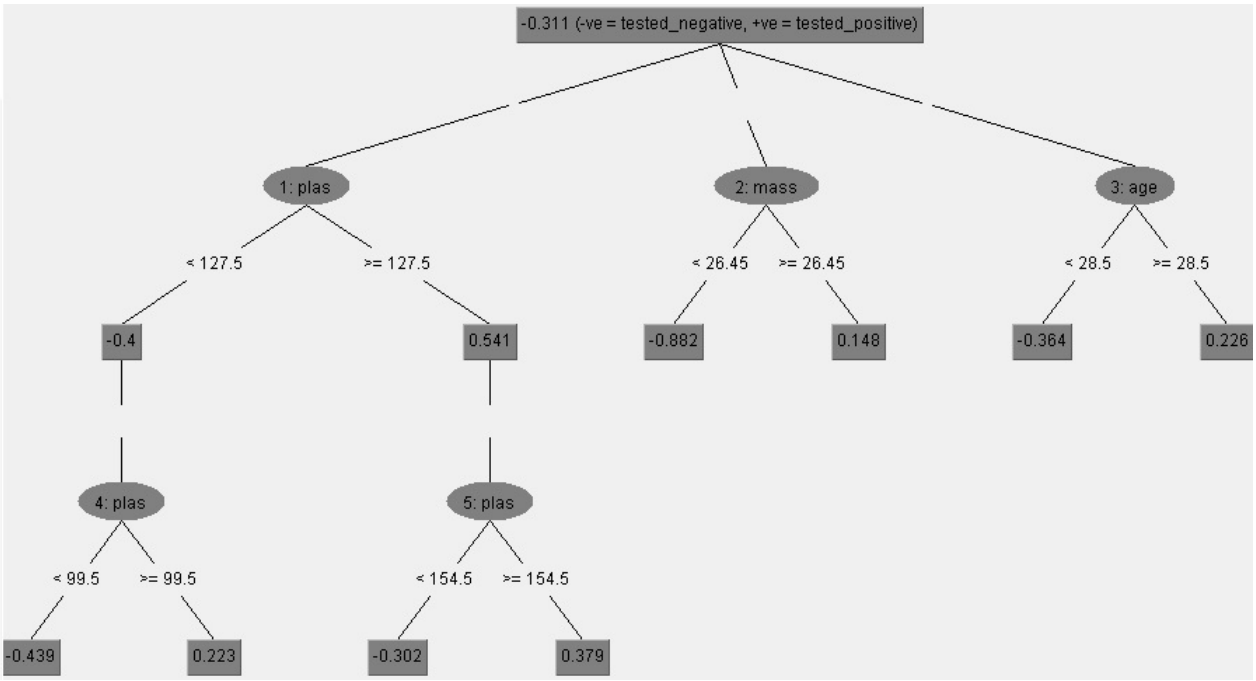

Figura 4.4: *ADTree* del conjunto de datos Diabetes

Utilizando los valores predeterminados de la Figura 4.3, el tiempo para construir el árbol es de 0.08s, más adelante, en los experimentos, se muestran los ajustes del árbol y la precisión de clasificación. Los resultados al aplicarle validación cruzada con 10 particiones y 5 iteraciones para el ajuste del árbol del algoritmo *ADTree* de manera estadística se muestra en la Figura 4.5.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         568               73.9583 %
Incorrectly Classified Instances       200               26.0417 %
Kappa statistic                          0.4074
Mean absolute error                      0.3758
Root mean squared error                  0.4166
Relative absolute error                 82.6765 %
Root relative squared error             87.4029 %
Total Number of Instances              768
```

Figura 4.5: Resultados estadísticos del clasificador *ADTree* del conjunto de datos Diabetes

Como se puede apreciar el número de casos clasificados correctamente es 568, que representa el 73.95 % de precisión de clasificación. Los casos que no se clasificaron correctamente son 200, que representan el 26.04 %. El valor de Kappa estadística encontrado fue de 0.4074, lo que representa una concordancia moderada. Los errores mostrados en la Figura 4.5 se calculan como se indica en la Tabla 4.2. Para esta ejecución, el resultado del promedio del error absoluto es de 0.3758, el error cuadrático medio es de 0.4166, el error absoluto relativo resulta ser 82.67 % y la raíz del error cuadrático relativo y es 87.40 %. Lo que muestra que, estas medidas indican que con los parámetros predeterminados el clasificador obtenido no tiene una precisión aceptable.

Tabla 4.2: Medidas de desempeño para predicción numérica

Medición de Desempeño	**Fórmula**
Promedio del error absoluto	$\frac{\lvert p_1-a_1\rvert+\ldots+\lvert p_n-a_n\rvert}{n}$
Raíz del error al cuadrado	$\sqrt{\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{n}}$
Error absoluto relativo	$\frac{\lvert p_1-a_1\rvert+\ldots+\lvert p_n-a_n\rvert}{\lvert a_1-\overline{a}\rvert+\ldots+\lvert a_n-\overline{a}\rvert}$
Raíz del error cuadrático relativo	$\sqrt{\frac{(p_1-a_1)^2+\ldots+(p_n-a_n)^2}{(a_1-\overline{a})^2+\ldots+(a_n-\overline{a})^2}}$

donde p son valores de predicción y a son valores actuales

Otro dato mostrado en la ventana de resultados de Weka es la matriz de generalización y de recuerdo. En la Figura 4.6 se muestran las precisiones detalladas por clase. En la primera columna de esa tabla se muestra la precisión de datos verdaderos positivos (*TP Rate*), que determinan las precisiones correctas. Un falso positivo (*FP Rate*) se produce cuando el resultado se predijo incorrectamente, es decir, clasifica un valor positivo cuando en realidad es un dato negativo. La tercera columna muestra la precisión alcanzada para cada clase, por ejemplo, con datos de tipo *"tested_positive"*

se alcanza una precisión del 77.9 %. La cuarta columna es la memoria o *recall* del clasificador, su valor aumenta cuando hay pocos falsos negativos. El *recall* mide que las instancias de una clase C se clasifiquen como clase C, aunque otras instancias también se clasifiquen como clase C, sin serlo. Para este experimento el *recall* es de 0.807 para la clase "tested_positive". La *precision* aumenta su valor cuando hay pocos falsos positivos. La medida de datos de entrenamiento (*F-Measure*) es un umbral de probabilidad de salida del clasificador por validación cruzada. La curvatura ROC (*ROC Area*) muestra la curva de valores positivos y los valores negativos mientras más área arroje en los resultados, será mejor el clasificador. En la figura 4.6 muestra los valores de las precisiones detalladas.

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.838     0.444      0.779     0.838     0.807      0.802    tested_negative
                 0.556     0.162      0.648     0.556     0.598      0.802    tested_positive
Weighted Avg.    0.74      0.346      0.733     0.74      0.734      0.802
```

Figura 4.6: Resultados de precisiones detalladas por clase

La matriz de confusión se muestra en la Figura 4.7. Se puede observar que los casos con etiqueta *"tested_positive"* son los más dificiles de clasificar correctamente para el clasificador. Esto se deduce al observar que el modelo comete más errores de clasificación.

```
=== Confusion Matrix ===

   a   b   <-- classified as
 419  81 |   a = tested_negative
 119 149 |   b = tested_positive
```

Figura 4.7: Matriz de confusión

Después de mostrar los resultados del clasificador *ADTree* para el conjunto de datos Diabetes, se presentan ahora los resultados del mismo algoritmo pero ahora con el conjunto de datos Hepatitis. Como se mencionó anteriormente, en este conjunto de datos hay un desbalance notable, y, para la mayoría de los clasificadores la precisión de clasificación alcanzada es baja. Los parámetros utilizados son los mismos en la Figura 4.3. El árbol generado en este caso es mostrado en la Figura 4.8.

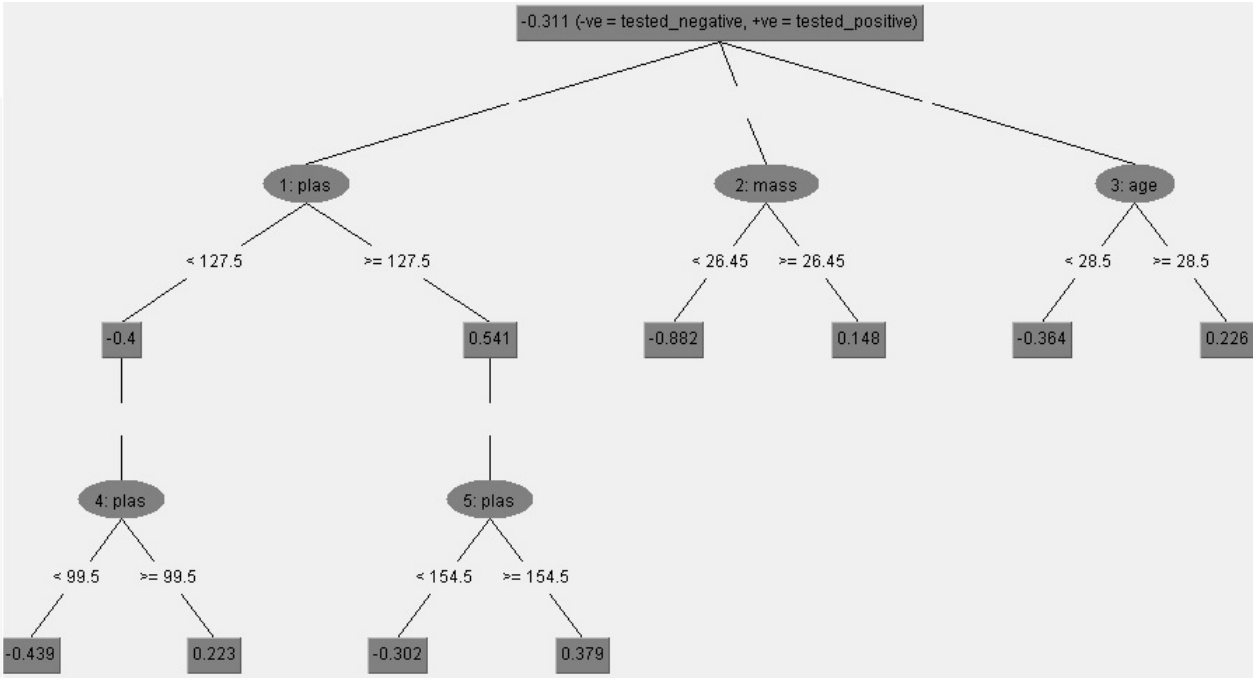

Figura 4.8: Fragmento del árbol del conjunto de datos Hepatitis

El tiempo de entrenamiento para este método fue de 0.03s, con el número de iteraciones del ajuste del árbol igual a 5. Se observó que mientras mayor sea este valor, más lento es el aprendizaje. Los resultados estadísticos se presentan en la Figura 4.9.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         121               78.0645 %
Incorrectly Classified Instances        34               21.9355 %
Kappa statistic                          0.2806
Mean absolute error                      0.2751
Root mean squared error                  0.3741
Relative absolute error                 83.3047 %
Root relative squared error             92.388  %
Total Number of Instances              155

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.375     0.114      0.462     0.375     0.414      0.787    DIE
                 0.886     0.625      0.845     0.886     0.865      0.787    LIVE
Weighted Avg.    0.781     0.519      0.766     0.781     0.772      0.787

=== Confusion Matrix ===

   a   b   <-- classified as
  12  20 |   a = DIE
  14 109 |   b = LIVE
```

Figura 4.9: Resultados estadísticos del clasificador *ADTree* del conjunto de datos Hepatitis

De manera general, se puede decir que el número de instancias correctamente clasificadas fueron 121, lo que representa el 78.06 %. Por otra parte, las instancias que no fueron clasificadas correctamente son 34, que representa el 21.93 %. El promedio del error absoluto alcanza el 0.2751 y el error absoluto relativo es de 83.30 %.

Los resultados mostrados en la matriz de confusión indican que los elementos clasificados correctamente en cada una de sus clases son: 12 elementos clasificados correctamente en la clase morir (*Die*), 119 fueron clasificados en la clase vivir (*Live*). De esta manera se puede observar, que la clase que muestra problemas de clasificación es la clase morir, esto sucede por la mínima cantidad de datos que presenta este conjunto de datos.

4.2.1.2. Efecto de la variación del parámetro *numOfBoostingIterations* sobre el desempeño de *ADTree*

En los conjuntos de datos Diabetes y Hepatitis se realizó el entrenamiento de *ADTree* con un ajuste de 5 iteraciones. A continuación se modificará el parámetro de*numOfBoostingIterations* incrementándolo en 5 unidades por cada experimento, es decir, en la siguiente evaluación se hará con *numOfBoostingIterations* igual a 10 y la

tercera evaluación se hará con 15 y así sucesivamente hasta llegar al valor 100. Los resultados se presentan de manera gráfica. El objetivo de ir aumentando el valor del parámetro *numOfBoostingIterations* es para observar como se comporta este algoritmo. Primero se mostrarán las gráficas de resultados correspondientes al conjunto de datos Diabetes y después la del conjunto de datos Hepatitis.

En la figura 4.10 se muestra el porcentaje de precisión de clasificación del algoritmo del conjunto de datos diabetes.

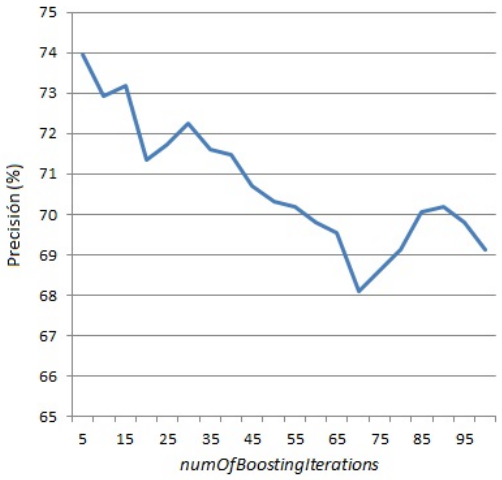

Figura 4.10: Precisión del algoritmo *ADTree* del conjunto de datos Diabetes

Como se puede observar en la figura 4.10, la precisión de clasificación disminuye al ir aumentando el número de iteraciones, se puede apreciar que una precisión alta ocurre en el valor *numOfBoostingIterations* igual a 5 y también en el valor 15. Se puede ver también que la precisión de clasificación más baja es con el valor de 70. En la Figura 4.11 se presenta el aumento del tamaño del árbol *ADTree* tanto en sus nodos como en sus hojas, al ser modificado el parámetro *numOfBoostingIterations*.

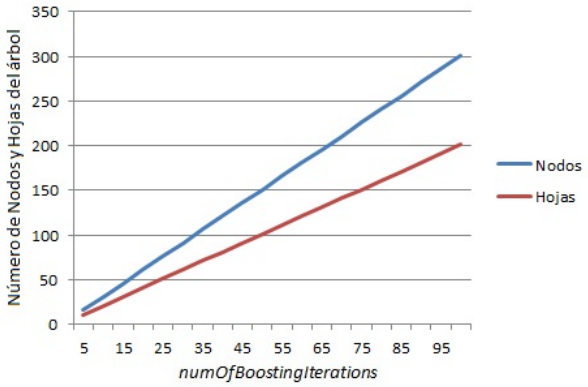

Figura 4.11: Aumento del *ADTree* en nodos y hojas del conjunto de datos Diabetes

Se puede observar en la figura 4.11, que por cada número de iteraciones hay un aumento en el árbol, la línea azul indica el aumento de nodos y la línea roja el aumento de las hojas. Mientras más iteraciones, más grande se hace el árbol para obtener un mejor resultado de precisión de clasificación. El tiempo que se lleva acabo en la construcción de cada árbol por cada iteración se muestra en la Figura 4.12.

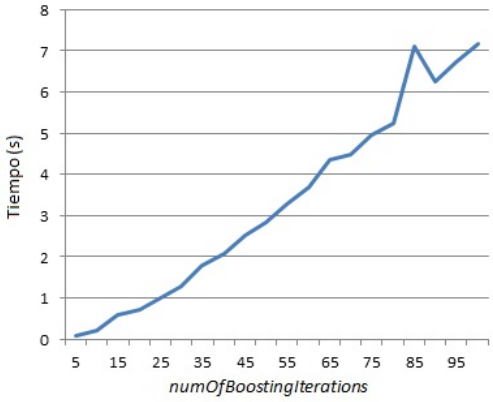

Figura 4.12: Tiempo en segundos de creación del *ADTree* del conjunto de datos Diabetes

En la figura 4.12 se ve que el tiempo aumenta al ir elevando el valor del parámetro *numOfBoostingIterations,* como era de esperarse. Con 80 iteraciones se aprecia que hay un brinco considerable en el tamaño del árbol, sin embargo, el crecimiento es notorio en el resto de la gráfica. Este cambio repentino en el crecimiento del árbol fue analizado, sin encontrarse una explicación satisfactoria.

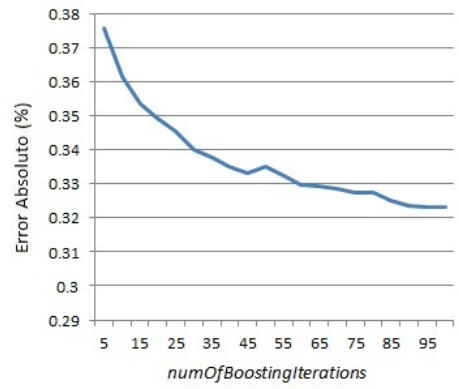

Figura 4.13: Promedio del error absoluto del *ADTree* del conjunto de datos Diabetes

En la figura 4.13 se puede ver que, mientras más grande sea el número de iteraciones para el ajuste del árbol, el promedio del error absoluto disminuye. Se observa que en los primeros valores de modificación del parámetro *numOfBoostingIterations* el error es alto y se estabiliza al manipular los valores del parámetro en valores arriba de 60, ajustándose en un rango entre 0.32 % y 0.33 % de precisión de clasificación. A continuación se mostrarán las gráficas de resultados para el conjunto de datos Hepatitis.

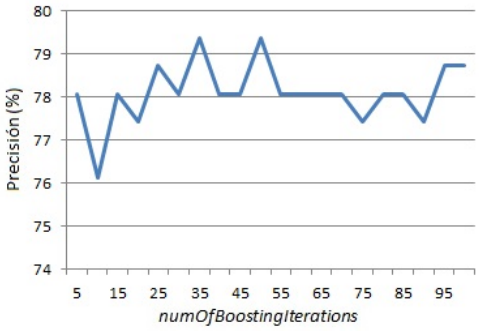

Figura 4.14: Precisión del algoritmo *ADTree* del conjunto de datos Hepatitis

En la figura 4.14 se puede ver que la precisión del algoritmo *ADTree* del conjunto de datos Hepatitis está en un rango de 77 % a un 79 %. La precisión de clasificación se mantiene constante, sólo hay dos aumentos en la precisión en los valores del parámetro parámetro *numOfBoostingIterations* el primero cuando el valor es 35 y el segundo cuando el valor se encuentra en 50, teniendo como resultado un valor de precisión de clasificación de un 83 %.

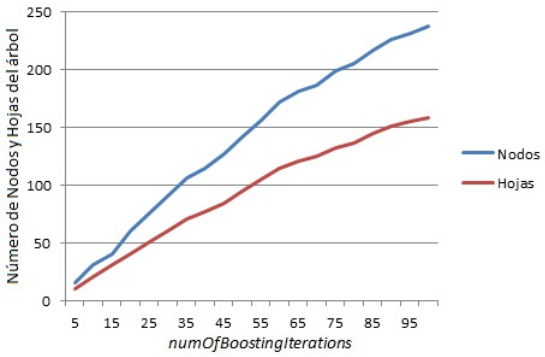

Figura 4.15: Aumento de *ADTree* en nodos y hojas del conjunto de datos Hepatitis

Como se observa en la figura 4.15, el árbol va en aumento por cada valor manipulado del parámetro *numOfBoostingIterations.* La línea azul representa el aumento de los nodos y la línea roja representa el aumento de las hojas del árbol para obtener una precisión de clasificación favorable.

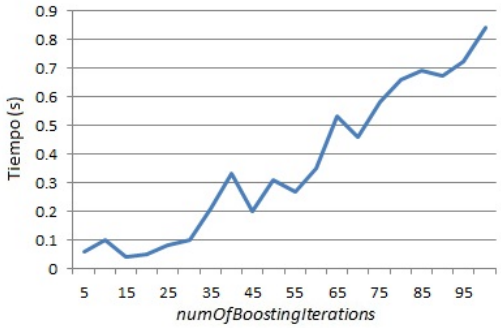

Figura 4.16: Tiempo en segundos de creación del *ADTree* del conjunto de datos Hepatitis

En la figura 4.16 se observa que al modificar el parámetro *numOfBoostingIterations* el tiempo de aprendizaje muestra algunas variaciones por cada valor modificado, estos cambios en las variaciones de tiempo pueden ser por el tamaño de datos que tiene el conjunto de datos Hepatitis. Pero como es de esperarse, el aprendizaje se hace más lento en los últimos valores del parámetro *numOfBoostingIterations.*

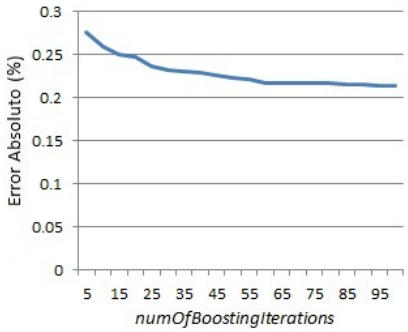

Figura 4.17: Promedio del error absoluto del ADTree del conjunto de datos Hepatitis

El promedio del error absoluto disminuye, como se puede ver en la Figura 4.17. En un principio se encuentra con valores altos de error con valores entre 5 y 15 del parámetro *numOfBoostingIterations*, después es estabilizado con en un rango entre 0.2 % y 0.25 % con los demás valores de manipulación del parámetro.

4.2.1.3. Comentarios sobre el desempeño de *ADTree*

El desempeño que tiene el algoritmo *ADTree* en los conjuntos de datos Diabetes y Hepatitis es diferente. El primero presenta 768 instancias de las cuales indican que 500 no tienen Diabetes y 268 personas con Diabetes, en pocas palabras, este conjunto de datos está balanceado. En el caso del conjunto de datos Hepatitis tiene 155 instancias, de las cuales 123 personas que tienen Hepatitis sobrevivieron y 32 personas murieron, se puede ver que en realidad son pocas instancias y hay un desbalance de personas que murieron con un 20 % y un 80 % de personas que sobrevivieron. En general, puede observarse que *ADTree* no es un buen clasificador para conjuntos de datos desbalanceados.

4.2.1.4. Clasificación con C4.5

El algoritmo C4.5 desarrollado por Ross Quinlan, en 1993, es probablemente el árbol de decisión más utilizado para problemas de clasificación. El valor de confianza predeterminado en Weka se establece en 25 % y funciona razonablemente bien en la mayoría de los casos. Posiblemente debería ser alterado a un valor más bajo lo que hace que la poda sea más drástica. Hay otro parámetro importante en este algoritmo, cuyo efecto es la eliminación de las pruebas para la que casi todos los ejemplos de entrenamiento tienen el mismo resultado, estas pruebas suelen ser de poca utilidad. Por lo consiguiente, las pruebas no están incorporadas en el árbol de decisión a menos que tengan dos resultados en un número pequeño de instancias. El valor por defecto

de este es 2, pero es controlable y quizás debe aumentarse para las tareas que tienen una gran cantidad de datos ruidosos [33].

Para ser utilizado, este tipo de algoritmo, en Weka se tiene que seleccionar el algoritmo J48, que es una implementación del algoritmo C4.5, los parámetros de este algoritmo se muestran en la figura 4.18 mismos que se explican a continuación.

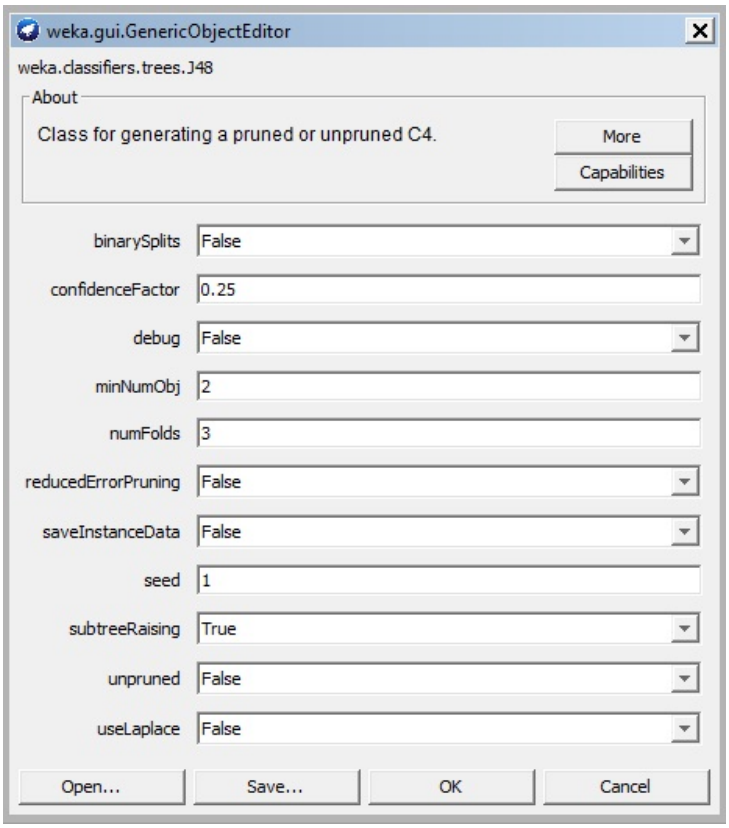

Figura 4.18: Parámetros de C4.5

binarySplits: Si se selecciona en *true*, cada nodo puede tener solamente dos hijos. (Por omisión esta en *false* y así se mantendrá al momento de hacer pruebas).

confidenceFactor: El factor de confianza utilizado para la poda. Como valor establecido es el 0.25 que representa el 25 % . Si se le da un valor menor, el algoritmo hará más poda en el árbol. (Este parámetro se mantendrá con su valor establecido para las pruebas).

debug: Si se establece en *true*, el clasificador puede incluir información adicional a la consola de resultados de Weka.

minNumObj: El valor mínimo de instancias por hoja. (Este valor será manipulable en las pruebas para la eliminación de ruido).

numFolds: Determina la cantidad de datos que se utilizan para la poda reduciendo el error.

reducedErrorPruning: Da la opción de que se utilice la poda de reducción de errores.

saveInstanceData: Da la opción de guardar los valores de entrenamiento para verlos en otro momento.

seed: Establece una semilla para que los datos sean aleatorios para aplicarle poda de reducción.

subtreeRaising: Indica si se considera la operación de elevación de subárbol al podar.

unpruned: Establece si se realiza la poda.

A continuación se presentan los resultados de la aplicación de C4.5 sobre los conjuntos de datos Diabetes y Hepatitis. La Figura 4.19 muestra el resultado presentado en Weka cuando el parámetro *minNumObj* tiene valor 5.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         573               74.6094 %
Incorrectly Classified Instances       195               25.3906 %
Kappa statistic                          0.4318
Mean absolute error                      0.316
Root mean squared error                  0.4308
Relative absolute error                 69.5357 %
Root relative squared error             90.3918 %
Total Number of Instances              768

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.824     0.399      0.794     0.824     0.809      0.775    tested_negative
                 0.601     0.176      0.647     0.601     0.623      0.775    tested_positive
Weighted Avg.    0.746     0.321      0.742     0.746     0.744      0.775

=== Confusion Matrix ===

   a   b   <-- classified as
 412  88 |   a = tested_negative
 107 161 |   b = tested_positive
```

Figura 4.19: Resultados arrojados del algoritmo C4.5 del conjunto de datos Diabetes

Al manipular el valor del parámetro *minNumObj* con un valor inicial de 5, el resultado obteniendo en la precisión de clasificación es de 573, que representa el 74.61 % de instancias clasificadas correctamente y 195 que representa el 25.39 % de instancias clasificadas incorrectamente. El promedio del error relativo es de 0.316 y el error absoluto relativo es de 69.53 %. Como se puede observar la clase *tested_positive* presenta problemas de clasificación.

La matriz de confusión muestra que 412 datos fueron clasificados en la clase *tested_ negative*, 88 datos no fueron considerados o no pertenecían para esa clasificación, en la clase *tested_ positive* fueron clasificados correctamente 161 datos pero 107 no fueron encontrados pertenecientes a esa clase. La Figura 4.20. muestra un fragmento del árbol correspondiente a los resultados generados por el algoritmo C4.5

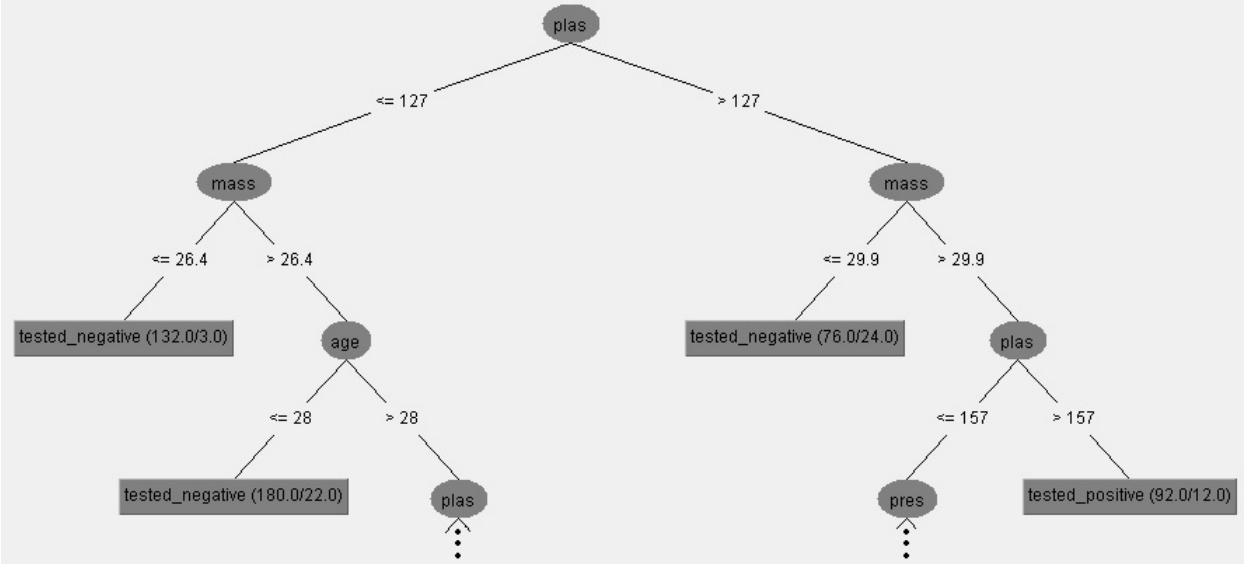

Figura 4.20: Fragmento del árbol C4.5 del conjunto de datos Diabetes

Por cada atributo del conjunto de datos Diabetes, el algoritmo C4.5 elige el más eficaz y divide el conjunto de muestras en subconjuntos enriquecidos de una clase u otra. Su criterio se basa en medidas de entropía. En resumen por cada atributo que el algoritmo elija como el parámetro de mayor ganancia y se elige como parámetro de decisión y se divide en dos partes sucesivamente como se ve en la Figura 4.20.

A continuación se muestran los resultados estadísticos del conjunto de datos Hepatitis, con un valor inicial del parámetro *minNumObj* igual a 5.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         123               79.3548 %
Incorrectly Classified Instances        32               20.6452 %
Kappa statistic                          0.2044
Mean absolute error                      0.2498
Root mean squared error                  0.3852
Relative absolute error                 75.6406 %
Root relative squared error             95.1277 %
Total Number of Instances              155

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.219     0.057      0.5       0.219     0.304      0.678    DIE
                 0.943     0.781      0.823     0.943     0.879      0.678    LIVE
Weighted Avg.    0.794     0.632      0.756     0.794     0.76       0.678

=== Confusion Matrix ===

   a   b   <-- classified as
   7  25 |   a = DIE
   7 116 |   b = LIVE
```

Figura 4.21: Resultados arrojados del algoritmo C4.5 del conjunto de datos Hepatitis

Como se describe en la Figura 4.21, el parámetro *minNumObj* tiene un valor inicial de 5, con este valor se obtiene de manera general un número de instancias clasificadas correctamente de 123, que representa el 79.35 % y un número de instancias clasificadas incorrectamente de 32, que representa el 20.64 %. El promedio del error absoluto alcanza el 0.2498, y el error absoluto relativo es de 75.64 %. La clase quepresenta más errores de clasificación es la clase morir.

El resultado de la matriz de confusión indica los elementos clasificados correctamente por clase, como se puede ver en la Figura 4.21. Un total de 7 elementos fueron clasificados correctamente en la clase de morir, y 116 en la clase de vivir. La matrix de confusión también muestra los datos que no fueron clasificados correctamente, que es lo restante de los elementos de cada clase. Un número de 25 elementos no fueron considerados para pertenecer a la clase de morir, y 7 no fueron clasificados en la clase de vivir. En la Figura 4.22 se presenta un fragmento del árbol generado con el algoritmo C4.5 cuando es aplicado al conjunto de datos Hepatitis.

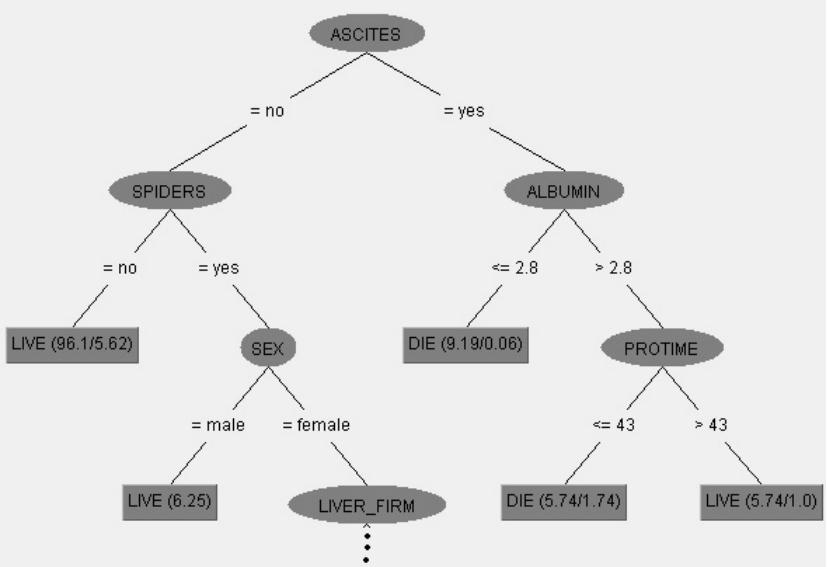

Figura 4.22: Fragmento del árbol C4.5 del conjunto de datos Hepatitis

4.2.1.5. Efecto de la variación de parámetro *minNumObj* sobre el desempeño de C4.5

De manera similar al procedimiento efectuado anteriormente, en este experimento se modifica el parámetro *minNumObj*, empezado desde un valor igual a 5. Se incrementará el valor de este parámetro en 5 unidades por cada experimento, es decir, en la siguiente evaluación se hará con *minNumObj* igual a 10 y la tercera evaluación se hará con *minNumObj* igual a 15, y así sucesivamente hasta llegar al valor 100. Esto con el objetivo de examinar la influencia de este valor en la precisión de clasificación del algoritmo. Como primer resultado se muestra la Figura 4.23, donde se observa la precisión de clasificación del algoritmo C4.5 con el conjunto de datos Diabetes.

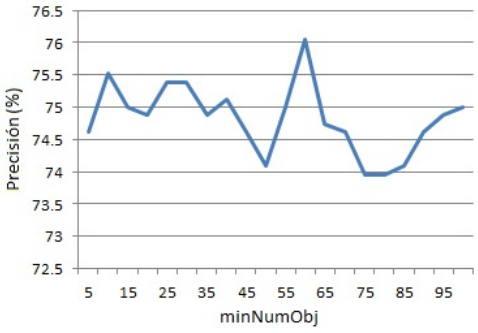

Figura 4.23: Precisión de clasificación del árbol C4.5 aplicado al conjunto de datos Diabetes

En la Figura 4.23 se puede observar que la precisión de clasificación alcanzada

presenta variaciones al ir aumentando el valor del parámetro *minNumObj* . Una buena precisión de clasificación se obtiene con un valor de 60, obteniendo como resultado un 76 %. El rango de precisión de clasificación se mantiene entre un mínimo de 74 % y un máximo de 75.5 % para los otros valores probados.

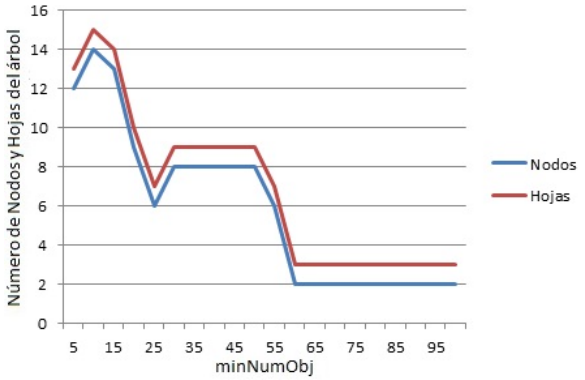

Figura 4.24: Poda en nodos y hojas del árbol C4.5 para el conjunto de datos Diabetes

La Figura 4.24 muestra la poda generada en nodos y hojas por cada iteración del algoritmo C4.5. Como se observa en la Figura 4.24, cuando el valor del parámetro *minNumObj* es menor, el árbol generado es grande, al ir aumentando el valor de este parámetro, el árbol elimina nodos que no son necesarios. En la misma Figura 4.24, la línea azul representa los nodos y la línea roja representa las hojas.

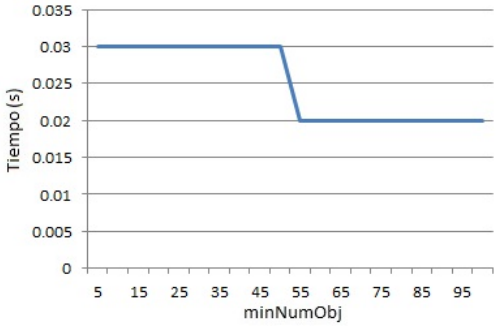

Figura 4.25: Tiempo en segundos de creación de C4.5 del conjunto de datos Diabetes

El tiempo de entrenamiento para el C4.5 es casi constante, y se considera rápido para este conjunto de datos. En la Figura 4.25 se puede observar que el tiempo producido se encuentra en un rango entre 0.02s a 0.03s. Deducimos que esta es una de las razones por las que este algoritmo es uno de los más ampliamente utilizados.

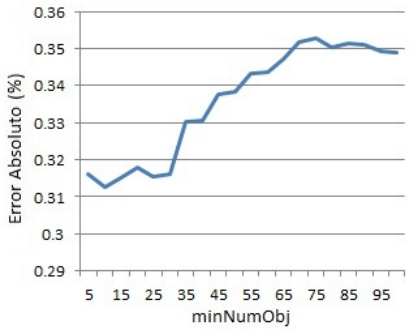

Figura 4.26: Promedio del error absoluto de C4.5 del conjunto de datos Diabetes

En la Figura 4.26 se puede observar que el promedio del error absoluto es el más pequeño si el valor del parámetro *minNumObj* es entre los valores 5 a 30. Al ir aumentando el valor de este parámetro, el promedio del error absoluto es aumentado y se mantiene estable con valores entre 70 a 100.

A continuación se muestran las pruebas realizadas al algoritmo C4.5 usando el conjunto de datos Hepatitis. En la Figura 4.27 se puede observar la precisión de clasificación del algoritmo C4.5 para el conjunto de datos mencionado anteriormente.

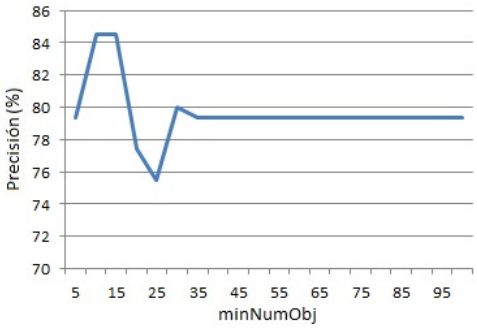

Figura 4.27: Precisión del árbol C4.5 del conjunto de datos Hepatitis

Al variar los valores del parámetro *minNumObj*, se obtiene una alta precisión de clasificación (79 % a 84 %) para valores bajos, menores a 25. Con valores del parámetro mayores a 30, se obtiene una precisión de clasificación menor, que, sin embargo, se mantiene casi constante: entre un 78 % y 80 %.

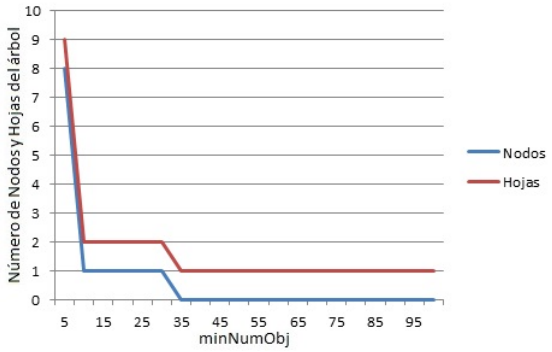

Figura 4.28: Poda de C4.5 en nodos y hojas del conjunto de datos Hepatitis

De igual manera, como sucedió en el conjunto de datos Diabetes, el árbol con un valor mínimo en el parámetro *minNumObj* es de tamaño grande. Al aumentar el valor del parámetro *minNumObj*, disminuy es tamaño del árbol.

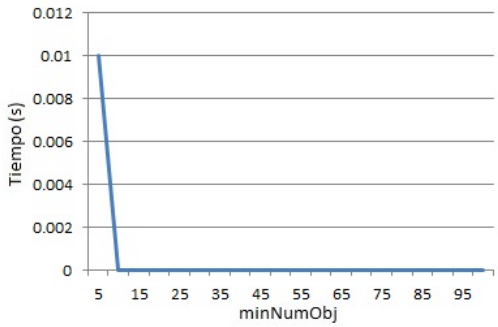

Figura 4.29: Tiempo en segundos de creación de C4.5 del conjunto de datos Hepatitis

Se puede observar que con un valor de 5 en el parámetro *minNumObj*, el tiempo de entrenamiento es de 0.01s. Para valores más grandes del parámetro, los tiempos de entrenamiento decaen considerablemente. Esto significa que el algoritmo realizó el aprendizaje de manera muy rápida. Esto es de esperarse, ya que considerar más objetos por nodo, se realiza menos procesamiento.

El promedio del error absoluto se observa en la figura 4.30. Donde muestra que con un valor de 5 en el parámetro *minNumObj*, el resultado del promedio del error absoluto es de 0.25 %. Cuando es aumentado el valor del parámetro *minNumObj*, el resultado del promedio del error absoluto es incrementado hasta llegar a un valor de 0.33 %. Al incrementar el valor a 40 del parámetro *minNumObj*, se mantiene casi constante el promedio del error absoluto, quedando en un rango de 0.3 % a 0.35 %.

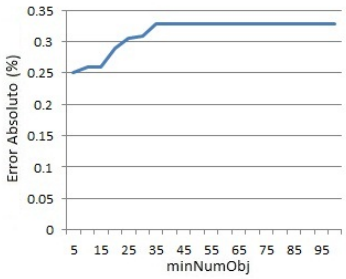

Figura 4.30: Promedio del error absoluto de C4.5 del conjunto de datos Hepatitis

4.2.1.6. Comentarios sobre el desempeño de C.45

A diferencia del algoritmo *ADTree,* C4.5 no hace crecer el árbol de decisión, sólo lo reduce utilizando una técnica de poda para eliminar datos que causan ruido o incongruencias para hacer la clasificación. Se puede apreciar que la precisión de clasificación adquirida del conjunto de datos Diabetes no disminuye, más bien, se mantiene en un rango entre 74 % a 75.5 % al ir aumentando el valor del parámetro *minNumObj* como se puede ver en la Figura 4.23, en el conjunto de datos Hepatitis en las primeras iteraciones al ir aumentando el valor del parámetro *minNumObj* se puede observar que se genera un desequilibrio notable y después se vuelve constante con una misma precisión, es decir, los valores de los resultados ya no cambian y se mantienen de forma lineal como se ve en la Figura 4.27. Conforme es aumentado el valor del parámetro *minNumObj*, el árbol generado es disminuido, ocasionando un rápido aprendizaje para lograr la precisión de clasificación de los conjuntos de datos Diabetes y Hepatitis, de igual manera se puede apreciar que se comporta de forma idéntica al arrojar los resultados del promedio del error absoluto ya que este algoritmo tiende a generar mas errores de clasificación para estos dos conjuntos de datos.

4.2.2. Clasificación con máquina de soporte vectorial (SVM)

Para el uso de SVM, se eligió el algoritmo llamado optimización mínima secuencial (SMO). Este algoritmo fue desarrollado por John Platt en 1998, es ampliamente utilizado en aplicaciones reales [59, 60].

Para seleccionar SVM usando SMO en Weka, se tiene que elegir en las opciones de clasificación en la carpeta de *functions* y dar clic en SMO, los parámetros configurables por el usuario son presentados en la Figura 4.31.

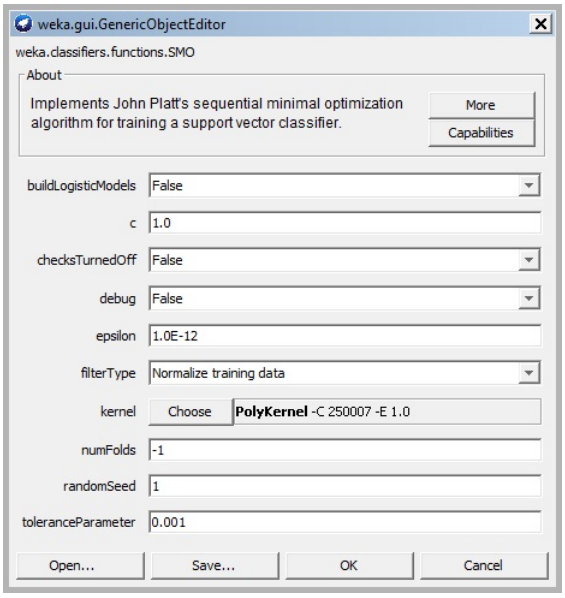

Figura 4.31: Parámetros de SVM entrenada con SMO

buildLogisticModels: Se utiliza para adaptarse a modelos logísticos para las salidas. (Este parámetro se mantendrá en *false* para las pruebas realizadas para los conjuntos de datos Diabetes y Hepatitis).

c: Es el parámetro de complejidad C. (Este parámetro se ajustará durante las pruebas realizadas).

checksTurnedOff: Este parámetro se tiene que utilizar con precaución ya que se activa cuando los controles requieren de mucho tiempo para arrojar un resultado, de esta manera se mantendrá en *false*.

debug: Si se selecciona en *true* puede incluir información a la consola de resultados.

epsilon: Controla la tolerancia de error. (No se cambia en los experimentos).

FilterType: Determina si se cambian los datos de forma estándar o normalizados.

kernel: Se selecciona el tipo de Kernel a utilizar. (Se utilizará para las pruebas PolyKernel y RBFKernel).

numFolds: Número de particiones para realizar la validación cruzada. (El -1 indica la utilización de datos de entrenamiento, no hay cambio durante las pruebas).

randomSeed: Semilla de números aleatorios para validación cruzada. (No hay cambios para las pruebas).

toleranceParameter: Parámetro de tolerancia. (No debe ser cambiado, según la documentación de Weka).

4.2.2.1. Efecto de la variación del parámetro C sobre el desempeño de SVM

En las siguientes pruebas se modificó el parámetro c iniciándolo, con un valor de 5 en la primera prueba y aumentándolo en esa misma cantidad hasta llegar al valor de 100. Se probaron dos tipos de funciones *kernel* para medir la precisión de clasificación en cada conjunto de datos, dichas funciones son la polinomial (PolyKernel) y la de base radial (RBFKernel).

Los resultados obtenidos son presentados de manera gráfica, como se realizó anteriormente para los árboles. La gráfica de la Figura 4.32 muestra los resultados de precisión de clasificación del conjunto de datos Diabetes, utilizando las funciones kernel antes mencionadas. El parámetro γ se mantuvo con el valor predeterminado, que es $1/n$. Donde n es el número de patrones.

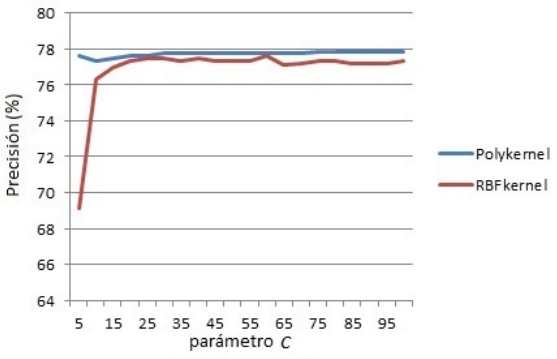

Figura 4.32: Precisión de SVM entrenada con SMO del conjunto de datos Diabetes

En la Figura 4.32 se puede observar que el valor del parámetro c modifica la precisión de clasificación, la función polinomial permite obtener valores más altos de precisión de clasificación que la función de base radial. Para valores altos del parámetro c se puede decir que la SVM se comporta casi igual.

En la Figura 4.33 muestra el tiempo empleado en cada prueba para cada valor del parámetro c.

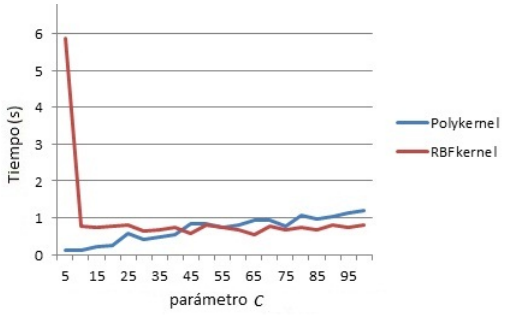

Figura 4.33: Tiempo en segundos de clasificación de SVM entrenada con SMO del conjunto de datos Diabetes

En la Figura 4.33 se puede observar que el tiempo de entrenamiento parece ser dependiente del valor del parámetro c, la relación es casi lineal con una constante pequeña. Entre mayor sea el valor del parámetro c, se penalizan más los errores de clasificación, por lo que el algoritmo tarda más tiempo en encontrar la solución.

La función utilizada como kernel también fue variada en los experimentos, para observar el efecto en la clasificación. Se observó que cuando se usa la función Polykernel el entrenamiento es independiente del parámetro c, mientras que con la función RBFKernel el tiempo de entrenamiento de la SVM aumenta linealmente con el valor de c.

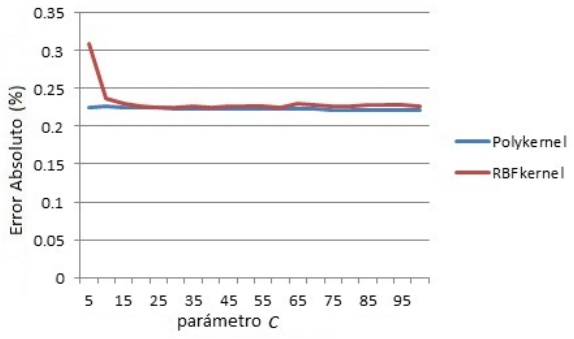

Figura 4.34: Promedio del error absoluto de SVM entrenada con SMO del conjunto de datos Diabetes

En la Figura 4.34 se muestra el error generado para cada valor del parámetro c. El promedio del error absoluto es disminuido ligeramente conforme se aumenta el valor de c. En general, puede decirse que el promedio de error absoluto de clasificación es bajo con SVM. La función Polykernel es la que presenta mejores resultados de clasificación.

A continuación se muestran los resultados obtenidos del conjunto de datos Hepatitis. Los resultados de precisión de clasificación se obtienen al ir ajustando el parámetro c

como se ve en la Figura 4.35.

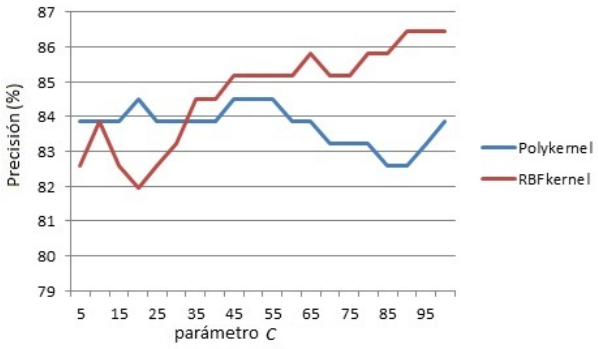

Figura 4.35: Precisión de SVM entrenada con SMO del conjunto de datos Hepatitis

La precisión de clasificación muestra una diferencia notable dependiendo del tipo de kernel utilizado, y del valor del parámetro c.

Con la función de base radial se mejora la precisión de clasificación cuando el valor de c aumenta. Con la función Polykernel la precisión de clasificación se mantiene constante en un rango entre un 82 % a un 85 % para valores de c que se encuentran entre 5 y 45. Para valores de c mayores a 50 y usando la función Polykernel, el clasificador no alcanza mejores resultados.

El tiempo de entrenamiento de la SVM se muestra en la gráfica 4.36.

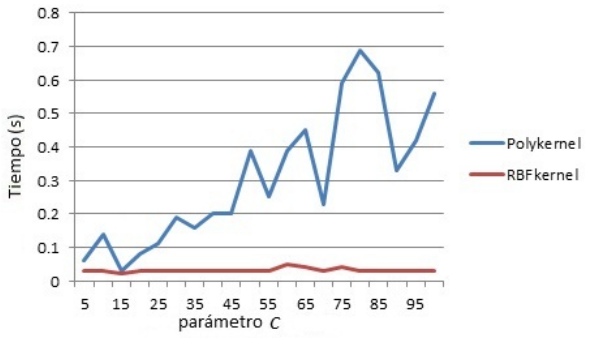

Figura 4.36: Tiempo en segundos de clasificación de SVM entrenada con SMO del conjunto de datos Hepatitis

El tiempo de entrenamiento al usar el tipo de *kernel* Polykernel es más grande que al usar la función de base radial. Se puede observar en la gráfica de la Figura 4.36 que el parámetro c afecta el tiempo de entrenamiento de la SVM cuando se usa una función polinomial.

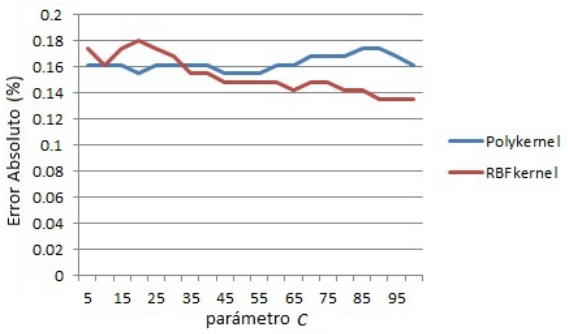

Figura 4.37: Promedio del error absoluto de SVM entrenada con SMO del conjunto de datos Hepatitis

En la Figura 4.37 se muestra el promedio del error absoluto obtenido para cada valor del parámetro *c*. Con la función Polykernel, el error aumenta conforme es aumentado el valor del parámetro *c*. Con la función de base radial, el promedio del error inicia con valores altos, y después disminuye. Se puede decir que este kernel es favorable para clasificación del conjunto de datos Hepatitis.

4.2.2.2. Comentarios sobre el desempeño de SVM

Para el conjunto de datos Diabetes, la función de *kernel* que arrojan valores altos de precisión de clasificación al ser aumentado el parámetro c es la función polinomial. En el caso del conjunto de datos Hepatitis, el tipo de *kernel* que arroja buenos resultados, a pesar que es un conjunto de datos desbalanceado, es la función de base radial. El tiempo de entrenamiento es más rápido y resulta ser casi independiente del valor del parámetro c con este tipo de kernel, además, los valores del promedio del error absoluto son más bajos que con el otro *kernel* probado.

4.2.3. Comparativa entre los métodos de clasificación anteriores

Se mostrará una comparación de las siguientes mediciones realizadas: precisión de clasificación, tiempo de aprendizaje y el promedio del error absoluto.

Las primeras comparaciones que se presentan corresponden al conjunto de datos Diabetes y después se presentan las correspondientes al conjunto de datos Hepatitis. Como primera comparativa, se presenta la Figura 4.38. Como se puede ver, el algoritmo que presenta los valores más bajos de precisión de clasificación es *ADTree*. Arriba del desempeño de ese algoritmo se encuentra el de C4.5, que corresponde al desempeño intermedio de precisión de clasificación. El algoritmo que obtiene los valores más altos

de precisión de clasificación es SVM utilizando función polinomial. Al utilizar SVM con un kernel de base radial, se puede observar que inicia con valores bajos de precisión de clasificación, pero después adopta un comportamiento similar al obtenido con el otro kernel.

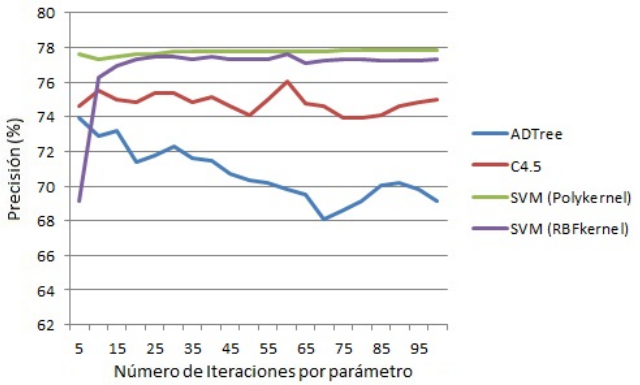

Figura 4.38: Precisión de los algoritmos *ADTree*, C4.5 y SVM del conjunto de datos Diabetes

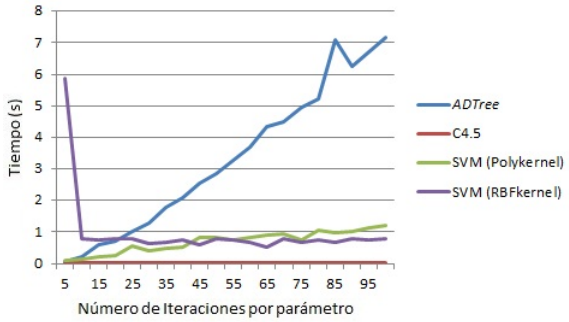

Figura 4.39: Tiempo en segundos de clasificación de *ADTree*, C4.5 y SVM del conjunto de datos Diabetes

En la Figura 4.39 se muestra el tiempo de entrenamiento de cada algoritmo. De todos los presentados, el algoritmo que consume un menor tiempo de entrenamiento es el C4.5 que es representado por la línea color marrón en la parte inferior de la gráfica. Arriba de este tiempo se encuentra el que le toma a la SVM, considerando cualquiera de los dos tipos de *kernel* utilizados. El algoritmo más lento para su entrenamiento es el *ADTree,* representado por la línea azul en la parte superiorde la misma Figura 4.39.

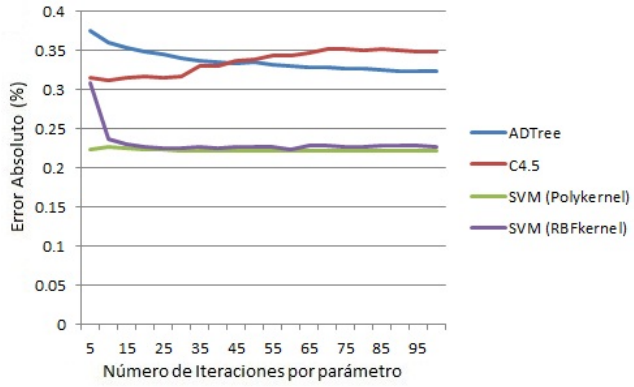

Figura 4.40: Promedio del error absoluto de *ADTree*, C4.5 y SVM del conjunto de datos Diabetes

El promedio del error absoluto se observa en la Figura 4.40. El algoritmo que presenta menores valores de error es la SVM con el tipo de función Polykernel, seguido por el mismo clasificador pero utilizando la función de base radial como kernel y representado por la línea morada en la misma figura. Los algoritmos C4.5 y *ADTree* presentan valores altos del error absoluto, como se puede ver en la parte superior de la Figura 4.40.

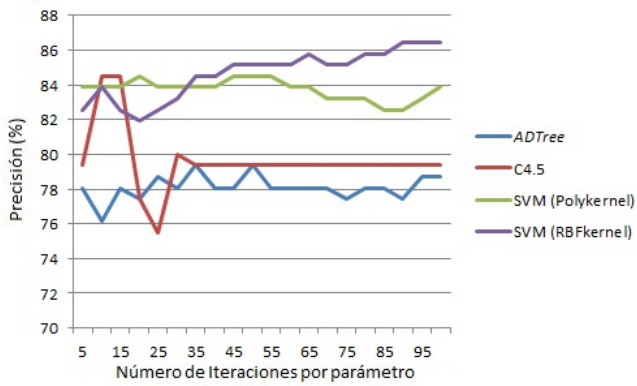

Figura 4.41: Precisión de los algoritmos *ADTree*, C4.5 y SVM del conjunto de datos Hepatitis

La Figura 4.41 muestra los resultados de precisión de clasificación de cada algoritmo para el conjunto de datos Hepatitis. El algoritmo que muestra valores altos es de nueva cuenta la SVM, pero ahora usando la función de base radial. Con una función kernel polinomial los el desempeño de clasificación el ligeramente disminuido.

El algoritmo *ADTree* es el que presenta los niveles más bajos de precisión de clasificación, representado por la línea azul en la parte inferior. Nuevamente, el algoritmo

C4.5 se encuentra en la parte intermedia del desempeño.

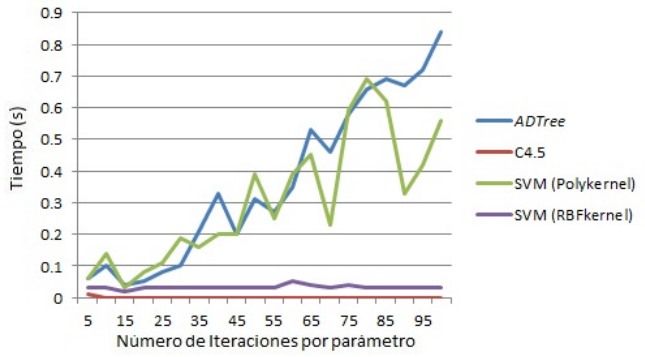

Figura 4.42: Tiempo en segundos de clasificación de *ADTree*, C4.5 y SVM del conjunto de datos Hepatitis

El tiempo de aprendizaje se muestra en la Figura 4.42. El algoritmo que se entrena más rápidamente es C4.5 mostrado en la línea roja en la parte inferior. El que le sigue es SVM con la función de base radial, representado por la línea morada, seguido de SVM con función polinomial en color verde. El algoritmo más lento para su entrenamiento es *ADTree*, como se puede ver en la línea de *color* azul de la Figura 4.42.

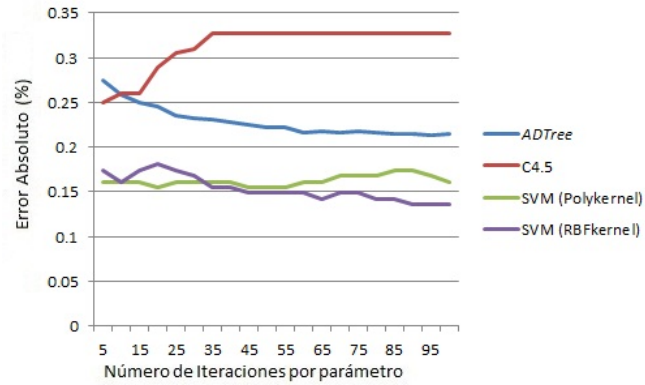

Figura 4.43: Promedio del error absoluto de *ADTree*, C4.5 y SVM del conjunto de datos Hepatitis .

En la Figura 4.43 se exhiben los errores absolutos de cada algoritmo. La SVM es la que presenta valores menores de error cuando se usa el kernel de base radial, seguido por el mismco clasificador pero usando función polinomial. El algoritmo *ADTree* se encuentra en la parte intermedia y el C4.5 muestra valores altos del error.

Se puede afirmar que, considerando los experimentos realizados, las SVM producen un buen desempeño para los conjuntos de datos analizados.

4.3. Implementación de una interfaz gráfica de usuario con Weka

Para poder utilizar los algoritmos de clasificación que se mencionan en esta tesis, se implementó una interfaz gráfica de usuario (GUI) desarrollada en lenguaje Java. Esta interfaz utiliza los clasificadores incluidos en Weka que fueron empleados en las secciones anteriores. Se utilizaron los elementos mostrados en la Tabla 4.3.

Tabla 4.3: Elementos utilizados para el desarrollo de la interfaz gráfica de usuario

Características	**Propiedades**
NetBeans	IDE 7.0.1
Weka versión desarrollador	Archivos weka.jar y libsvm.jar

Los algoritmos de Weka pueden ser utilizados en proyectos mediante la inclusión de los archivos comprimidos (archivos jar) que se proporcionan con Weka. En este trabajo usamos Netbeans para la realizacion de la interfaz gráfica de usuario. Es necesario añadir el paquete Weka al proyecto de Netbeans para la GUI desarrollada. La Figura 4.44 muestra la estructura de las bibliotecas utilizadas.

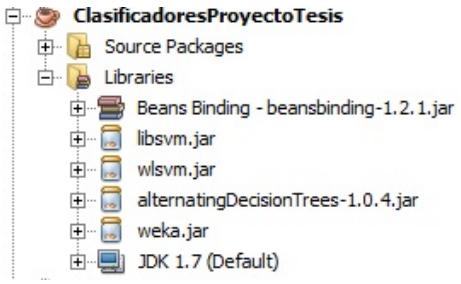

Figura 4.44: Añadir paquetes de Weka al proyecto

Los paquetes se agregan en forma de bibliotecas, y se encuentran almacenados en la carpeta donde se instaló Weka. Cabe mencionar que es necesario descargar la versión de desarrollador de Weka para poder hacer uso de las mismas. Las bibliotecas de Weka contienen las clases que implementan los algoritmos. Para llamar y hacer uso de cada algoritmo, se crea un objeto del tipo adecuado, y luego se modifican sus propiedades o parámetros. Por ejemplo, para usar al algoritmo *ADtree*, C4.5 y SMO, el código fuente sería como se muestra a continuacion:

```
weka.classifiers.trees.ADTree adtree = new weka.classifiers.trees.ADTree();
    weka.classifiers.trees.J48 j48 = new weka.classifiers.trees.J48();
weka.classifiers.functions.SMO smo = new weka.classifiers.functions.SMO();
```

Para modificar los parámetros de cada algoritmo, se usan los métodos de acceso correspondientes. Por ejemplo, el algoritmo C4.5, que en Weka se implementa como la clase J48, los parámetros se establecen de la siguiente forma:

```
j48.setBinarySplits(false);
j48.setConfidenceFactor(1.0f);
j48.setDebug(false);
j48.setMinNumObj(2);
j48.setNumFolds(3);
j48.setReducedErrorPruning(false);
j48.setSaveInstanceData(false);
j48.setSeed(1);
j48.setSubtreeRaising(true);
j48.setUnpruned(false);
j48.setUseLaplace(false);
```

Una vez que se han creado los objetos necesarios para usar los algoritmos, hay que introducir los conjuntos de datos que se desean evaluar con los diferentes clasificadores. El formato nativo para Weka es .arff, sin embargo, la herramienta cuenta con varios filtros para poder convertir datos almacenados en archivos con otros formatos al formato nativo. En este trabajo sólo se usaron archivos en formato nativo de Weka. Para agregar los conjuntos de datos hay que especificar la ruta en donde se encuentra almacenado el archivo que los contiene. Esto se puede lograr utilizando el fragmento de código mostrado en la Figura 4.45. En la implementacion de la GUI, la ventana para cargar archivos es similar a la mostrada en la Figura 4.49.

```
weka.core.converters.ArffLoader loader = new weka.core.converters.ArffLoader();
String ruta = "C:\\Users\\prof-1234\\Documents\\ThesisRodolfo2013\\Enfermedades\\diabetes.arff";
loader.setFile(new File(ruta));
loader.setUseRelativePath(true);
weka.core.Instances data = loader.getDataSet();
data.setClassIndex(data.numAttributes() - 1);
```

Figura 4.45: Introducir la ruta del archivo de conjuntos de datos

La explicación del código de la Figura 4.49 es la siguiente. Primero, se crea un objeto de tipo ArffLoader (objeto `loader` en el código mostrado), que sirve para leer los datos del archivo. El objeto `loader` requiere conocer la ruta y el nombre del archivo. Esto se realiza mediante la creación de un objeto anónimo de tipo `File` y la invocación del método `setFile` del objeto `loader`. Una vez que se ha realizado lo anterior, se puede generar un conjunto de datos invocando al método `getDataSet` del objeto `loader`. El

resultado se guarda en la referencia `data`, que es de tipo `Instances`. El atributo de clase es el último en los conjuntos de datos utilizados, por lo tanto, esto se especifica invocando al método `setClassIndex` del objeto `data`.

```
int sizeDataSet = data.size();
java.util.Random rand = new java.util.Random();
rand.setSeed(System.currentTimeMillis());
data.randomize(rand);
weka.core.Instances testing = new weka.core.Instances(data,  0,(int) (sizeDataSet * 0.3));
for (int i = 0; i < sizeDataSet * 0.3; i++) {
    data.remove(i);
}
```

Figura 4.46: Evaluación del 30 % de elementos del conjunto de datos

Una vez cargado el conjunto de datos, se elige para el entrenamiento, un 30 % de las instancias del conjunto de datos de manera aleatoria, y el 70 % restante se utiliza para prueba. La forma de realizar esto se muestra en la Figura 4.46.

Para realizar el entrenamiento de los clasificadores, se usa el método `buildClassifier`, que recibe como argumento un conjunto de datos. En nuestro caso, se le asigna el 30 % elegido aleatoriamente. A continuación se muestra en la Figura 4.47 como se realiza el entrenamiento usando como ejemplo al algoritmo C4.5. En la misma figura se presenta la forma de establecer algunos de los parámetros mencionados en las secciones anteriores.

```
j48.setBinarySplits(true);
j48.setMinNumObj(80);
j48.buildClassifier(data);
weka.classifiers.Evaluation eval = new weka.classifiers.Evaluation(testing);
eval.evaluateModel(j48, testing);
System.out.println("" + eval.toSummaryString());
```

Figura 4.47: Parámetros del algoritmo C4.5 en java

Como parte de las funcionalidades de la GUI, se le permite al usuario modificar el porcentaje de datos usados para el entrenamiento, y el resto es usado para probar el clasificador. Esto último se hace por medio de una barra deslizable. Se agregó también un área donde se muestran los resultados. Una vista de la interfaz se muestra en la Figura 4.48.

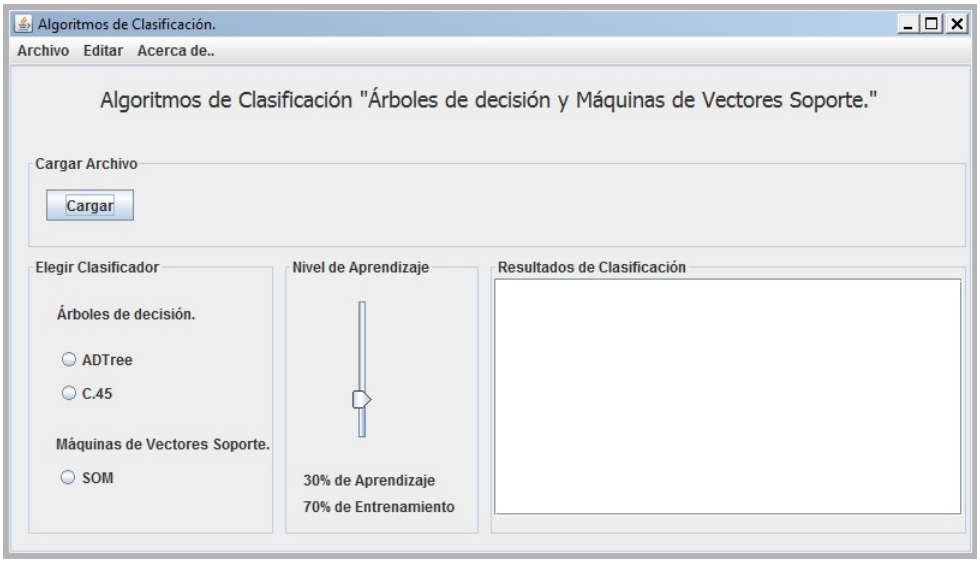

Figura 4.48: Ventana principal de la interfaz gráfica de usuario

Al hacer clic en el boton cargar, aparece una ventana para poder posicionarse en donde se encuentra el archivo del conjunto de datos (Figura 4.49).

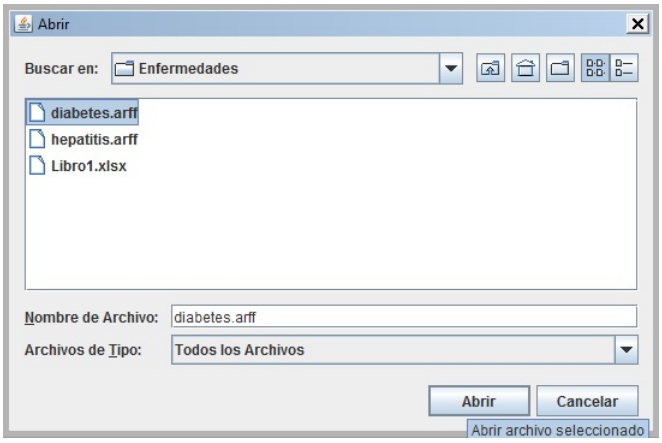

Figura 4.49: Cargar Archivo

Al elegir el conjunto de datos, la ruta de posicionamiento del mismo aparece en la ventana principal del sistema como se observa en la Figura 4.55.

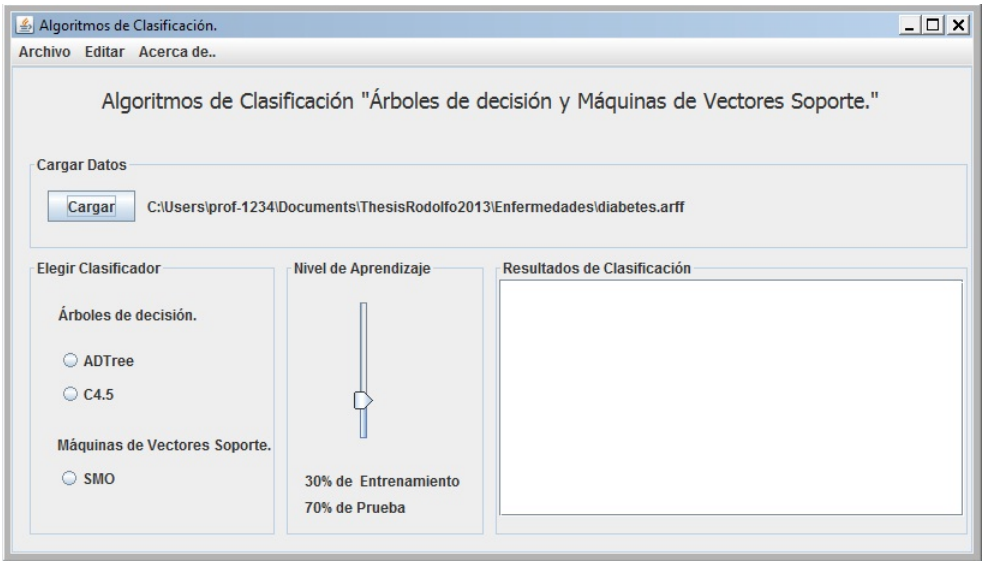

Figura 4.50: Introducir ruta del archivo a la interfaz

Al ingresar la ruta del archivo se podrán utilizar los algoritmos que se muestran en la Figura 4.55, si no se ha ingresado la ruta del conjunto de datos, aparecerá una ventana de error indicando que hace falta ingresar la ruta del archivo. Esto se muestra en la Figura 4.51.

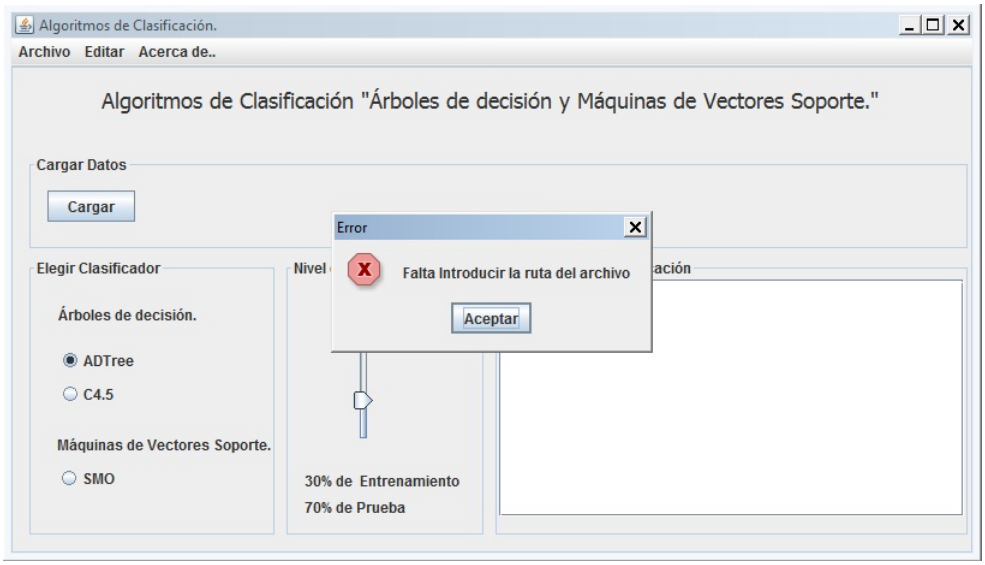

Figura 4.51: Mensaje de error

Si se elige el algoritmo *ADTree,* se abre una ventana que contiene los parámetros correspondientes a este algoritmo. El usuario puede cambiar o modificar como requiera. La Figura 4.52 presenta la GUI para este propósito.

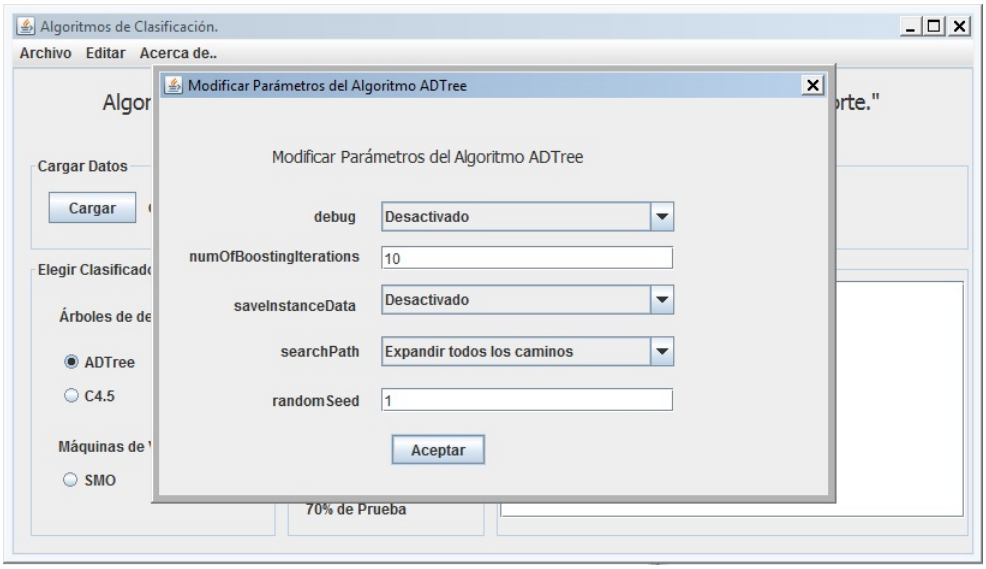

Figura 4.52: Ventana de parámetros del algoritmo *ADTree*

Después de modificar los parámetros del algoritmo y dar clic en aceptar, aparecen los resultados en el área de texto mostrada en la Figura 4.53

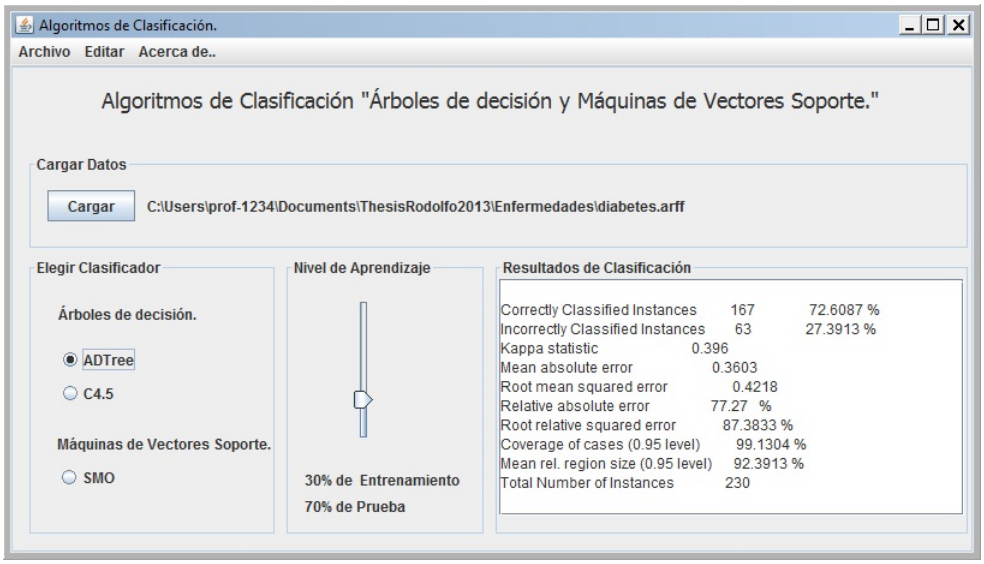

Figura 4.53: Resultados estadísticos del algoritmo *ADTree*

Si se desea modificar el porcentaje de datos usados para entrenamiento, se ajusta la barra de desplazamiento, como se muestra en la Figura 4.54

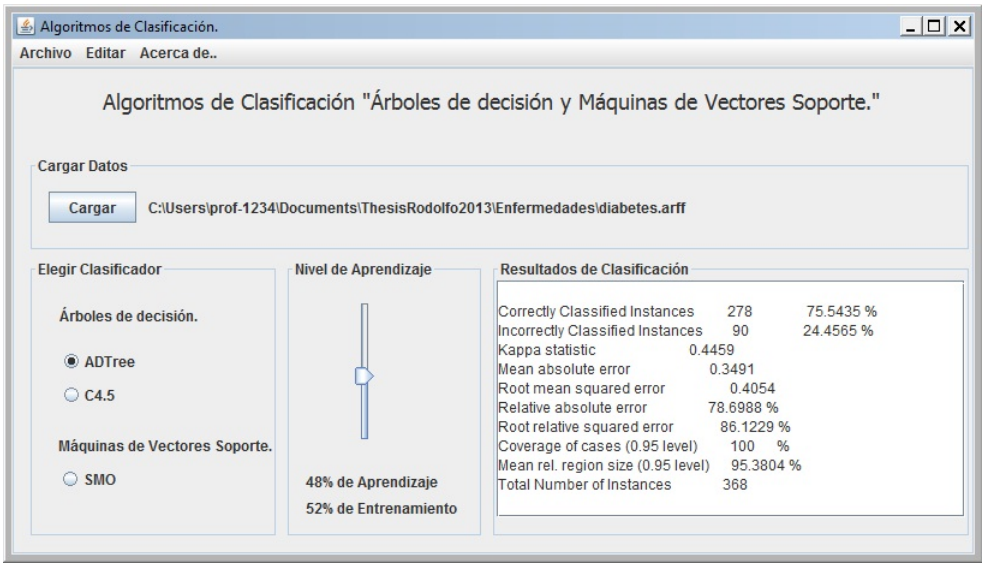

Figura 4.54: Variando valores de entrenamiento del algoritmo *ADTree*

De la misma forma si se selecciona el algoritmo C4.5 aparecerá la ventana donde contienen todos los parámetros que se pueden modificar por el usuario como se ve en la Figura 4.55

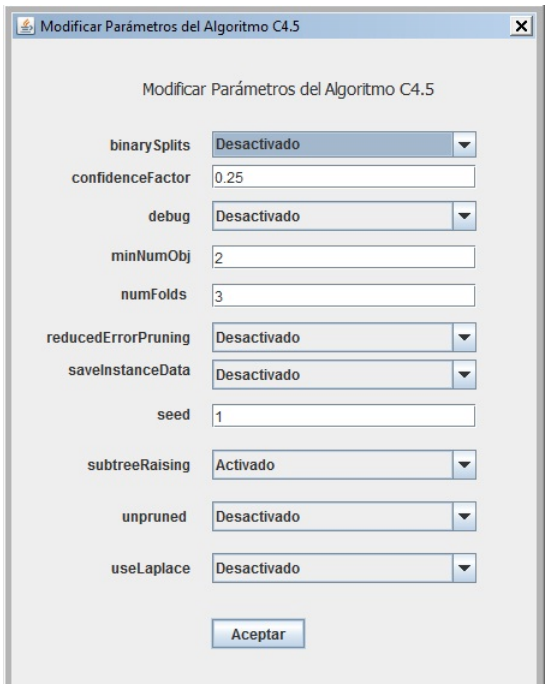

Figura 4.55: Ventana de parámetros del algoritmo C4.5

Al dar clic en aceptar después de modificar los parámetros deseados, aparecerán los resultados estadísticos en el área de resultados de clasificación como se ve en las Figuras

4.56 y 4.57.

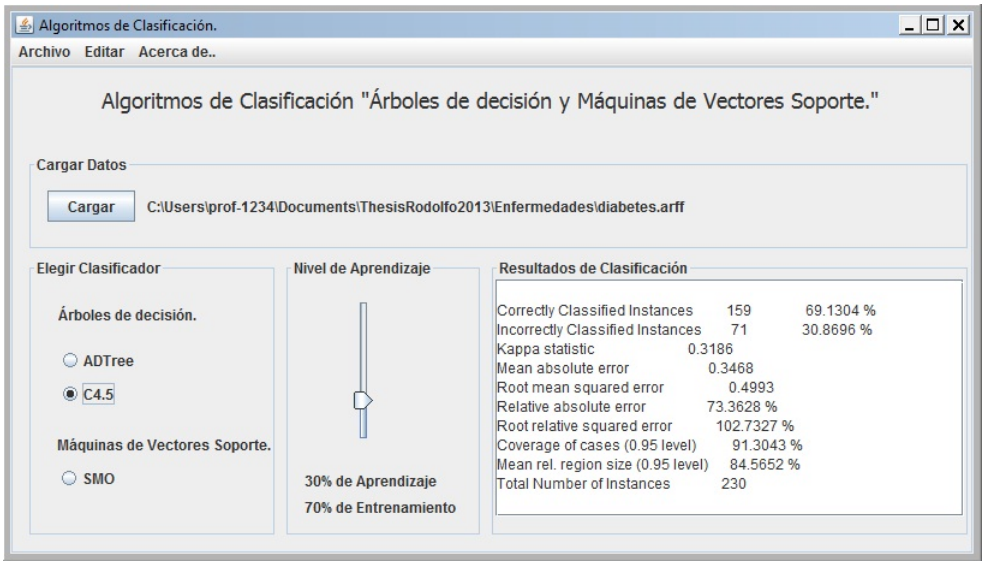

Figura 4.56: Resultados de clasificación del algoritmo C4.5

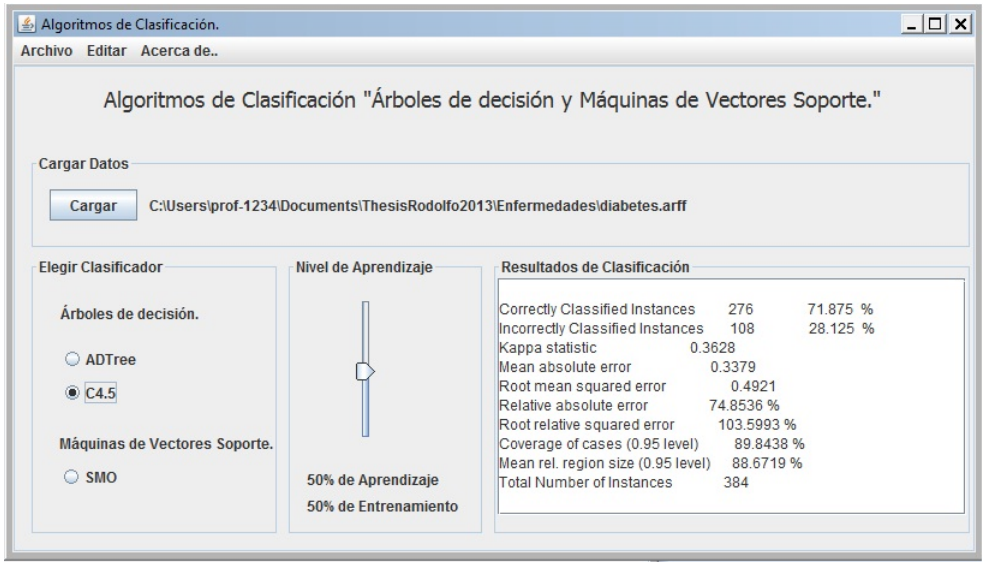

Figura 4.57: Ajustando el nivel de aprendizaje del algoritmo C4.5

Por último, la Figura 4.58 presenta la GUI desarrollada para establecer los parámetros de SMO.

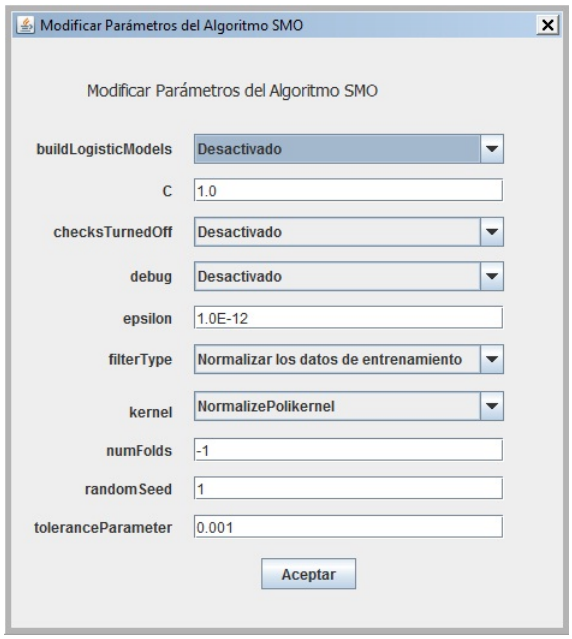

Figura 4.58: Ventana de modificación de parámetros del algoritmo SMO

La Figura 4.59 muestra la ventana de resultados para el conjunto diabetes usando el clasificador SVM entrenada con SMO.

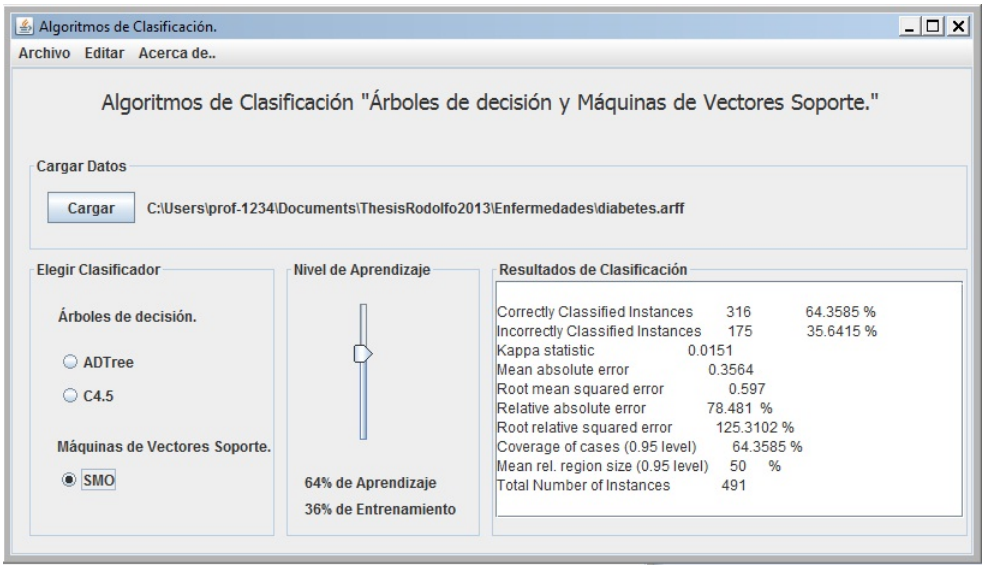

Figura 4.59: Resultados de clasificación del algoritmo SMO

Conclusiones

En este trabajo se realizó una comparación entre dos de los algoritmos más utilizados en minería de datos, los árboles de decisión y las máquinas de soporte vectorial. Estos dos tipos de algoritmos fueron aplicados a conjuntos de datos del área de medicina. Los conjuntos de datos fueron utilizados del repositorio UCI, y corresponden a dos de las enfermedades que ocupan los primeros lugares de mortalidad en México, estas son la Diabetes y la Hepatitis.

Los algoritmos que se compararon son *ADTree*, C4.5 y SVM entrenada utilizando Optimización Mínima Secuencial. En los experimentos, se modificaron los parámetros de cada clasificador y se realizaron las siguientes mediciones: precisión de clasificación, el tiempo de aprendizaje de clasificación y el promedio del error absoluto, número de nodos y hojas al crear los árboles y el tipo de *kernel* utilizado en SVM.

En los experimentos con el algoritmo *ADTree,* se modificó el valor del parámetro *numOfBoostingIterations.* Se pudo observar mediante los resultados obtenidos, que el tiempo de aprendizaje se hace más lento cuando se incrementa el valor de este parámetro. Esto se puede explicar con el hecho de que de acuerdo al valor de *numOfBoostingIterations,* el árbol generado incrementa el número de sus nodos. En general, el algoritmo *ADTree* presenta resultados bajos de precisión de clasificación. También se pudo observar que este algoritmo muestra grandes errores de clasificación para los conjuntos de datos Diabetes y Hepatitis.

Para el algoritmo C4.5, el parámetro modificado fue *minNumObj.* Al manipular su valor en cada experimento, se pudo observar que afecta los resultados de precisión de clasificación. En el caso del conjunto de datos Diabetes, dicha precisión se encuentra dentro de un rango de 74 % a 75.5 %, . En el caso del conjunto de datos Hepatitis, los resultados de precisión de clasificación muestran que después de un valor de *minNumObj* igual a 30, el algoritmo C4.5 tiende arrojar resultados similares. Es

probable que esto se deba a que por cada incremento del valor del parámetro, se eliminan datos que no son necesarios para realizar la clasificación, por lo tanto, al no haber más datos inecensarios, arrojará un mismo resultado. El tiempo de aprendizaje para este algoritmo es muy rápido. Dentro de los conjuntos de datos Diabetes y Hepatitis presentan resultados altos del promedio del error absoluto en la clasificación.

En en caso de las máquinas de soporte vectorial usando Optimización Mínima Secuencial para su entrenamiento, se modificó el parámetro de regularización *c,* y se probaron dos funciones de *Kernel* llamados Polykernel y RBFkernel en Weka. Para el primer kernel, se observó que al incrementar el valor del parámetro *c* en cada experimento, los mejores resultados obtenidos fueron con el conjunto de datos Diabetes. El tiempo de aprendizaje es muy rápido y presenta menos errores al realizar la clasificación. Al utilizar el kernel de función de base radial, RBFkernel, y ser modificado el parámetro *c*, se pudo observar que los valores precisión de clasificación más altos fueron para el conjunto de datos Hepatitis. El tiempo de aprendizaje fue rápido, además presenta valores bajos de error de clasificación.

En resumen, se puede decir que los árboles de decisión no son muy adecuados para la extracción del conocimiento de los conjuntos de datos Diabetes y Hepatitis. Sin embargo, el algoritmo que es adecuado para la extracción del conocimiento de los conjuntos de datos de las enfermedades Diabetes y Hepatitis es SVM. A su vez, se puede concluir que al utilizar como kernel el tipo de función de base radial, en conjuntos de datos desbalanceados, puede ser favorable. como se observó en el conjunto de datos Hepatitis.

Para finalizar, se puede decir que las técnicas de clasificación actuales no sustituyen al especialista humano, sin embargo, si constituyen una herramienta para facilitar el diagnóstico de enfermedades.

Trabajos futuros

Como trabajos a futuro se presentan los siguientes mejoras que se pueden implementar a esta tesis:

- Comparativa con más algoritmos de clasificación: Evaluar los conjuntos de datos de Diabetes y Hepatitis para extraer el conocimiento con más algoritmos de clasificación y conocer el comportamiento de cada algoritmo.
- Obtener datos reales de personas: Crear conjuntos de datos de personas que tengan enfermedades de Diabetes y Hepatitis en México con los principales síntomas y análisis de laboratorio.
- Hacer multidisciplinario este trabajo, involucrando con expertos del área médica: Tener opiniones de diferentes médicos para comprender más a detalle las enfermedades Diabetes y Hepatitis y de esta forma poder generar algoritmos con una mayor precisión en la clasificación.

Bibliografía

[1] M. Ing. Aguirre Botello, "MÉXICO, PRINCIPALES CAUSAS DE MORTALIDAD DESDE 1938." http://www.mexicomaxico.org/Voto/MortalidadCausas.htm, Mayo 2011. Consultada el 29/04/2013.

[2] B. Vanguardia, "Superan muertes por hepatitis c a las de virus vih." http://www.vanguardia.com.mx/superanmuertesporhepatitiscalasdevirusvih -1279137.html, Mayo 2012. Consultada el 29/04/2013.

[3] T. Chen and I. L. Estrada Delas, *Caligrafía China*. Cultural China Series, 2003.

[4] P. Engineering, "ENIAC: Celebrating Penn Engineering History." http://www.seas.upenn.edu/about-seas/eniac/, Febrero 2013. Consultada el 15/02/2013.

[5] C. A. Coello Coello, "Colossus: El Secreto Mejor Guardado por los ingleses durante la Segunda Guerra Mundial." http://www.lania.mx/ ccoello/historia/papers/colossus.html, Abril 2000. Consultada el 15/02/2013.

[6] M. Nguyen, "Electronic computers within the ordnance corps." http://ftp.arl.army.mil/mike/comphist/61ordnance/index.html. Consultada el 15/02/2013.

[7] P. W. Brian Napper, "Early Electronic Computers (1946-51)." http://www.computer50.org/mark1/contemporary.html, Agosto 1999. Consultada el 05/02/2013.

[8] R. G. Langenbach, *Introducción al proceso de datos*. PRENTICE-HALL, INC.,Englewood Cliff, N.J., 1976.

[9] U. de Málaga. UCIENCIA, "Cintas magnéticas." http://www.uciencia.uma.es/Coleccion-cientifico-tecnica /Informatica/ Galeria/Cintas-magneticas, Febrero 2013. Consultada el 07/02/2013.

[10] C. A. Coello Coello, "El origen del miedo a las computadoras," *Cinvestav*, pp. 68–71, Abril-Junio 2007.

[11] Saberia.com, "¿Quién inventó el vinilo?." http://www.saberia.com/ 2010/03/quien-invento-el-vinilo/, 2009-2013. Consultada el 07/02/2013.

[12] H. F. Rentería Toledo, "Disquete." http://www.tecnotopia.com.mx/mecatronica/disquete.htm, 2003-2010. Consultada el 07/02/2013.

[13] P. W. Nobel Prize, "The history of the integrated circuit." http://www.nobelprize.org/educational/ physics/integrated_circuit/history/index.html, Febrero 2013. Consultada el 02/02/2013.

[14] O. Charles, "Medios de Almacenamiento." http://www.xtimeline.com/evt/view.aspx?id=987905, 2008-2013. Consultada el 07/02/2013.

[15] E. Duro, J. Quesada, D. Rosso, G. Vera, and J. Sequeira, *Cuadernillo de medio ambiente para comunidades indigenas.* UNICEF, 2007. Grupo de apoyo a emprendimientos productivos en comunidades marginadas.

[16] R. Camps Paré, *Introducción a las bases de datos.* www.uoc.edu: UOC La universidad virtual, 2002.

[17] O. Pons, N. Marín, J. M. Medina, S. Acid, and M. A. Vila Mirandam, *Introducción a las Bases de Datos El Modelo Relacional.* Magallanes, 25; 28015 Madrid España: Paraninfo, S.A., 2005.

[18] C. J. DATE, *Introducción a los Sistemas de Bases de Datos.* Pearson Educacion, 7ma ed., 2001.

[19] F. Gil, J. Albrigo, and J. Do Rosario, *SISTEMAS DE GESTIÓN DE BASE DE DATOS SGBD / DBMS.* No. 1 in 1, Universidad de Carabobo, Facultad Experimental de Ciencias y Tecnología, Febrero 2005.

[20] W3Schools.com, "Introduction to sql." http://www.w3schools.com/sql/sql_intro.asp, 1999-2013. Consultada el 15/02/2013.

[21] degreedirectory.org, "What is SQL?." http://www.degreedirectory.org/articles/What_is_SQL.html, 2003-2013. Consultada el 15/02/2013.

[22] A. Cobo Yera, *Diseño y programación de bases de datos*. Avenida de Austrias S/N Imapares local 5, semiesquina a Plaza Castilla 328029 Madrid, España: Visión Libros, 2007.

[23] D. Costal Costa, *El modelo relacional y el álgebra relacional*. UOC La universidad virtual, 2012.

[24] A. Weitzenfeld, *Ingeniería de software orientada a objetos con UML, Java e Internet*. Cengage Learning Editores, 2005.

[25] P. Rob and C. Coronel, *Sistemas de bases de datos: diseño, implementación y administración*. Ciencias e Ingenierías, 5 ed., Agosto 2004.

[26] B. A. Forouzan, *Introducción a la ciencia de la computación de la manipulacion de datos a la teoria de la computación*. International Thomson Editores, ilustrada ed., 2003.

[27] D. C. Hay, "Different Kinds of Data Models: History and a Suggestion," *THE DATA ADMINISTRATION NEWSLETTER*, Octubre 2010.

[28] P. Reyes, "Introducción al sql de interbase: Ddl y dml," *Artículos técnicos Grupo Danysoft:*, p. 11, Abril 2002.

[29] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. 500 Sansome Street, Suite 400, San Francisco, CA 94111: Morgan Kaufmann, 2 ed., 2006.

[30] A. Roche, "Árboles de decisión y series de tiempo," Master's thesis, Facultad de Ingeniería, UDELAR, Diciembre 2009.

[31] J. R. Quilan, *Programs for Machine Learning*. San Francisco California: Morgan Kaufmann, 1993.

[32] V. et al., *Visión por Computador*. URJC, 2003. (cap. 5).

[33] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. 500 Sansome Street, Suite 400, San Francisco, CA 94111: Morgan Kaufmann, 2 ed., 2005.

[34] D. García Morate, *Manual de Weka*. Creative Commons Reconocimiento-NoComercial-SinObraDerivada 2.0, 2003.

[35] R. Aler, *Tutorial Weka 3.6.0.* http://www.slideshare.net/RenRojasCastillo/tutorial-weka, 2009. Consultada el 1/03/2013.

[36] L. Córdoba Fallas, "Weka." http://mineriadedatoscr.blogspot.mx/2011/06/weka.html, Junio 2011. Consultada el 1/03/2013.

[37] D. Figuerola, *Diabetes.* MASSON, S.A., 4a ed., Septiembre 2003.

[38] F. F. Rocca and J. C. Plá, "Diabetes mellitus." http://www.smu.org.uy/publicaciones/libros/historicos/dm/, Febrero 2013. Consultada el 21/02/2013.

[39] M. de Trabajo y Promoción del Empleo, "Diabetes mellitus: Como enfermedad generadora de discapaciad." http://www.mintra.gob.pe/, 2007. Consultada el 15/02/2013.

[40] P. C. A. Network, "El páncreas." http://www.pancan.org/section_en_espanol/learn_about_pan_cancer/what_is_the_pancreas.php, 2013. Consultada el 21/02/2013.

[41] Especialistas Medicinas Alternativas, *Diabetes.* Especialistas Medicinas Alternativas, 2a ed., 2007.

[42] D. O. H. USA and H. SERVICES, "Nkudic." http://kidney.niddk.nih.gov/index.aspx.

[43] R. J. Marcano Pasquier, "Las pruebas de tolerancia a la glucosa." http://www.medicinapreventiva.com.ve/laboratorio/tolerancia_glucosada.htm, Enero 2013. Consultada el 7/03/2013.

[44] Y. S. of Medicine, "Diabetes Y Presión Sanguínea Alta." http://www.yalemedicalgroup.org/stw/Page.asp?PageID=STW024474, 2013. Consultada 7/03/2013.

[45] T. H. Institute, "Calculadora del índice de masa corporal." http://www.texasheartinstitute.org/HIC/Topics_Esp/HSmart/bmi_calculator_span.cfm, 2013. Consultada el 12/03/2013.

[46] R. López de Mántaras and L. Saitta, *EOAI European Conference on Artificial Intelligence*, vol. 110. Valencia, España: IOS Press, 16th ed., 2004.

[47] M. Patlak, B. Blumberg, M. Hilleman, and W. Rutter, "La historia de la hepatitis B." http://www7.nationalacademies.org/spanishbeyonddiscovery/LaFebrero 2000. Consultada el 18/02/2013.

[48] P. W. Nobelprize, "Autobiography." http://www.nobelprize.org/ nobel_prizes/ medicine/laureates/ 1976/blumberg-autobio.html, Febrero 2013. Consultada el 18/02/2013.

[49] D. F. Calmet Bruhn, "Hepatitis viral," *DIAGNOSTICO*, vol. 46, Enero-Marzo 2007.

[50] SHARP, "Anatomía y la función del hígado." http://www.sharpenespanol.com/healthinfo/content.cfm?pageid=P06162, 2013. Consultada el 18/02/2013.

[51] SHARP, "Hígado: Anatomía y funciones." http://www.sharpenespanol.com /healthinfo/content.cfm?pageid=P03769, 2013. Consultada el 18/02/2013.

[52] R. Romero Cabello, *Microbiología y parasitología humana*. Editorial médica panamericana S.A. de C.V., 3a ed., 2007.

[53] N. Cordeiro, R. Taroco, and H. Chiparelli, *TEMAS DE BACTERIOLOGÍA Y VIROLOGÍA MÉDICA*. UIAD Instituto de higiene, 2006.

[54] M. E. Borbolla Sala, I. E. Juárez Rojop, N. González Álvarez, L. García Vanegas, O. E. Piña Gutiérrez, and A. Rodríguez León, "Hepatitis A en Tabasco," *Salud en Tabasco*, vol. 14, pp. 732–737, Enero-Agosto 2008.

[55] MedlinePlus, "La Enciclopedia Ilustrada de Salud." http://www.nlm.nih.gov/medlineplus/spanish/encyclopedia.html, Enero 2013. Consultada 7/03/2013.

[56] About.com,"Spider Angiomas." http://hepatitis.about.com/od/stu/g/spiders.htm, Febrero 2010. Consultada el 12/03/2013.

[57] D. Alejandro Soza, "Hepatitis enfermedades del higado." Hepatitis.cl. Consultada el 18/10/2013.

[58] L. M. Molinero, "Medidas de concordancia para variables cualitativas." http://www.seh-lelha.org/concor2.htm, Diciembre 2001. Consultada el 25/07/2013.

[59] D. Candel Contardo, *Algoritmo Tipo SMO para la AD-SVM Aplicado a Clasiicación Multicategoría*. PhD thesis, Universidad Técnica Federico Santa María Departamento de Informática, Enero 2011.

[60] I. Rodríguez Luján, *Selección de variables mediante programación cuadrática*. PhD thesis, Universidad Autónoma de Madrid Escuela Politécnica Superior Departamento de Ingenier?a Informática, Febrero 2009.