UNIVERSIDAD CENTRAL “MARTA ABREU” DE LAS VILLAS

FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN

Predicción de la evolución hacia la hipertensión arterial en la adultez desde la adolescencia utilizando técnicas de aprendizaje automatizado

Trabajo de Diploma en Ciencia de la Computación

Autor

Wilfredo Alfonso González

Tutores

Dra. Leticia Arco García

Dr. Guillermo Alberto Pérez Fernández

Santa Clara, 2013

A mis padres

A mis abuelas

A mi hermana

A mis amigos

A mis tutores Dra.Leticia Arco García y Dr. Guillermo Alberto Pérez Fernández por su asesoría y orientación en este trabajo.

A las profesoras MSc. Enislay Ramentol y Dra. Yaima Filiberto de la Universidad de Camagüey por su colaboración en el desarrollo de este trabajo.

A todos los profesores que me formaron y enseñaron.

A los amigos que estuvieron conmigo en los momentos más difíciles.

A mi familia por su apoyo incondicional.

Resumen

Las recomendaciones de la Sociedad Europea de Hipertensión Arterial reconoce que la hipertensión arterial en la edad pediátrica es un problema médico que ha ido incrementándose con repercusiones negativas presentes y futuras; y hace referencia a la ausencia de estrategias o políticas de salud organizadas que enfrenten eficazmente la enfermedad en este ámbito. Existen estudios sobre el pronóstico de la conversión de un adolescente prehipertenso en un adulto hipertenso en Cuba. Sin embargo, los índices de exactitud obtenidos son aun bajos, ya que solo se alcanza el 80% de casos bien clasificados; de ahí que es necesario seguir trabajando la predicción, con mejor exactitud, de adultos hipertensos a partir de adolescentes prehipertensos. El objetivo de este trabajo consiste en aplicar técnicas de aprendizaje automatizado que permitan pronosticar la conversión de un adolescente prehipertenso en un adulto hipertenso en Cuba, elevando los índices de exactitud. Los principales resultados son: (1) Se seleccionaron las técnicas de selección de rasgos, edición de conjuntos de entrenamiento, y de aprendizaje automatizado, a aplicar que puedan contribuir a obtener mejor exactitud en la clasificación de adultos hipertensos a partir de adolescentes prehipertensos, (2) se aplicaron los algoritmos para el preprocesamiento de los datos y de aprendizaje automatizado para pronosticar la hipertensión en la adultez a partir de adolescentes prehipertensos, y (3) se realizó una comparación estadística de los clasificadores aplicados al conjunto de pacientes prehipertensos estudiados, obteniéndose un 88% de clasificación correcta al aplicar técnicas de edición para problemas con clases desbalanceadas al conjunto de datos.

Abstract

The recommendations of the European Society of Hypertension recognizes that hypertension in children is a medical problem that has been increasing with negative impacts present and future, and refers to the absence of health strategies or policies that address effectively organized disease in this area. Studies on the prognosis of the conversion of a teenager in an adult hypertensive prehypertension in Cuba. However, the obtained accuracy rates are even lower, as it only reaches 80% of cases correctly classified, hence the need to continue working the prediction, with better accuracy, hypertensive adults from prehypertensive adolescents. The objetive of this work is to apply machine learning techniques that will predict the conversion of a teenager in an adult hypertensive prehypertension in Cuba, raising accuracy rates. The main results are: (1) were selected feature selection techniques, editing training sets, and machine learning, to apply that can contribute to obtain better accuracy in the classification of hypertensive adults from adolescents prehypertension (2) algorithms were applied to the data preprocessing and machine learning to predict hypertension in adulthood from prehypertensive adolescents, and (3) there was a statistical comparison of classifiers applied to all prehypertensive patients studied, obtaining 88% correct classification by applying editing techniques for problems with unbalanced classes to the data set.

Tabla de contenidos

Introducción

Las recomendaciones de la Sociedad Europea de Hipertensión Arterial (HTA) bajo la supervisión de la Sociedad Europea de Cardiología para el tratamiento de la HTA en niños y adolescentes reconocen que la HTA en la edad pediátrica es un problema médico que ha ido incrementándose con repercusiones negativas presentes y futuras (Berenson 2002). Del mismo modo hacen referencia a la ausencia de estrategias o políticas de salud organizadas que enfrenten eficazmente la enfermedad en este ámbito.

Muchas publicaciones abordan la temática de la hipertensión arterial (HTA) en la adolescencia, en función de las diferentes situaciones de riesgo que inciden en en el desarrollo de un adolescente hipertenso. La mayoría se limitan a describir la problemática y todas le dan la debida importancia a la prevención cardiovascular (Pérez Fernández 2012).

No se conoce a ciencia cierta, debido a la escasez de estudios de seguimiento desde la adolescencia hacia la adultez, cuando podría ocurrir la conversión del riesgo en enfermedad en determinado adolescente con factores de riesgo en su etapa de transición hacia la adultez (Pérez Fernández 2012).

El patrón o trayecto que sigue la Presión Arterial en el tiempo ha sido examinado específicamente por dos importantes investigaciones: el Estudio Muscatine (Berenson 2002) y el de Bogalusa (McNiece 2007), ambos transcendentales en el acápite del comportamiento y variabilidad tensional hacia la adultez desde la adolescencia. Estas investigaciones han identificado elementos a tener en cuenta, entre ellos: presencia de una elevación tensional previa no acorde con la edad, historia familiar de HTA, presencia de aumento de la masa ventricular izquierda, color de la piel, peso, talla, estatus socioeconómico, bajo peso al nacer, el sobrepeso y la obesidad.

Aunque diversos estudios de corte longitudinal han reconocido el hecho de que las cifras de PA en la niñez y adolescencia influyen sobre los valores de PA en la adultez (Lurbe 2010). Lamentablemente la existencia de este tipo de estudio longitudinal sobre el trayecto no es mayoritaria, sobre todo en Latinoamérica. Aunque es importante destacar dos resultados de estudios de este tipo en Cuba. Uno de ellos publicado en (Pérez Fernández 2012) donde se diseñaron dos modelos de predicción para pronosticar la conversión de un adolescente prehipertenso en un adulto hipertenso, mediante análisis multivariados que dan lugar a la definición de dos criterios diferentes para un índice integral de riesgo. El primero de ellos se basó en la V de Cramer y consistió en la suma de los riesgos

presentes ponderada por el valor de la V de Cramer de cada riesgo. Se resumen los valores de la V de Cramer obtenidos para cada factor de riesgo y se da a cada riesgo un orden de importancia de acuerdo al valor de la V de Cramer. Con este modelo obtuvieron una exactitud o porcentaje de buena clasificación de 61.6%. El segundo modelo diagnostica el índice de riesgo de HTA basado en regresión logística y tuvo un índice de exactitud de 70.4%. Otro estudio interesante se presenta en (Pérez Fernández 2012) donde determinan modelos predictivos mediante regresión logística y árboles de decisiones utilizando la técnica de CHAID (chi square interaction detector) y se evaluaron los clasificadores mediante áreas bajo la curva ROC (Receiver Operating Characteristic). El árbol de decisiones fue capaz de clasificar acertadamente al 80% de los casos, superando los dos modelos anteriores. Los resultados presentados en (Pérez Fernández 2012; Pérez Fernández 2012) reflejan resultados importantes sobre estudios del pronóstico de la conversión de un adolescente prehipertenso en un adulto hipertenso, no obstante, aun los índices de exactitud son bajos, ya que el mejor resultado obtenido fue del 80% de casos bien clasificados. Todo ello constituye una problemática a la cual aún no se le ha dado respuestas definitivas, lo cual justifica el **planteamiento del problema** siguiente.

Existen estudios sobre el pronóstico de la conversión de un adolescente prehipertenso en un adulto hipertenso en Cuba; ellos utilizan V de Cramer, regresión logística y árboles de decisiones. Sin embargo, los índices de exactitud obtenidos son aun bajos, ya que solo se alcanza el 80% de casos bien clasificados; de ahí que es necesario seguir trabajando la predicción, con mejor exactitud, de adultos hipertensos a partir de adolescentes prehipertensos.

El **objetivo general** del trabajo de diploma consiste en aplicar técnicas de aprendizaje automatizado que permitan pronosticar la conversión de un adolescente prehipertenso en un adulto hipertenso en Cuba, elevando los índices de exactitud previamente obtenidos.

Este se desglosa en los siguientes **objetivos específicos**:

1. Seleccionar las técnicas de selección de rasgos, edición de conjuntos de entrenamiento, y de aprendizaje automatizado en general, a aplicar que puedan contribuir a obtener mejor exactitud en la clasificación de adultos hipertensos a partir de adolescentes prehipertensos.
2. Aplicar algoritmos para el preprocesamiento de los datos y algoritmos de aprendizaje automatizado para pronosticar la hipertensión en la adultez a partir de los adolescentes prehipertensos.

3. Realizar una comparación estadística de los clasificadores aplicados al conjunto de pacientes prehipertensos estudiados.

Las **preguntas** planteadas son:

- ¿Cuáles técnicas de aprendizaje automatizado, de selección de rasgos y de edición de conjuntos de entrenamiento contribuirán a elevar la exactitud de la clasificación de adultos hipertensos a partir de adolescentes prehipertensos?
- ¿Cómo aplicar los métodos de aprendizaje con clases desbalanceadas a la clasificación de adultos hipertensos a partir de adolescentes prehipertensos?
- ¿Qué consideraciones se requieren para realizar un estudio comparativo de los clasificadores empleados a partir de la aplicación de técnicas de preprocesamiento del conjunto de datos?

Justificación:

La hipertensión arterial (HTA), constituye una problemática de salud para cualquier nivel de atención sanitaria, independientemente del tipo de régimen o condición socioeconómica imperante (Pérez Fernández 2005; Mancia 2007; McNiece 2007). Cifras alarmantes indican que podría existir un incremento de hasta el 60 % del total de hipertensos a nivel mundial, lo que significaría la elevación de las cifras de un billón en la actualidad hasta 1.65 billones para el año 2025 (Pérez Fernández 2005).

En (Pérez Fernández 2012) se hace referencia a publicaciones donde se plantea que hace más de tres décadas la mayor parte de la comunidad médica en todo el mundo consideraba a la HTA esencial como una entidad propia del adulto, y no es solo hasta 1977, de la Primera Tarea de Fuerza Americana para el control de la hipertensión arterial en niños y adolescentes, se dio inicio a una nueva era en el enfrentamiento de esta enfermedad, en edades precoces de la vida, donde su detección temprana se impone.

En el año 2009, las recomendaciones de la Sociedad Europea, bajo la supervisión de la Sociedad Europea de Cardiología para el tratamiento de la HTA en niños y adolescentes, además de reconocer que la HTA en la edad pediátrica es un problema médico que ha ido incrementándose con repercusiones negativas presentes y futuras muy poco alentadoras; hacen referencia, a la ausencia de estrategias o políticas de salud organizadas que enfrenten eficazmente la enfermedad en este ámbito.

Diversos estudios de corte longitudinal han reconocido el hecho de que las cifras de PA en la niñez y adolescencia influyen sobre los valores de PA en la adultez (Lurbe 2010). Lamentablemente la

existencia de este tipo de estudio longitudinal no es mayoritaria, sobre todo en Latinoamérica. Cuba no está exenta de esta realidad, aunque es importante destacar que ya se han realizado estudios de este tipo con buenos resultados, no obstante, es necesario continuar aplicando técnicas que permitan predecir con mayor exactitud la hipertensión en la adultez (Pérez Fernández 2012; Pérez Fernández 2012).

De ahí que se justifica el desarrollo de esta investigación que permitirá predecir la evolución hacia la hipertensión arterial en la adultez desde la adolescencia utilizando técnicas de aprendizaje automatizado que partirán de conjuntos de ejemplos descritos por rasgos ya estudiados como son: la presencia de una elevación tensional previa no acorde con la edad, obesidad, historia familiar de HTA o la presencia de un aumento de la masa ventricular izquierda detectada por ecocardiografía, color de la piel no blanca, peso, talla y estatus socioeconómico y bajo peso al nacer, entre otros.

Así, como se plantea en (Pérez Fernández 2012): "El logro de lo anterior sería una nueva manera de actuar sobre el riesgo, lo que en ocasiones es realmente difícil si tenemos en cuenta la gran variabilidad de los factores de riesgo en cualquier individuo. Con esto indudablemente, se "enmarcaría" mejor al adolescente con mayores y objetivas posibilidades de ser hipertenso en la adultez, y el nivel de precisión se vería incrementado de manera elocuente".

Factibilidad de la investigación:

Contamos con la asesoría del Dr. C. Guillermo A. Pérez Fernández quién lleva años estudiando la predicción de la evolución hacia la hipertensión arterial en la adultez desde la adolescencia. Uno de sus principales resultados consistió en el diseño de dos modelos de predicción para pronosticar la conversión de un adolescente prehipertenso en un adulto hipertenso, mediante análisis multivariados que dan lugar a la definición de dos criterios diferentes para un índice integral de riesgo, utilizando V de Cramer y regresión logística (Pérez Fernández 2012). Otro estudio interesante se presenta en (Pérez Fernández 2012) donde determinan modelos predictivos mediante regresión logística y árboles de decisiones utilizando la técnica de CHAID (chi square interaction detector).

Podemos utilizar el conjunto de pacientes que componen la muestra de 125 adolescentes seleccionados por muestreo aleatorio simple con diagnóstico de preHTA, del total de 385 prehipertensos diagnosticados. De estos 125 pacientes se conoce si son hipertensos o no en la adultez porque fueron seguidos durante 8 años. Se identificaron 82 adolescentes (65,60 %) que se convirtieron en hipertensos establecidos en la adultez; mientras que el 34,40 % permaneció en la

categoría de preHTA. Los rasgos que describen a los pacientes son: antecedente familiar de HTA, antecedente familiar de obesidad, antecedente familiar de diabetes mellitus, antecedente familiar de cardiopatía isquémica, antecedente familiar de insuficiencia cardíaca, antecedente familiar de accidente vascular encefálico, ambiente familiar, peso al nacer, rendimiento académico, peso corporal, talla, edad, sexo, color de la piel, prehipertensión arterial, hipertensión arterial esencial, índice de masa corporal, sobrepeso y obesidad.

Existe la herramienta que incluye los principales algoritmos que permiten realizar predicciones; además, incluyen filtros que permiten aplicar técnicas de selección de rasgos, identificar asociaciones entre los rasgos, visualizar los resultados y aplicar medidas de validación de las clasificaciones realizadas.

El laboratorio de Inteligencia Artificial del CEI tiene gran experiencia en la aplicación de técnicas de aprendizaje automatizado para la predicción, y en la utilización del Weka y el Keel como herramientas que incluyen los principales algoritmos a utilizar, por otro lado, el diplomante tiene habilidades en la utilización del Weka. Además, en el CEI-UCLV se han desarrollado y se desarrollan nuevos métodos de aprendizaje automático que pudieran ser aplicables a este problema con el objetivo de obtener mayor exactitud en la predicción de la hipertensión en la adultez; entre ellos se encuentran: algoritmos para la selección de rasgo (Gómez Díaz 2010) y edición de conjuntos de entrenamiento (Caballero Mota 2008), clasificadores que combinan el razonamiento basado en casos con los sistemas basados en reglas y con los sistemas neuroborrosos (Rodríguez Sarabia 2009), clasificadores basados en reglas, aquellos que resuelven problemas cuando las clases están desbalanceadas (Filiberto Cabrera 2011), redes neuronales recurrentes (Guevara Yanes 2007; Bonet Cruz 2009), nuevos modelos de redes bayesianas para la clasificación (Arboláez Malboa 2008; Chávez Cárdenas 2009), trabajo con tipos de datos conjunto y las consecuentes particularidades que tienen que tener los algoritmos que utilicen estos tipos de datos (González Castellanos 2010).

No será necesario adquirir nuevo equipamiento para desarrollar la investigación, ya que se puede utilizar el equipamiento del Laboratorio de Inteligencia Artificial del CEI-UCLV.

Estructura de la tesis:

El presente trabajo se encuentra estructurado en introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos. En el primer capítulo se realiza un análisis de los principales métodos de aprendizaje automatizado. Se comentan algunas técnicas para la

ingeniería de los datos, métodos de aprendizaje supervisado, particularizando en aquellas técnicas que permiten trabajar con problemas con clases desbalanceadas, así como las medidas que permiten la validación de los clasificadores. El capítulo 2 aborda elementos de la hipertensión arterial y las herramientas que permiten su predicción desde la adolescencia. Se describe el conjunto de pacientes a estudiar y se muestran los resultados de las técnicas de aprendizaje previamente aplicadas. Se muestran las principales herramientas que permiten trabajar con métodos de aprendizaje automatizado. El tercer capítulo muestra los experimentos realizados a partir de 125 pacientes estudiados y los resultados de aplicar técnicas de aprendizaje automatizado a la predicción de la hipertensión arterial desde la adolescencia.

1 El aprendizaje automatizado: clasificación

El campo del aprendizaje automatizado tiene que ver con la pregunta de cómo desarrollar programas computacionales que automáticamente mejoren con experiencia. Muchas aplicaciones de aprendizaje automatizado han sido desarrolladas en los últimos años. Al mismo tiempo, han existido importantes avances en la teoría y algoritmos que constituyen el núcleo de este campo (Mitchel 1997).

El aprendizaje automatizado se relaciona con conceptos y resultados de varios campos, incluyendo la estadística, la inteligencia artificial, la filosofía, la teoría de la información, la biología, la ciencia cognitiva, la complejidad computacional y la teoría del control. La mejor manera de aprender y utilizar el aprendizaje automatizado es verlo desde todas las perspectivas y comprender la configuración del problema a resolver, los algoritmos a aplicar, y todas las consideraciones sobre estos (Mitchel 1997).

Existen varios procesos importantes que pueden mejorar significativamente los resultados de aplicar técnicas de aprendizaje automatizado o problemas prácticos de predicción, uno de ellos es la correcta estimación de los parámetros requeridos en los algoritmos a aplicar y otro lo constituye la forma de realizar la ingeniería de los datos. A continuación describiremos estos elementos y los algoritmos de aprendizaje automatizado clásicos que serán aplicados para predecir la hipertensión en la adultez a partir de indicadores presentados en la adolescencia de los pacientes. Finalmente, comentaremos brevemente cuáles son las principales herramientas que se pueden utilizar para realizar el estudio comparativo de los resultados de aplicar diversos algoritmos de aprendizaje automatizado al problema en cuestión.

1.1 Ingeniería de los datos

La ingeniería de los datos es una forma de procesar los datos de entrada de forma tal que se aplique mejor el esquema de aprendizaje escogido y las salidas resultantes sean más efectivas (Witten, Frank et al. 2011); ya que en ocasiones, la forma en que viene dada la información originalmente no es la más adecuada para adquirir conocimiento a partir de ella. En esas situaciones se hace necesaria la aplicación de algún tipo de transformación para adecuar los datos al posterior proceso de aprendizaje, como por ejemplo normalización o cambio de escala, discretización, generalización o extracción de atributos. Esta última transformación está estrechamente relacionada con la selección

de rasgos, que consiste en extraer conjuntos de atributos a partir de la combinación de los originales. Además, es posible preprocesar los conjuntos de entrenamiento a partir de la proyección de los datos, limpieza de datos y aplicación de técnicas de edición.

A continuación comentaremos cuatro formas fundamentales de preprocesar los datos, previo a la aplicación de técnicas de aprendizaje automatizado, ellas son: la normalización, la discretización, la selección de rasgos y la edición del conjunto de entrenamiento.

1.1.1 Normalización

Para la mayoría de las aplicaciones es conveniente transformar los datos de modo que el proceso de cálculo mantenga propiedades de estabilidad. En el caso más sencillo, estas transformaciones son lineales respecto a los datos de entrada, y a veces, también a los de salida. A este proceso se le llama normalización y, básicamente, consiste en la transformación de un conjunto de datos a otro, con media cero y desviación estándar uno.

Existen varias expresiones que permiten normalizar conjuntos de datos, una de las más utilizadas es la que se presenta en la expresión (1.1), donde $\bar{x}_n$ es la media de los valores de la variable x_{in} y se define según la expresión (1.2), m es la cantidad de elementos del conjunto de datos y la desviación estándar σ_n se calcula según la expresión (1.3).

$$\hat{x}_{i,n} = \frac{x_{in} - \bar{x}_n}{\sigma_n} \tag{1.1}$$

$$\bar{x}_n = \frac{1}{m}\sum_{i=1}^{m} x_{i,n} \tag{1.2}$$

$$\sigma_n = \sqrt{\frac{1}{m-1}\sum (x_{i,n} - \bar{x}_n)^2} \tag{1.3}$$

Los valores resultantes se encuentran por defecto en el rango [0,1] de los datos usados en calcular los intervalos de normalización. Excepto con la escala y los parámetros de conversión, se puede modificar eso, ya que con escala = 2.0 y la conversión = - 1.0 se obtienen valores en el rango [-1,+1].

1.1.2 Discretización

La discretización es una técnica para dividir atributos continuos en partes, en un número de intervalos y asociar cada uno con un valor discreto. Aunque los datos continuos son muy comúnes en

aplicaciones del mundo real, hay un número grande de algoritmos de clasificación, los cuáles manipulan atributos discretos solamente. Para usar estos algoritmos cuando los datos son continuos, todos los valores continuos primero deben convertirse en valores discretos. Esto provoca la necesidad de discretizar. El proceso de discretización es también beneficioso para algunos algoritmos de clasificación ya sea para aumentar la velocidad (Catlett 1991) o para mejorar la exactitud (Dougherty, Kohavi et al. 1995 1995). Por consiguiente, el proceso de discretización es una tarea esencial para preprocesar los datos.

Dougherty, Kohavi y Sahami (Dougherty, Kohavi et al. 1995 1995) han revisado una variedad de métodos de discretización y llevaron a cabo un estudio comparativo. Según su terminología, los métodos de discretización pueden estar clasificados por tres ejes diferentes: global y local, supervisado y no supervisado, y estático y dinámico.

La ventaja de los métodos globales a diferencia de los métodos locales es la independencia de los algoritmos de aprendizaje. Un conjunto de datos que es discretizado por un método global puede ser usado en cualquier algoritmo de clasificación que manipule atributos discretos. Se espera que los métodos supervisados conduzcan a un mejor desempeño comparado con los métodos no supervisados, ya que la discretización utiliza la información de las etiquetas de las clases.

1.1.3 Selección de rasgos

La selección de rasgos es una de las técnicas que más se aplica al preprocesar datos previo a la aplicación de algoritmos de aprendizaje automatizado (Caballero Mota 2008). El problema de selección de rasgos consiste en encontrar el subconjunto de atributos que mejor describe los objetos del dominio; usualmente este subconjunto se encuentra maximizando o minimizando una función objetivo. Se puede considerar como un problema de búsqueda en un espacio de estados, donde cada estado corresponde con un subconjunto de atributos, y el espacio engloba todos los posibles subconjuntos que se pueden generar (2^{n-1} para n atributos). El proceso de selección de rasgos puede entenderse como el recorrido de dicho espacio hasta encontrar un estado (combinación de rasgos) que optimice alguna función definida sobre un conjunto de atributos. Las técnicas de selección de rasgos pueden ser organizadas en tres categorías, dependiendo de cómo ellas combinen la búsqueda del subconjunto de rasgos con la construcción del modelo de clasificación: métodos de filtrado, envoltura y sistemas integrados (Saeys, Inza et al. 2007). Las técnicas de filtrado (*filter*) calculan la relevancia de los rasgos basadas solamente en las propiedades de los datos, hacen una valoración

independiente basada en las características generales de éstos. Se llama técnica de filtrado, porque el conjunto de atributos se filtra para producir el subconjunto más prometedor antes de que comience el proceso de clasificación (Witten and Frank 2005). En esta técnica lo más usual es el cálculo de la importancia de cada rasgo, de manera que el subconjunto de rasgos final esté formado sólo por los de mayor importancia, descartando el resto. Por otro lado, las técnicas de envoltura (*wrapper*) generan varios subconjuntos de rasgos y se evalúan mediante el entrenamiento y prueba del modelo de clasificación. Se hace la búsqueda del subconjunto de rasgos utilizando el error informado por el clasificador como la evaluación del subconjunto de rasgos. Como consecuencia, la selección de rasgos está directamente relacionada con un modelo de clasificación específico. Sin embargo, como el espacio de subconjuntos de rasgos crece exponencialmente con el número de rasgos, se usan métodos heurísticos para guiar la búsqueda por un subconjunto óptimo. Estos métodos de búsqueda pueden ser divididos en dos clases: determinísticos y aleatorios.

Existen diferentes enfoques y técnicas para seleccionar atributos relevantes, tales como el enfoque lógico combinatorio, el estadístico, las técnicas de optimización de colonias de hormigas, las de computación evolutiva, las basadas en la Teoría de los Conjuntos Aproximados, entre otros.

Esta reducción de dimensionalidad hecha por un proceso de selección de un subconjunto de rasgos puede traer varias ventajas a un sistema de clasificación supervisado, como una disminución en el costo de adquisición de los datos, una mejora en la comprensión del modelo final de clasificación, una inducción más rápida del modelo final de clasificación y un aumento en la exactitud del clasificador (Larrañaga, Calvo et al. 2006).

En general las técnicas de selección de rasgos ayudan, no sólo a reducir el costo de los modelos dado por la reducción de los rasgos, sino que en algunos casos puede mejorar el rendimiento. No siempre reducir rasgos es una ventaja para el problema, porque en realidad se elimina información, que no necesariamente tiene que ser irrelevante. Ésta es una gran desventaja de estos métodos, cuando las bases de datos no tienen los suficientes casos para ejemplificar la influencia de todos los rasgos, la aplicación de técnicas de selección puede eliminar rasgos importantes.

1.1.4 Edición del conjunto de datos

Las técnicas de edición se encargan de la reducción de la cantidad de objetos o ejemplos para mejorar el desempeño de los clasificadores con menor costo computacional y obtener mayor eficacia (en términos de precisión) en la clasificación; ya que existen conjuntos de entrenamiento que tienen

prototipos erróneamente etiquetados. La selección de los objetos de un dominio a incluir en un conjunto de entrenamiento (edición de conjuntos de entrenamiento) es un problema presente en todos los modelos computacionales que realizan inferencias a partir de ejemplos. Las técnicas de edición, aunque también producen la eliminación de prototipos, y con ello, la reducción de la matriz de aprendizaje, tienen como objetivo fundamental el obtener una muestra de entrenamiento de mejor calidad para una mejor precisión del sistema. Así, además de conseguir una reducción (en ocasiones notable) del conjunto de referencia, ellas proporcionan un conjunto de "calidad" que hace incrementar la precisión de los métodos de clasificación supervisada cuando se aplican sobre conjuntos editados y decrecer a su vez el costo computacional de dicha clasificación (Caballero Mota 2008).

Según (García and Villuendas 2005) los métodos de edición se pudieran agrupar en **i)** Métodos de selección de objetos (aquellos en los cuales solo se seleccionan objetos del conjunto de entrenamiento siguiendo cierto criterio) y **ii)** Métodos de construcción de objetos (aquellos en los que los nuevos prototipos pueden no estar en la matriz de entrenamiento).

Métodos de selección de objetos

El Edited Nearest Neighbor (ENN) es el primer método de edición reportado en la literatura basado en el error de clasificación. Fue creado por Wilson (Wilson 1972) y consiste en la eliminación de todos los objetos que son mal clasificados por el k-NN. Este proceso se realiza por lotes, por lo que los objetos son marcados primero, y después se eliminan simultáneamente todos los marcados.

El método All-kNN, desarrollado por Tomek (Tomek 1976), es una de las variantes del ENN de Wilson. Consiste en aplicar varias pasadas de k-NN con distintos valores de k, y eliminar cualquier objeto que haya sido mal clasificado en cualquiera de ellas. Por la heurística que utiliza, en problemas con clases muy entremezcladas puede eliminar un número excesivo de objetos.

Aha presentó una serie de algoritmos de aprendizaje basado en instancias (Instance-Based Learning, IBL) (Aha and Kibler 1991; Aha). Entre estos algoritmos se encuentran los siguientes:

- IB1 (Instance Based Llearning Algorithm 1) fue simplemente el algoritmo 1-NN.
- IB2, presentado en 1991, es un algoritmo incremental (va adicionando objetos disponibles si cumplen un determinado criterio). Kibler y Aha, en (Kibler and Aha 1987), llaman a este algoritmo el Algoritmo Creciente o Aditivo (Growth). La lista resultante es construida adicionando a ella los objetos de la matriz de entrenamiento que son mal clasificados con

respecto a los elementos que ya están en la lista. IB2 es sumamente sensible al ruido porque los casos ruidosos normalmente serán mal clasificados, y así casi siempre pueden ser retenidos.

Métodos de construcción de objetos

Koplowitz y Brown (Koplowitz and Brown 1978) hicieron una modificación del algoritmo de Wilson, le llamaron Generalized Editing (GE). Ellos estaban preocupados con la posibilidad de eliminar demasiados prototipos del conjunto de entrenamiento en el procedimiento de la Edición de Wilson. Este método consiste en eliminar algunos de los prototipos y además puede cambiar etiquetas de las clases de otros objetos. Se puede considerar como una técnica para modificar la estructura de la muestra de entrenamiento (a través de re-etiquetar algunos de los objetos de entrenamiento) y no solamente para eliminar objetos atípicos (Barandela and Gasca 2002).

Hamamoto y colaboradores (Hamamoto and Uchimura 1997) desarrollaron cuatro métodos de Bootstrap. Los objetos son construidos por la media de los k-Vecinos más Cercanos de su misma clase, de objetos seleccionados al azar. El usuario define el número de intentos que se realizarán, de los que selecciona el mejor. Según Bezdek (Bezdek and Kuncheva 2004) la familia de Bootstraps permite obtener muy buenos resultados. Este método depende fuertemente del azar y presupone la existencia de un espacio donde se defina una suma entre objetos y el producto de un objeto por un escalar para que las muestras puedan ser promediadas.

1.2 Métodos para el aprendizaje supervisado

Los métodos de clasificación son un mecanismo de aprendizaje, donde la tarea es tomar cada instancia y asignarla a una clase particular. Las clases entre las que puede elegir el procedimiento de clasificación se pueden describir en gran cantidad de formas dependiendo del problema en particular.

La clasificación puede dividirse en tres procesos fundamentales: preprocesamiento de los datos, selección del modelo de clasificación y, entrenamiento y prueba del clasificador (Bonet 2008).

Entre los métodos de clasificación más usados están las redes bayesianas, las redes neuronales artificiales, los algoritmos basados en instancias, los multiclasificadores, las máquinas de soporte vectorial, los algoritmos basados en reglas y los árboles de decisión (Mitchel 1997).

1.2.1 Redes bayesianas

Las redes bayesianas constituyen un entorno para realizar razonamiento interesante en condiciones inciertas. Ellas son redes probabilísticas que han probado ser muy útiles en la solución de problemas de diagnóstico médico, entre otras tantas aplicaciones.

Una red bayesiana es un grafo acíclico dirigido con una distribución de probabilidad asociada a cada nodo. Los nodos en la red representan las variables, atributos o rasgos del dominio de aplicación, y los arcos entre los nodos representan las relaciones de dependencia entre las variables.

Una red bayesiana es un par (D, P), donde D es un grafo acíclico dirigido, $\mathrm{P}=[\mathrm{p}(x_1|\tau_1),\ldots,\mathrm{p}(x_n|\tau_n)]$ es un conjunto de n distribuciones de probabilidad condicionales, una por cada variable x_i (nodos del grafo), y τ_i es el conjunto de padres del nodo x_i en D. El conjunto P define la distribución de probabilidad conjunta asociada, como muestra la ecuación (1.4):

$$p(x) = \prod_{i=1}^{n} p(x_i|\tau_i) \qquad x = (x_1, x_2, \ldots, x_n) \tag{1.4}$$

Los clasificadores Bayesianos producen probabilidades estimadas en lugar de clasificaciones duras. Para cada valor de una clase, ellos estiman la probabilidad que una instancia dada pertenezca a la clase. Las probabilidades pueden ser tomadas de otros clasificadores, por ejemplo, las probabilidades pueden ser obtenidas de un árbol de decisión calculando la frecuencia relativa de cada clase en una hoja y desde una lista de decisión examinando las instancias que cubren a una regla particular. Estos clasificadores son una forma alternativa de representar una distribución de probabilidad condicional.

1.2.2 Redes neuronales artificiales

Las redes neuronales artificiales constituyen una de las áreas de la Inteligencia Artificial que ha despertado mayor interés en los últimos años. Son modelos matemáticos inspirados en la Biología que han sido ampliamente utilizados en la resolución de problemas complejos. Constituyen herramientas matemáticas para el diseño de modelos que permiten obtener las relaciones existentes entre datos involucrados en tareas de regresión, clasificación, reconocimiento de patrones, etc. En particular, este enfoque resulta muy ventajoso cuando se trata de la identificación de sistemas no lineales donde la mayoría de los métodos clásicos no conducen a resultados aceptables, la razón de esto es que las redes neuronales son capaces de resolver problemas cuya solución por otros métodos convencionales resulta extremadamente difícil.

Las redes neuronales artificiales pueden ser definidas como una estructura distribuida, de procesamiento paralelo, formada de neuronas artificiales (llamadas también elementos de procesamiento), interconectadas por un gran número de conexiones (sinapsis), las cuales son usadas para almacenar conocimiento. Cada neurona recibe como entrada un conjunto de señales discretas o continuas, las pondera e integra, y transmite el resultado a las neuronas conectadas a ella. Las neuronas están conectadas entre sí en forma de un grafo acíclico dirigido.

Por lo general el flujo de cálculo se realiza desde las entradas, activando cada una de las neuronas ocultas según sean las conexiones, hasta las neuronas de salidas. Cada conexión entre dos neuronas tiene una determinada importancia asociada denominada peso sináptico o, simplemente, peso. En los pesos se suele guardar la mayor parte del conocimiento que la red neuronal tiene sobre la tarea en cuestión. El proceso mediante el cual se ajustan estos pesos para lograr un determinado objetivo se denomina aprendizaje o entrenamiento y el procedimiento concreto utilizado para ello se conoce como algoritmo de aprendizaje o algoritmo de entrenamiento.

Una red neuronal puede ser caracterizada por el modelo de la neurona, el esquema de conexión que presentan sus neuronas, o sea su topología, y el algoritmo de aprendizaje empleado para adaptar su función de cómputo a las necesidades del problema particular.

Existen varios modelos de Redes Neuronales Artificiales que se emplean en la solución de problemas de clasificación supervisada, uno de los más referenciados es el Perceptron multicapa (MLP). El MLP es reconocido como la mejor red neuronal para solucionar un problema de clasificación supervisada a partir de ejemplos, no obstante, se le señala que se comporta como una caja negra y no facilita la interpretación de los resultados.

1.2.3 Métodos basados en instancias

El algoritmo k-NN (k-Nearest Neighbor) es uno de los algoritmos tradicionales del aprendizaje basado en instancias perteneciente al enfoque perezoso. El aprendizaje en este tipo de algoritmo consiste simplemente en el almacenamiento del conjunto de datos de entrenamiento. Cuando se presenta una nueva instancia para ser clasificada son recuperadas desde la memoria las instancias similares relacionadas con esta y posteriormente utilizadas para clasificar la instancia. Una de las desventajas del enfoque basado en instancias es que el costo de clasificar una nueva instancia puede ser alto ya que todos los cálculos tienen lugar en el momento de la clasificación.

El k-NN asume que todas las instancias se corresponden a puntos en el espacio n-dimensional. El vecino más cercano a una determinada instancia está definido en términos de la distancia Euclidiana estándar. De forma más precisa, dada una instancia x descrita mediante el vector de rasgos $x_1, x_2,...,x_n$ donde x_i denota el valor del i-ésimo atributo de la instancia x, la distancia entre dos instancias x y y está definida por la expresión (1.5):

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1.5}$$

1.2.4 Métodos basados en reglas

Las reglas tienen un antecedente, o precondición, donde se verifican los valores de los rasgos predictores, mientras el consecuente, o conclusión, da la clase o las clases que se aplican a las instancias al amparo de esa regla, o quizás da una distribución de probabilidad sobre las clases. Generalmente, las precondiciones son lógicamente AND consecutivos, y todas las pruebas deben tener éxito si la regla se completa. Sin embargo, en algunas formulaciones de reglas, las precondiciones son expresiones lógicas generales en vez de conjunciones simples. Algunas reglas pueden formularse lógicamente con OR consecutivo. Así, si algún rasgo coincide con el valor de la precondición, se aplica la regla y la conclusión es aplicada a la instancia. Sin embargo, los conflictos surgen cuando varias reglas con conclusiones diferentes tienen aplicación.

Es fácil leer una de las reglas de clasificación obtenida a partir de un árbol de decisiones. Una regla es generada desde la raíz a cada hoja. El antecedente de la regla incluye una condición para cada nodo en el camino de la raíz a esa hoja, y el consecuente de la regla es la clase asignada por la hoja. Este procedimiento produce reglas que son inequívocas, en el orden en el cual son ejecutadas es irrelevante. Sin embargo, en general, las reglas que no se obtienen de árboles de decisiones son más complejas, ya que las obtenidas por árboles de decisiones son usualmente podadas para eliminar pruebas redundantes.

1.2.5 Árboles de decisión

Un árbol de decisión es un modelo de predicción que dado un conjunto de datos construye diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven

para representar y categorizar una serie de condiciones que suceden de forma sucesiva, para la resolución de un problema.

Un árbol de decisión recibe como entrada un objeto o una situación descrita por medio de un conjunto de atributos y a partir de esto devuelve una respuesta decisión. Los valores que pueden tomar las entradas y las salidas pueden ser valores discretos o continuos. También existen variantes de árboles de decisión borrosos.

El árbol de decisión suele contener nodos internos, nodos de probabilidad, nodos hojas y arcos. Un nodo interno contiene una prueba sobre algún valor de una de las propiedades. Un nodo de probabilidad indica que debe ocurrir un evento aleatorio de acuerdo a la naturaleza del problema. Un nodo hoja representa el valor que devolverá el árbol de decisión. Y finalmente las ramas brindan los posibles caminos que se tienen de acuerdo a la decisión tomada.

Cada nodo del árbol está conformado por un atributo, las ramas que salen de los nodos, corresponden a los posibles valores del atributo correspondiente. Un árbol de decisión clasifica a un ejemplar, filtrándolo de manera descendente, hasta encontrar una hoja, que corresponde a la clasificación buscada.

Los árboles de decisión resultan excelentes clasificadores, realizan el proceso de selección de rasgos embebido en la propia construcción del árbol de decisión y permiten la generación de reglas que son aplicables a la clasificación de nuevos objetos.

1.2.6 Máquinas de Soporte Vectorial

Máquina de soporte vectorial (Support Vector Machine; SVM), es una técnica de aprendizaje supervisado que se desarrolló partiendo de la teoría de aprendizaje estadístico y basada en el principio de minimización de riesgo estructural. Es usada tanto para clasificación, como para regresión (Vapnik 1995).

Concretamente, fundamenta las decisiones de clasificación, no basadas en todo el conjunto de datos, sino en un número finito y reducido de casos, que constituyen los “vectores soporte”. Las SVM puede dividirse en lineales y no lineales, estas últimas en dependencia de diferentes funciones núcleo (kernel).

Algunas de las funciones núcleo más comúnmente usadas son la polinomial y la gaussiana de base radial o también conocida como función de base radial (Radial Basic Function; RBF), que se muestran en las ecuaciones (1.6) y (1.7),respectivamente.

$$k(x, x') = \langle x \cdot x' \rangle^d \tag{1.6}$$

$$k(x, x') = exp\left(\frac{\|x - x'\|}{2\sigma^2}\right) \tag{1.7}$$

1.2.7 Multiclasificadores

A pesar de los muchos estudios hechos hasta la actualidad en relación a los clasificadores no existe todavía uno por excelencia, por lo que se hace difícil seleccionar el clasificador que logre encontrar una mejor frontera de decisión para separar las clases. En la búsqueda de mejores métodos de clasificación aparece una tendencia a combinar varios clasificadores en el mismo problema. En esto último se basan los algoritmos llamados multiclasificadores, utilizar varios clasificadores y combinar sus diferentes salidas con el objetivo de alcanzar un mejor resultado (Polikar 2006).

Un sistema multiclasificador puede ser mejor que un clasificador simple, ya que algunos algoritmos ejecutan búsquedas que pueden llevar a diferentes óptimos locales: cada clasificador comienza la búsqueda desde un punto diferente y termina cercano al óptimo. Existe la expectativa de que alguna vía de combinación puede llevar a un clasificador con una mejor aproximación.

Hay una serie de algoritmos desarrollados, algunos para problemas generales como *bagging* y *boosting* y otros para problemas específicos, pero todos tienen como partes fundamentales: la selección de los clasificadores de base y la elección de la forma de combinar las salidas (Bonet 2008).

Bagging (Breiman 1996) es uno de los primeros algoritmos de multiclasificadores, se basa en crear diferentes conjuntos de entrenamiento, extraídos del conjunto inicial de manera aleatoria y con reemplazo, con lo cual asegura la diversidad. Este modelo necesita la selección de un modelo de clasificador inestable, o sea, un modelo que con pequeños cambios obtenga valores diferentes (Witten and Frank 2005). Además usa un único modelo de clasificador y la combinación de los clasificadores resultantes se realiza con la técnica de voto mayoritario. Puede ser aplicado en métodos de aprendizaje con predicciones numéricas, en los cuales las salidas individuales, que son números reales, son promediadas. *Bagging* usa para combinar las salidas voto mayoritario pero, en aras de lograr un mayor rendimiento, comúnmente es usado un estimado de probabilidad en lugar de

una salida concreta, estas probabilidades estimadas por los modelos son promediadas (Witten and Frank 2005) y la clase más probable es asignada. Variaciones a este algoritmo es el caso de *ramdom forests* (Breiman 2001) que es construido con árboles de decisión como modelo de clasificador, y para crear los conjuntos de entrenamiento, usa el método de muestreo con reemplazo como en *bagging*, o puede usar subconjuntos de rasgos.

Boosting (Schapire 1990) es parecido a *bagging* porque usa el método de crear bases de entrenamiento aleatorias con reemplazo, a partir de la base original y un único modelo de clasificación para los clasificadores de base. Sin embargo, este algoritmo se realiza de manera secuencial, donde los clasificadores se van entrenando uno detrás del otro porque usan información del anterior. Otra diferencia es que *boosting* le da un peso al modelo por su rendimiento, en lugar de dar peso igual a todos los modelos. El reemplazo es realizado estratégicamente de forma que los casos mal clasificados tienen mayor probabilidad, que los bien clasificados, de pertenecer al conjunto de entrenamiento del siguiente clasificador del sistema. Existen muchas variantes que utilizan la idea de *boosting*, siendo *AdaBoost* una de la más utilizada (Freund and Schapire 1997). Es una versión más general que se ha dividido en *AdaBoost.M1* y *AdaBoost.R*, para problemas de clasificación y de regresión respectivamente. *AdaBoost* usa como método de combinación el voto mayoritario pesado.

Existen muchos modelos de multiclasificadores que utilizan subconjuntos de rasgos diferentes como los descritos por Kuncheva (Kuncheva 2004). La elección de estos subconjuntos de rasgos puede ser realizada de diferentes vías. Entre ellas podemos mencionar la selección aleatoria (*Random Selection*), conocida también como método del subespacio aleatorio (*ramdom subspace method*), donde cada clasificador es construido sobre un subconjunto aleatorio de rasgos de tamaño d. Se han obtenido buenos resultados utilizando clasificadores con $d=n/2$ rasgos, donde n es el número total de rasgos. Con el método de selección aleatoria se han obtenido buenos resultados cuando hay información redundante la cual está dispersa por todos los rasgos en lugar de estar concentrada en un subconjunto (Ho 1998; Skurichina 2001). Otro método propone combinar *bagging* con selección aleatoria (*Random Selection*). Se seleccionan B subconjuntos aleatorios de la base de entrenamiento y para cada uno de ellos se seleccionan R subconjuntos de rasgos. El multiclasificador consta de $L=B+R$ clasificadores. Esta combinación de *bagging* con selección aleatoria busca alcanzar una mayor diversidad entre los clasificadores que con la utilización de cualquiera de los dos de manera

independiente. Se ha demostrado que esta combinación es más ventajosa que si se usan por separado.

1.3 Peculiaridades de los problemas con clases desbalanceadas

En algunos problemas de clasificación la cantidad de instancias que presentan las clases son diferentes. Específicamente, cuando se enfrentan a un conjunto de datos con sólo dos clases, el problema de desbalance se produce cuando una clase es representada por un gran número de instancias, mientras que la otra está representada sólo por unos pocos, tal y como se muestra en la Figura 1.1, tomada de (Chawla 2004; Fernández 2010).

Muchos ejemplos negativos (clase mayoritaria) frente a pocos y dispersos ejemplos positivos

Clases balanceadas con clústeres bien definidos

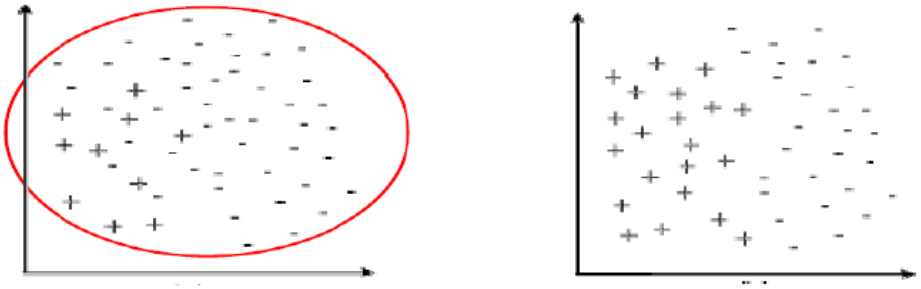

Figura 1.1 Representación gráfica del problema de desbalance de clases.

El problema del desbalance en los conjuntos de datos es muy importante (Yang and Wu 2006), ya que está implícito en la mayoría de aplicaciones del mundo real, tales como la clasificación de imágenes de satélite (Suresh 2008), la gestión de riesgos (Huang 2006), los datos de la proteína (Provost 2001) y, en particular en las aplicaciones médicas (Kilic 2007; Mazurowski 2008; Peng 2008). Es importante señalar que la clase minoritaria por lo general representa el concepto de interés, por ejemplo en nuestro estudio los pacientes que llegaron a ser hipertensos, mientras que la otra clase representa la contrapartida de ese concepto (los pacientes saludables), como se detallará en el capítulo 2.

Los algoritmos de clasificación estándar tienen un sesgo hacia la clase mayoritaria, ya que las reglas que predicen el mayor número de ejemplos son ponderados positivamente durante el proceso de aprendizaje en favor de la precisión métrica. En consecuencia, las instancias que pertenecen a la clase minoritaria están mal clasificadas con mayor frecuencia que las que pertenecen a la clase mayoritaria (Fernández 2010).

Un gran número de métodos se han propuesto para tratar el problema del desbalance de las clases. Estos enfoques se pueden clasificar en dos grupos: los modelos internos que crean nuevos algoritmos o modifican los ya existentes teniendo en cuenta el problema del desbalance entre las clases (Barandela 2003; Wu and Chang 2005; Xu 2007) y enfoques externos que preprocesan los datos con el fin de disminuir el efecto de su desbalance (Estabrooks 2004). Además, son sensibles a los costos de soluciones de aprendizaje que incorporan los datos y métodos algorítmicos al nivel de asumir mayores costos de clasificación errónea de las muestras de la clase minoritaria y buscar para minimizar los errores de alto costo (Sun 2007).

1.3.1 Métodos de edición

Entre los métodos reportados en la literatura con mejores resultados en el preprocesamiento de estos tipos de datos encontramos: SMOTE (Chawla 2002), SMOTE-Tomek Links (Batista 2004), SMOTE-ENN (Batista 2004), Borderline-SMOTE1 (Han 2009) y Safe-Level-SMOTE (Bunkhumpornpat 2009).

SMOTE

SMOTE (Synthetic Minority Over-sampling Technique) es un método de pre-procesamiento de los datos para balancear la distribución de clases para cambiar el espacio de los datos al aplicar clasificadores usuales en dominios desbalanceados. SMOTE genera instancias positivas de datos de otras instancias en el conjunto de datos original seleccionando los k vecinos más cercanos y usándolos para realizar operaciones aritméticas para generar nuevas instancias.

Borderline_SMOTE

Un nuevo método de aprendizaje que toma muestras sobre conjuntos de datos desbalanceados. Balancea la distribución de clases para cambiar el espacio de los datos al aplicar clasificadores usuales en dominios desbalanceados. *Borderline_SMOTE* genera instancias positivas de datos de otras instancias en el conjunto de datos original seleccionando los k vecinos más cercanos y usándolos para realizar operaciones aritméticas para generar nuevas instancias.

SPIDER2

Es un método de preprocesamiento de selección de instancias. Tiene el objetivo de filtrar y reetiquetar difíciles ejemplos de clases mayoritarias y local *over-sampling* de clases minoritarias. Este método consiste en dos fases que corresponden al preprocesamiento de clases mayoritarias y

minoritarias respectivamente. En la primera fase, se indican las características de ejemplos de la clase mayoritaria, y elimina o reetiqueta los ejemplos ruidosos de una de estas. En la segunda fase, se identifican las características de ejemplos de la clase minoritaria considerando los cambios introducidos en la primera fase. Entonces, los ejemplos ruidosos son amplificados reproduciendo la clase minoritaria.

Safe Level SMOTE

Nivel seguro de SMOTE para trabajar con el problema de clases desbalanceadas. Método de preprocesamiento del conjunto de datos. Balancea la distribución de la clase para modificar el espacio de los datos al aplicar los clasificadores usuales en los dominios desbalanceados. Safe Level SMOTE va generando instancias de datos positivos de otras instancias en el conjunto de datos original seleccionando los k vecinos más cercanos y usándolos para realizar las operaciones aritméticas para generar la nueva instancia.

SMOTE (Synthetic Minority Over-sampling Technique) + ENN (Edited Nearest Neighbor)

La motivación detrás de este método es similar a SMOTE_Tomek Links. ENN tiende a eliminar más ejemplos que los efectuados por Tomek Links, así que se espera que proporcione más a fondo la limpieza de datos. En contraste con NCL que es un método de *undersampling,* ENN es utilizado para eliminar los ejemplos en ambas clases. Así, cualquier ejemplo que es clasificado por sus tres vecinos más cercanos está distante del conjunto de entrenamiento.

SMOTE-RSB*(Preprocessing using SMOTE and Rough Sets Theory)

Este método es una nueva propuesta de híbridos que usa el SMOTE (Ramentol, Bello et al. 2010) y trabaja sobre instancias sintéticas para conjuntos de datos de gran desbalance y se basa en dos pasos fundamentales:

- Construir nuevos ejemplos sintéticos de la clase minoritaria usando el SMOTE.
- Mejorar la calidad de estas nuevas muestras a través de las técnicas basadas en la edición con el (Rough Sets Theory, RST) y la más baja aproximación de un subconjunto, actuando sobre instancias artificiales de la clase minoritaria creadas por el algoritmo SMOTE.

SMOTE (Synthetic Minority Over-sampling Technique) + TL (Tomek Links)

Frecuentemente, no se definen bien los grupos de las clases ya que algunos ejemplos de la clase mayoritaria podrían estar invadiendo el espacio de la clase minoritaria. Al contrario también puede

ser verdadero. Para crear clases bien definidas se propone aplicar *Tomek Links* al conjunto de entrenamiento y posteriomente SMOTE.

1.3.2 Clasificadores

Clasificadores estándares como las redes neuronales artificiales, las máquinas de soporte vectorial y el C4.5 han sido adaptados en varias investigaciones para el tratamiento de problemas con clases desbalanceadas (Visa and Ralescu 2005). A continuación mencionaremos algunos algoritmos para manejar clases desbalanceadas a partir de adaptaciones a métodos que generan árboles de decisión principalmente, ya que estos no presentan muchos problemas con el desbalance de clases, por lo que se logra predecir correctamente mayor cantidad de instancias.

AdaBoost usa todo el conjunto de datos para entrenar cada clasificador, pero después de cada iteración, da mayor peso a las instancias mal clasificadas, con la meta de clasificar los ejemplos correctamente en la próxima iteración que eran incorrectamente clasificados durante la iteración actual. Después de cada iteración se disminuyen los pesos de instancias correctamente clasificadas. Además, otro peso se asigna a cada clasificador individual que depende de su exactitud global que se usa entonces en la fase de la prueba; más confianza se da a los clasificadores más exactos. Finalmente, cuando un nuevo caso se somete, cada clasificador da un voto pesado, y la etiqueta de la clase se selecciona por la mayoría (Freund and Schapire 1997). Dos mejoras a este método son *AdaBoostM1* y *AdaBoostM2*.

AdaC2 produce un conjunto de árboles de decisión de un conjunto de ejemplos dados. Cada caso tiene un costo asociado que depende de la clase. Intenta resolver el problema del desbalance aumentando el peso de las instancias de la clase minoritaria en cada iteración del algoritmo *boosting*.

Por otro lado, *Bagging* o *bootstrap* consiste en agregar diferentes entrenamientos a los clasificadores con las réplicas del *bootstrapped*, formando un nuevo conjunto de datos para entrenar cada clasificador aplicando *resampling*. *BalanceCascade* es otra propuesta que lleva a cabo un doble conjunto de aprendizaje, es decir, combina *bagging* y *boosting* (también con una técnica de preprocesamiento). Trabaja de una manera dirigida, y por consiguiente los clasificadores tienen que ser entrenados secuencialmente. Una propuesta similar es *Easy Ensemble*.

Por otra parte, *Data Boost-IM* combina el algoritmo de *AdaBoost.M1* con una estrategia de generación de datos. Identifica los ejemplos difíciles primero y entonces lleva a cabo un proceso de balance, siempre para ambas clases.

MSMOTEBoost introduce las instancias sintéticas en cada iteración de *AdaBoost.M2*, usando el algoritmo de datos preprocesados *MSMOTE*. Los pesos de las nuevas instancias son proporcionales al número total de instancias en el nuevo conjunto de datos. Sus pesos siempre son el mismo (en todas las iteraciones y para todas las instancias nuevas), considerando que los pesos de las instancias del conjunto de datos original se normalizan de semejante manera que ellos forman una distribución con las nuevas instancias. Después de entrenar un clasificador, se ponen al día los pesos de las instancias del conjunto de datos originales, entonces otra prueba a la fase es aplicando (de nuevo, modificando la distribución de peso). La repetición de este proceso también trae más diversidad en los datos de entrenamiento que generalmente benefician el aprendizaje del conjunto.

RUSBoost elimina las instancias de las clases mayoritarias al azar por el *undersampling* en el conjunto de datos en cada iteración. En este caso, no es necesario asignar los nuevos pesos a las instancias. Es suficiente normalizar los pesos de las instancias restantes simplemente en el nuevo conjunto de datos con respecto a la suma total de pesos. Después de entrenar un clasificador, se actualizan los pesos de las instancias del conjunto de datos originales y se repite el proceso, logrando mayor diversidad en los datos de entrenamiento que generalmente benefician el aprendizaje del conjunto. Otro método que utiliza *undersampling* es *Under Bagging*. Consiste en ir disminuyendo el número de instancias de la clase de mayoritaria por eliminación al azar, en este caso todas las instancias de la clase minoritarias son incluidas en el nuevo *bootstrap*. Otra propuesta que realiza *undersampling* es *Under Bagging2.* Mientras que *Under Over Bagging* hace uso de técnicas de *oversampling* y de *undersampling*. Otras propuestas son los métodos *II Votes* y *Over Bagging*.

1.4 Medidas para la validación de clasificadores

Las medidas de la calidad de la clasificación se construyen a partir de una matriz de confusión, como se muestra en la Tabla 1.1, que registra correctamente e incorrectamente los ejemplos reconocidos de cada clase.

Tabla 1.1. Matriz de confusión para problemas con dos clases.

	Predicciones Positivas	**Predicciones negativas**
Clase Positiva	Verdaderos Positivos (VP)	Falso Negativo (FN)
Clase Negativa	Falsos Positivos (FP)	Verdaderos Negativos (VN)

A partir de la Tabla 1.1 se pueden obtener cuatro indicadores del desempeño que miden la calidad de la clasificación para las clases positivas y negativas de forma independiente:

- Tasa de verdaderos positivos $TVP = \frac{VP}{(VP+FN)}$: es el porcentaje de casos positivos correctamente clasificados como pertenecientes a la clase positiva.
- Tasa de verdaderos negativos $TVN = \frac{VN}{(FP+VP)}$: es el porcentaje de casos negativos correctamente clasificados como pertenecientes a la clase negativa.
- Tasa de falsos positivos $TFP = \frac{FP}{(FP+VN)}$: es el porcentaje de casos negativos mal clasificados como pertenecientes a la clase positiva.
- Tasa de falsos negativos $TFN = \frac{FN}{(VP+FN)}$: es el porcentaje de casos positivos mal clasificados como pertenecientes a la clase negativa.

Sin embargo, la evaluación de los procesos de aprendizaje con datos desbalanceados tiene sus propias características. Por ejemplo, la medida empírica más usada que es la Exactitud (Accuracy), según expresión (1.8), no distingue entre el número de etiquetas correctas de diferentes clases, lo cual en el ámbito de los problemas de desbalance puede conducir a conclusiones erróneas. Por ejemplo, un clasificador que obtiene una precisión de 90% en un conjunto de datos con un valor de IR=9 (*Imbalance Rate, IR*) (Orriols-Puig and Bernad´o-Mansilla 2009), podría no ser válida si no se clasifica correctamente cualquier instancia de la clase minoritaria.

$$ACC = \frac{VP+VN}{VP+FN+FP+VN} \tag{1.8}$$

Una medida apropiada que podría ser utilizada para medir el rendimiento de la clasificación de conjuntos de datos con mayor desbalance es el área bajo la curva ROC (Receiver Operating Characteristic) (Bradley 1997). En estos gráficos se reconoce el hecho de que la capacidad de cualquier clasificador no puede aumentar el número de verdaderos positivos sin aumentar los falsos positivos. El área bajo la curva ROC (AUC) (Huang and Ling 2005) proporciona un número único de resumen para el desempeño de algoritmos de aprendizaje. Para calcular AUC sólo hay que obtener el área bajo la curva como se muestra en la expresión (1.9).

$$AUC = \frac{1+TVP-TFP}{2} \quad (1.9)$$

1.5 Consideraciones finales del capítulo

Al aplicar técnicas de aprendizaje supervisado no solo es necesario seleccionar correctamente los algoritmos a aplicar así como la definición de sus parámetros, sino que es imprescindible realizar una correcta ingeniería de los datos, donde algunos métodos a aplicar pueden ser la normalización, discretización, selección de rasgos y edición del conjunto de entrenamiento.

Se identificaron los algoritmos de clasificación clásicos basados en redes bayesianas, redes neuronales, basados en instancias, basados en reglas, árboles de decisión, máquinas de soporte vectorial y metaclasificadores que combinan estos métodos para su posible aplicación en la predicción de adultos hipertensos a partir de adolescentes prehipertensos.

Adicionalmente, se estudiaron los clasificadores y técnicas de edición para problemas con clases desbalancedas, ya que el problema a estudiar tiene estas características. Uno de los métodos que mejores resultados arroja al aplicarse a problemas con clases desbalancedas son aquellos que generan árboles de decisión, no obstante, también han sido modificadas algunas redes neuronales artificiales y máquinas de soporte vectorial con tales propósitos.

Finalmente, se precisaron algunas medidas que permiten reflejar la calidad de los resultados de la clasificación, señalando las particularidades al evaluar los resultados de clasificadores a problemas con clases desbalanceadas considerando el área bajo la curva ROC (AUC).

2 La hipertensión arterial y herramientas que permiten su predicción desde la adolescencia

En este capítulo presentaremos las características generales de la hipertensión arterial, las grandes afectaciones que provoca en la población y la necesidad del descubrimiento temprano de estados anormales de presión arterial, sobre todo desde la infancia y la adolescencia, para evitar la aparición de daños en órganos diana. Mostraremos los resultados de algunos trabajos precedentes que permiten predecir la hipertensión arterial en la adultez desde la adolescencia. Describiremos el conjunto de datos que constituye nuestro objeto de estudio en este proceso predictivo. Finalmente, comentaremos los elementos fundamentales de herramientas que permiten aplicar métodos de aprendizaje automatizado y que pudieran ser útiles para realizar nuevos experimentos acerca de la predicción de la hipertensión arterial en la adultez desde la adolescencia.

2.1 La hipertensión arterial

La hipertensión arterial (HTA) es una de las principales causas de morbilidad y mortalidad para cualquier nivel de atención sanitaria independientemente del tipo de condición socioeconómica imperante.

Cifras alarmantes indican que podría existir un incremento de hasta el 60 % de hipertensos a nivel mundial para el año 2025, lo que significaría el aumento de la población que la padece de un billón en la actualidad hasta 1,65 billones para ese año.

La comunidad médica ha considerado desde antaño a la HTA esencial como una entidad propia del adulto. La publicación, en 1977, de la Primera Tarea de la Fuerza Norteamericana para el diagnóstico, evaluación y tratamiento de la presión arterial (Fayyad, Piatetsky-Shapiro et al. 1996) en niños y adolescentes, actualizada después en 1987, 1996 y, por último, en el año 2004, propició el inicio de una nueva era en el enfrentamiento de esta enfermedad desde el referente de la cardiología preventiva.

Con la implementación en Cuba del Programa de Atención del Médico y la Enfermera de la Familia, como parte del perfeccionamiento del enfoque social de la atención médica, las acciones de salud dirigidas a la detección y control de la HTA en la adultez han sido considerables. El país ha realizado esfuerzos concretos para lograr un mejor conocimiento sobre la entidad; ejemplo de ello constituyó

su participación, en 1974, junto a otros 13 países en el proyecto "Programa para el control comunitario de la HTA", que sirvió como antecedente al Primer Programa Nacional para la Prevención y Control de la HTA; la realización en 1995 de la Primera Encuesta Nacional de Factores de Riesgo (ENFR) y actividades preventivas, así como el desarrollo del Segundo Taller Nacional sobre HTA en 1997; la conformación del Programa Nacional para la Prevención, Diagnóstico, Evaluación y Control de la HTA en 1998, y la realización en el 2001 de la Segunda ENFR.

Sin embargo, en la última década, el reconocimiento y las políticas sanitarias por parte del Sistema Nacional de Salud dirigidas hacia la HTA esencial desde la infancia y sus negativas consecuencias, no se han comportado como en la adultez. A principios del año 2006 se emite la nueva Guía Cubana para la Prevención, Diagnóstico, y Tratamiento de la HTA que fue actualizada en el 2008; la cual hace evidente las exiguas referencias a trabajos nacionales entorno a la HTA esencial desde la niñez y la adolescencia.

Cabal y colaboradores en Ciudad de La Habana, estudiaron 302 estudiantes de secundaria básica en edades entre los 12 y 14 años y determinaron una prevalencia de HTA de un 0,66%; a diferencia de Suárez *y* colaboradores en Santiago de Cuba, quienes encontraron que el 9,3 % de los adolescentes de 15 a 17 años eran hipertensos. Por su parte, Álvarez *y* colaboradores, en un estudio con 344 adolescentes entre 12 y 16 años en La Habana hallaron una prevalencia de HTA y prehipertensión (preHTA) de 4,7 % y 18,6 %, respectivamente. Los trabajos nacionales con resultados relacionados con el tema en la niñez se limitan a exponer los factores de riesgo encontrados, con un predominio del sobrepeso y la obesidad y comorbilidades asociadas, como la diabetes mellitus (DM).

No existen en Cuba informes de estudios epidemiológicos de corte longitudinal que permitan conocer la evolución clínica de los factores de riesgo desde la adolescencia hasta la adultez. En el municipio de Santa Clara se ha realizado un trabajo sostenido en la atención al paciente hipertenso adulto. No obstante, en la etapa de la adolescencia persisten deficiencias en la detección y tratamiento de la enfermedad en el nivel primario de atención, lo que se ha atribuido a insuficiencias en los conocimientos sobre la enfermedad por parte del personal de salud que atiende al paciente adolescente.

La necesidad del descubrimiento temprano de estados anormales de presión arterial, sobre todo desde la infancia, es indispensable ante la aparición de daños en órganos diana (corazón, riñón, cerebro, entre otros.).

2.2 Descripción del conjunto de pacientes estudiados

En este epígrafe describiremos el estudio realizado en (Pérez Fernández 2011) desde el universo considerado hasta la selección de la muestra que finalmente será sometida a la aplicación de métodos de clasificación supervisada.

Según (Pérez Fernández 2011), la investigación comenzó con la selección de tres escuelas del municipio Santa Clara empleando la técnica de muestreo por conglomerados al azar, entre las 11 de su tipo que existían en el año de comienzo de la investigación; en dichas escuelas se estableció un universo de 3 007 adolescentes registrados en el proyecto de Investigación "Pesquisaje Escolar en la Adolescencia de Hipertensión Arterial" con edades comprendidas entre 12 y 15 años (ambas incluidas). Se estableció como condición que no existiera diagnóstico previo de preHTA o HTA. En la primera etapa, luego de seleccionar las tres ESBU, teniendo en cuenta la proporción de matrícula de cada una de ellas, se escogió de manera aleatoria simple una muestra de 1 545 adolescentes pertenecientes al universo declarado de 3 007.

A los adolescentes escogidos, se les aplicó la encuesta clínico-epidemiológica que permitió el diagnóstico de los 513 adolescentes identificados como prehipertensos (385) e hipertensos (128).

La segunda etapa de la investigación correspondió al seguimiento clínico de los adolescentes hasta la etapa de la adultez. Del total de 385 individuos prehipertensos identificados en la primera etapa, se escogió por muestreo aleatorio simple a 125 adolescentes.

Los 125 individuos diagnosticados como prehipertensos en la adolescencia durante el pesquisaje escolar (rango de edades: entre 12 y 15 años) fueron seguidos anualmente durante ocho anos hasta la adultez (marzo 2001-marzo 2009) en la consulta de Cardiología del Hospital Universitario "Celestino Hernandez Robau" de la ciudad de Santa Clara, provincia de Villa Clara, Cuba. Tras este período de tiempo se investigó, la presencia de conversión a HTA establecida. El 100% de los pacientes que resultaron hipertensos fueron diagnosticados, despúes de la edad de 20 años (la edad superior en la adultez correspondió a los 23 años). No se constataron cambios significativos en sus dimensiones corporales, teniendo en cuenta la normal evolución del peso y la talla, con la edad del individuo. Del mismo modo, ningún sujeto desarrolló enfermedad crónica diferente de la presencia de HTA, durante el período evaluado. Más de la mitad, el 65.60% de los adolescentes prehipertensos evolucionaron tras 96 meses de seguimiento, a adultos hipertensos. Mientras que el 34.40% permaneció en categoría de prehipertensos.

Los rasgos considerados para cada paciente se describen en la Tabla 2.1, la Tabla 2.2, la Tabla 2.3 y la Tabla 2.4.

Tabla 2.1 Rasgos relacionados con la presión arterial.

	Rasgo	**Definición**	**Tipo**
1	Normotensión arterial	Valores de PAS o PAD menores del 90 percentil en tres o más ocasiones, según los patrones establecidos en el Cuarto Reporte de HTA	Cualitativa nominal dicotómica (si o no)
2	Prehipertensión arterial (preHTA)	Valores de PAS o PAD mayores o igual al 90 percentil y menor o igual al 95 percentil en tres o más ocasiones, según los patrones establecidos en el Cuarto Reporte Americano de HTA	Cualitativa nominal dicotómica (si o no)
3	Hipertensión arterial (HTA)	Valores de PAS o PAD por encima del percentil 95 en tres o más ocasiones, según los patrones establecidos en el Cuarto Reporte Americano de HTA y la Guía europea para el tratamiento de la HTA en la niñez y adolescencia	Cualitativa nominal dicotómica (si o no)

Tabla 2.2 Rasgos sociodemográficos y relacionados con la actividad escolar.

	Rasgo	**Definición**	**Tipo**
4	Edad	Años cumplidos hasta el día del interrogatorio, según la fecha de nacimiento	Cuantitativa (número de años cumplidos desde el nacimiento)
5	Sexo	Según sexo biológico de pertenencia	Cualitativa nominal dicotómica (masculino o femenino)
6	Color de la piel	Acorde al fenotipo	Cualitativa nominal dicotómica (blanca o no blanca)
7	Ambiente familiar	Caracterización integral cualitativa del ambiente familiar contenida en el Expediente Académico Escolar, realizada por el profesor guía del grupo escolar al que pertenecía el educando, previa discusión con el Equipo de Salud Escolar Municipal	Cualitativa nominal dicotómica (desfavorable o favorable)
8	Rendimiento académico	Caracterización docente contenida en el Expediente Académico Escolar elaborada por el profesor guía del grupo escolar a que pertenecía el educando, relacionada con el resultado docente del mismo, de acuerdo con lo que establece la Resolución Ministerial 216/89[1].	Cualitativa nominal dicotómica (desfavorable si el promedio académico del estudiante es inferior a 60 puntos, o favorable si el promedio académico del estudiante es superior a 60 puntos)

[1] Ministerio de Educación. Resolución Ministerial Nº 216/89 sobre la evaluación. La Habana: MINED; 1989. Ministerio de Educación.

Tabla 2.3 Rasgos epidemiológicos relacionados con los antecedentes familiares.

	Rasgo	**Definición**	**Tipo**
9	Antecedente patológico familiar de HTA	Existencia de antecedentes familiares de primer y segundo grados de consanguinidad diagnosticados con HTA	Cualitativa nominal dicotómica (si o no)
10	Antecedente patológico familiar de obesidad	Existencia de antecedentes familiares de primer y segundo grados de consanguinidad diagnosticados con obesidad	Cualitativa nominal dicotómica (si o no)
11	Antecedente patológico familiar de diabetes mellitus	Existencia de antecedentes familiares de primer y segundo grados de consanguinidad diagnosticados con diabetes mellitus	Cualitativa nominal dicotómica (si o no)
12	Antecedente patológico familiar de cardiopatía isquémica	Existencia de antecedentes familiares de primer y segundo grados de consanguinidad diagnosticados con cardiopatía isquémica	Cualitativa nominal dicotómica (si o no)
13	Antecedente patológico familiar de insuficiencia cardíaca	Existencia de antecedentes familiares de primer y segundo grados de consanguinidad diagnosticados con insuficiencia cardíaca	Cualitativa nominal dicotómica (si o no)
14	Antecedente patológico familiar de accidente vascular encefálico	Existencia de antecedentes familiares de primer[2] y segundo[3] grados de consanguinidad, diagnosticados con accidente vascular encefálico de causa hipertensiva	Cualitativa nominal dicotómica (si o no)

Tabla 2.4 Rasgos epidemiológicos relacionados con el peso corporal del adolescente.

	Rasgo	**Definición**	**Tipo**
15	Normopeso	Índice de masa corporal menor del 90 percentil según las tablas de crecimiento de la Organización Mundial de la Salud.	Cualitativa nominal dicotómica (si o no)
16	Sobrepeso	Índice de masa corporal mayor o igual del 90 e inferior al 95 percentil, según las tablas de crecimiento de la Organización Mundial de la Salud	Cualitativa nominal dicotómica (si o no)
17	Obesidad	Índice de masa corporal mayor o igual del 95 percentil según las tablas de crecimiento de la Organización Mundial de la Salud	Cualitativa nominal dicotómica (si o no)
18	Bajo peso al nacer	Recién nacido con un peso al nacer inferior a 2 500 gramos con independencia de la edad gestacional, según el Programa Cubano de Bajo Peso al Nacer	Cualitativa nominal dicotómica (si o no)

[2] Padre, madre y hermano.
[3] Abuelos paternos y maternos.

2.3 Resultados de técnicas de aprendizaje supervisado previamente aplicadas al conjunto de datos en estudio

En (Pérez Fernández 2012) se presentan análisis multivariados realizados sobre el conjunto de datos de los 125 paciente, que dan lugar a la definición de dos criterios diferentes para un índice integral de riesgo.

El primero de ellos se basa en la V de Cramer y consiste esencialmente en una suma de los riesgos presentes, suma que es ponderada por el valor de la V de Cramer de cada riesgo.

Entonces, se pronosticaron como sanos 23 pacientes que tienen realmente HTA (falsos negativos) y como enfermos 25 que sólo tienen preHTA (falsos positivos). Acertaron realmente con 59 (verdaderos positivos) y 18 (verdaderos negativos) para obtener una exactitud o porcentaje de buena clasificación de (50+18)/125=61.6%

El otro análisis realizado fue la obtención de un modelo predictivo a partir de un árbol de decisiones, mediante un proceso de búsqueda de interacciones entre los factores a partir de la técnica de CHAID (chi-square automatic interaction detector) y finalmente, se elabora un modelo predictivo basado en reglas de decisiones en el cuál no se forzó siquiera la entrada de la primera variable, sino todo es automáticamente detectado a partir de la significación del Chi-cuadrado. Entonces, se obtiene de este modo un arbol de decisiones que fue capaz de clasificar acertadamente al 80 % de los casos. Este resultado es el que se refleja en la Tabla 2.5 como un modelo predictivo o *score* basado en reglas de decisiones.

Tabla 2.5 Resultados de la aplicación de la técnica CHAID.

Grupos Reales	**Predicho**		**Porcentaje correcto**
	No (Solo preHTA)	**Si (HTA)**	
No (Solo preHTA)	29	14	67.4
Si (HTA)	11	71	86.6
Porcentaje general	32.0	68.0	**80.0**

2.4 Herramientas que permiten aplicar métodos de aprendizaje automatizado

En este epígrafe describiremos los elementos fundamentales de herramientas que permiten aplicar métodos de aprendizaje automatizado y que pudieran ser útiles para realizar nuevos experimentos acerca de la predicción de la hipertensión arterial en la adultez desde la adolescencia.

2.4.1 Weka

Weka es un software libre y su código fuente está totalmente disponible, permitiendo la modificación del mismo. Al ser un software de distribución gratuita posibilita su uso, copia, estudio, modificación y redistribución bajo la licencia GNU.

Weka es un sistema multiplataforma y de amplio uso probado bajo sistemas operativos Linux, Windows y Macintosh. Puede ser usado desde la perspectiva de usuario mediante las seis interfaces que brinda, a través de la línea de comando desde donde se pueden invocar cada uno de los algoritmos incluidos en la herramienta como programas individuales y mediante la creación de un programa Java que llame a las funciones que se desee.

Weka (versión 3.5.2) dispone de seis interfaces de usuario diferentes que pueden ser accedidas mediante la ventana de selección de interfaces (GUI Chooser), que constituye la interfaz de usuario gráfica (GUI: Grafic User Interface):

- Interfaz para línea de comando (Simple CLI: Command Line Interface): permite invocar desde la línea de comandos cada uno de los algoritmos incluidos en Weka como programas individuales. Los resultados se muestran únicamente en modo texto. A pesar de ser en apariencia muy simple es extremadamente potente porque permite realizar cualquier operación soportada por Weka de forma directa; no obstante, es muy complicada de manejar ya que es necesario un conocimiento completo de la aplicación. Su utilidad es pequeña desde que se fue recubriendo Weka con interfaces. Actualmente ya prácticamente sólo es útil como una herramienta de ayuda a la fase de pruebas. Es muy beneficiosa principalmente para los sistemas operativos que no proporcionan su propia interfaz para línea de comandos.

- Explorador (Explorer): interfaz de usuario gráfica para acceder a los algoritmos implementados en la herramienta para realizar el aprendizaje automatizado. Es el modo más usado y descriptivo. Permite realizar operaciones sobre un sólo archivo de datos.

- Experimentador (Experimenter): facilita la realización de experimentos en lotes, incluso con diferentes algoritmos y varios conjuntos de datos a la vez.

- Flujo de conocimiento (KnowledgeFlow): proporciona una interfaz netamente gráfica para el trabajo con los algoritmos centrales de Weka. Esencialmente tiene las mismas funciones del Explorador aunque algunas de ellas aún no están disponibles. El usuario puede seleccionar los

componentes de Weka de una barra de herramientas, y conectarlos juntos para formar un "flujo del conocimiento" que permitirá procesar y analizar datos.

- Visualizador de Arff (ArffViewer): interfaz para la edición de ficheros con extensión *arff*.
- Log: muestra la traza de la máquina virtual de acuerdo a la ejecución del programa.

Weka denomina instancia a cada uno de los casos proporcionados en el conjunto de datos de entrada, cada una de las cuales posee propiedades o rasgos que la definen. Los rasgos presentes en cada conjunto de datos son llamados atributos.

El formato de archivos con el que trabaja Weka es denominado *arff* (acrónimo de Attribute-Relation File Format). Este formato está compuesto por una estructura claramente diferenciada en tres partes:

Sección de encabezamiento: se define el nombre del conjunto de datos.

Sección de declaración de atributos: se declaran los atributos a utilizar especificando su tipo. Los tipos aceptados por la herramienta son:

a) Numérico (Numeric*)*: expresa números reales

b) Entero (Integer): expresa números enteros

c) Fecha (Date): expresa fechas

d) Cadena (String): expresa cadenas de textos, con las restricciones del tipo String

e) Enumerado: expresa entre llaves y separados por comas los posibles valores (caracteres o cadenas de caracteres) que puede tomar el atributo.

Sección de datos: se declaran los datos que componen el conjunto de datos.

El formato por defecto de los ficheros que usa Weka es el *arff*, pero eso no significa que sea el único que admita. Esta herramienta tiene intérpretes de otros formatos como CSV, que son archivos separados por comas o tabuladores (la primera línea contiene los atributos) y C4.5 que son archivos codificados según el formato C4.5, donde los datos se agrupan de tal manera que en un fichero *.names* estarán los nombres de los atributos y en un fichero .data los datos en sí. Al leer Weka ficheros codificados según el formato C4.5 asume que ambos ficheros (el de definición de atributos y el de datos) están en el mismo directorio, por lo que sólo es necesario especificar uno de los dos. Además, las instancias pueden leerse también de un URL (Uniform Resource Locator) o de una base de datos en SQL usando JDBC.

Las clases de Weka están organizadas en paquetes (Witten and Frank 2005; Witten, Frank et al. 2011). Un paquete es la agrupación de clases e interfaces donde las clases que lo formen estén relacionadas y se ubiquen en un mismo directorio. Esta organización de la estructura de Weka hace que añadir, eliminar o modificar elementos no sea una tarea compleja. Los 10 paquetes globales que conforman Weka son:

associations: contiene las clases que implementan los algoritmos de asociación.

attributeSelection: contiene las clases que implementan técnicas de selección de atributos.

classifiers: agrupa todas las clases que implementan algoritmos de clasificación y estas a su vez se organizan en subpaquetes de acuerdo al tipo de clasificador.

clusterers: contiene las clases que implementan algoritmos de agrupamiento.

core: paquete central que contiene las clases controladoras del sistema. Es usado en la mayoría de las clases existentes. Las clases principales del paquete "core" son: Attribute, Instance, e Instances.

datagenerators: paquete que contiene clases útiles en la generación de conjuntos de datos atendiendo al tipo de algoritmo que será usado.

estimators: clases que realizan estimaciones (generalmente probabilísticas) sobre los datos.

experiment: contiene las clases controladoras que permiten la realización de experimentos con varias bases y diferentes algoritmos.

filters: está constituido por las clases que implementan algoritmos de preprocesamiento.

gui: contiene todas las clases que implementan la interfaz visual con el usuario.

Las clases principales del Weka se encuentran en el paquete core que como su nombre lo indica es el núcleo en el cual se sostiene toda la aplicación. Estas clases incluyen desde operaciones tan simples como sustituir una subcadena dentro de una cadena hasta realizar trabajos de manejo de la memoria.

2.4.2 Keel

KEEL es un software escrito en Java y de código abierto que incluye algoritmos evolutivos para resolver problemas de minería de datos incluyendo regresión, clasificación, agrupamiento y minería de patrones, entre otros. Contiene una gran colección de algoritmos clásicos de extracción del conocimiento, técnicas de preprocesamiento (selección del conjunto de entrenamiento, selección de rasgos, discretización y manipulación de valores ausentes), algoritmos de aprendizaje basados en

inteligencia computacional, algoritmos evolutivos de aprendizaje de reglas basados en diferentes enfoques, y modelos híbridos como sistemas genéticos borrosos y redes neuronales evolutivas, entre otros. KEEL permite realizar un análisis completo de cualquier modelo de aprendizaje en comparación con los ya existentes, incluyendo un módulo estadístico de prueba para la comparación.

Los componentes principales del Keel son:

Diseño de experimentos: esta parte tiene el objetivo de diseñar los experimentos deseados usando una interfaz gráfica. Después de que el experimento es diseñado, la interfaz genera un archivo .ZIP que contiene una estructura de directorios con todos los archivos necesarios para dirigir esos experimentos en la computadora local. La interfaz también da al usuario permiso de añadir sus algoritmos para la experimentación siendo diseñada. El único requisito es aceptar el formato del archivo de entrada del KEEL. En la Figura 2.1 se muestra la estructura de los experimentos en Keel.

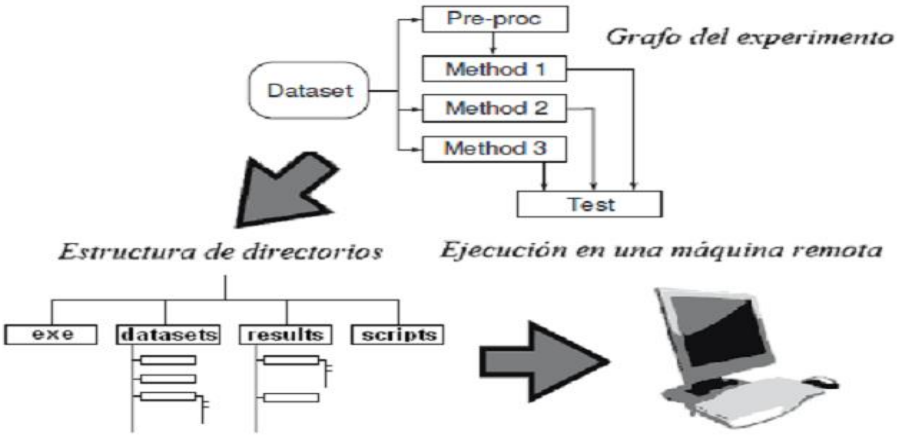

Figura 2.1 Estructura de los experimentos en Keel.

Existen muchas formas de configurar la experimentación que se va a realizar con los algoritmos de aprendizaje. En KEEL se nos proporcionan las siguientes opciones:

- El porcentaje de las particiones y el fichero global, generando con la herramienta los ficheros sobre los que se ejecutarán el algoritmo a partir del fichero global.
- Realizar un k-fold cross-validation (por defecto 10-fold cross validation).

Preparación de los Datos: Permite a los usuarios, crear particiones diferentes de sus bases de datos o las bases de datos disponibles dentro de la Web de KEEL. También, es posible editar, aplicar transformaciones, generar DataSets en el formato correcto de archivos UCI o ver gráficos detallados acerca de un conjunto de datos concreto.

Herramientas estadísticas: Un soporte muy importante en KEEL es la biblioteca de herramientas estadísticas que está desarrollada para comparar los resultados obtenidos por los algoritmos. Esta biblioteca está compuesta por:

- Test de Independencias: P-Pearson
- Test de Normalidad: Kolmogorov-Smirnov
- Test de Igualdad de Medias: ANOVA
- Test de Igualdad de Varianzas: Levene
- Comparación de dos Poblaciones: Test t-student, Test Wilcoxon, Test Binomial
- Comparación de más de 2 poblaciones: Ajuste de Bonferroni y Ajuste de Tamhane.

Extracción de Conocimiento: También contiene una Biblioteca de Algoritmos de Extracción de Conocimiento, supervisada y no supervisada, haciendo notar la incorporación de múltiples algoritmos de aprendizaje automatizado.

- Árboles de Decisión.
- Extracción de Reglas y Aprendizaje Supervisado.
- Extracción de Reglas e Inducción Descriptiva.
- Métodos Estadísticos de clasificación.
- Métodos Estadísticos para Regresión
- Otros Métodos Evolutivos para Clasificación
- Otros Métodos Evolutivos para Regresión
- Redes Neuronales
- Multiclasificadores - Combinación de Clasificadores
- Aprendizaje No Supervisado

El KEEL permite la aplicación de métodos de edición y de clasificación para conjuntos que tengan clases desbalanceadas. Esta es una ventaja del KEEL respecto a Weka y sugiere su uso en el problema que se aborda en este trabajo.

2.4.3 RapidMiner

RapidMiner[4], anteriormente conocido como Yet Another Learning Environment (YALE), es un software para el análisis y minería de datos. Permite el desarrollo de procesos de análisis de datos mediante el encadenamiento de operadores a través de un entorno gráfico. Se usa en investigaciones educativas, capacitación, creación rápida de prototipos y en aplicaciones empresariales. En una encuesta realizada por (KDnuggets), un periódico de minería de datos, RapidMiner ocupó el segundo lugar en herramientas de analítica y de minería de datos utilizadas para proyectos reales en 2009 y fue el primero en 2010.

La versión inicial fue desarrollada por el departamento de Inteligencia Artificial de la Universidad de Dortmund en 2001 y está hospedado en *SourceForge* desde el 2004.

RapidMiner proporciona más de 500 operadores orientados al análisis de datos, incluyendo los necesarios para realizar operaciones de entrada y salida, preprocesamiento de datos y visualización. También permite utilizar los algoritmos incluidos en Weka. Fue desarrollado en Java, permite la representación interna de los procesos de análisis de datos en ficheros XML y permite el desarrollo de programas a través de un lenguaje de script. Es extensible, incluye gráficos y herramientas de visualización de datos y dispone de un módulo de integración con R.

RapidMiner puede usarse de diversas maneras:

- A través de un GUI
- En línea de comandos
- En *batch* (lotes)
- Desde otros programas a través de llamadas a sus bibliotecas

2.4.4 Orange

Orange[5] es un programa para realizar minería de datos y análisis predictivo desarrollado en la facultad de informática de la Universidad de Ljubljana. Consta de una serie de componentes desarrollados en C++ que implementan algoritmos de minería de datos, así como operaciones de preprocesamiento y representación gráfica de datos.

[4] Se puede encontrar en http://sourceforge.net/projects/rapidminer/ y se distribuye bajo licencia AGPL
[5] Se puede encontrar en http://www.ailab.si/orange, se distribuye bajo licencia GPL

Los componentes de Orange pueden ser manipulados desde programas desarrollados en Python[6] o a través de un entorno gráfico. Los módulos de Orange son propuestos para extender su funcionalidad principal, o proporcionar capas para el uso más fácil de algunas técnicas de minería de datos.

Orange puede leer ficheros en formato native y en otros formatos. El formato notivo comienza con los nombres de los atributos, el tipo que tiene cada atributo (continuo, discreto, cadena). La tercera línea contiene la meta información para identificar los rasgos dependientes (clase), los rasgos irrelevantes (ignorar) o los meta rasgos (meta).

2.4.5 Knime

Knime (Konstanz Information Miner) es una plataforma de minería de datos que permite el desarrollo de modelos en un entorno visual. Fue desarrollado originalmente en el departamento de bioinformática y minería de datos de la Universidad de Constanza, Alemania, bajo la supervisión del profesor Michael Berthold. En la actualidad, la empresa KNIME.com GmbH, radicada en Zúrich, Suiza, continúa su desarrollo además de prestar servicios de formación y consultoría.

Knime está desarrollado sobre la plataforma Eclipse y programado, esencialmente, en java. Está concebido como una herramienta gráfica y dispone de una serie de nodos (que encapsulan distintos tipos de algoritmos) y flechas (que representan flujos de datos) que se despliegan y combinan de manera gráfica e interactiva.

Los nodos implementan distintos tipos de acciones que pueden ejecutarse sobre una tabla de datos:

- Manipulación de filas, columnas, etc., como muestreos, transformaciones, agrupaciones, predicciones, etc.
- Visualización (histogramas, etc.).
- Creación de modelos estadísticos y de minería de datos, como árboles de decisión, máquinas de vector soporte, regresiones, etc.
- Validación de modelos, como curvas ROC (receiver operating characteristic), etc.
- Scoring o aplicación de dichos modelos sobre conjuntos nuevos de datos.
- Creación de informes a la medida gracias a su integración con BIRT.

[6]Python Software Foundation *http://www.python.org/*

El carácter abierto de la herramienta hace posible su extensión mediante la creación de nuevos nodos que implementen algoritmos a la medida del usuario. Además, existe la posibilidad de utilizar la llamada directa y transparente a Weka, o de incorporar de manera sencilla código desarrollado en R o python/jython.

KNIME integra diversos componentes para aprendizaje automático y minería de datos a través de su concepto de fraccionamiento de datos (data pipelining) modular. La interfaz gráfica de usuario permite el montaje fácil y rápido de nodos para el preprocesamiento de datos (ETL: extracción, transformación, carga), para el análisis de datos, modelado y visualización. KNIME es desde 2006 utilizado en la investigación farmacéutica, pero también se utiliza en otras áreas, como: análisis de datos de cliente de CRM, inteligencia de negocio y análisis de datos financieros.

2.5 Conclusiones parciales

La hipertensión arterial (HTA), constituye una problemática de salud para cualquier nivel de atención sanitaria, independientemente del tipo de régimen o condición socioeconómica imperante. Actualmente, resulta de gran importancia estudiar la HTA en niños y adolescentes, además de reconocer que la HTA en la edad pediátrica es un problema médico que ha ido incrementándose con repercusiones negativas presentes y futuras muy poco alentadoras. Diversos estudios de corte longitudinal han reconocido el hecho de que las cifras de PA en la niñez y adolescencia influyen sobre los valores de la presión arterial en la adultez.

Existen pocos estudios a nivel mundial sobre la predicción de la evolución hacia la hipertensión arterial en la adultez desde la adolescencia. Afortunadamente, en Cuba existen algunas investigaciones al respecto. Uno de los principales resultados obtenidos en Cuba consistió en el diseño de dos modelos de predicción para pronosticar la conversión de un adolescente prehipertenso en un adulto hipertenso, mediante análisis multivariados que dan lugar a la definición de dos criterios diferentes para un índice integral de riesgo, utilizando V de Cramer y regresión. El otro estudio interesante determina modelos predictivos mediante regresión logística y árboles de decisiones utilizando la técnica de CHAID (chi square interaction detector). El mejor porciento de clasificación correcta obtenido fue 80%.

Contamos con los datos necesarios para seguir estudiando la predicción de la hipertensión en la adultez desde la adolescencia considerando una muestra de 125 adolescentes seleccionados por muestreo aleatorio simple con diagnóstico de preHTA. Se identificaron 82 adolescentes (65,60 %) que

se convirtieron en hipertensos establecidos en la adultez; mientras que el 34,40 % permaneció en la categoría de preHTA. Estan identificados los rasgos predictores a tener en cuenta en el estudio.

Existen herramientas que incluyen los principales algoritmos que permiten realizar predicciones; además, incluyen filtros que permiten aplicar técnicas de selección de rasgos, identificar asociaciones entre los rasgos, visualizar los resultados y aplicar medidas de validación de las clasificaciones realizadas. Se destacan Weka y Keel para el estudio del conjunto de casos. La primera por la gran gama de algoritmos de aprendizaje supervisado que ofrece, los filtros que permiten la ingeniería de los datos, las formas de validación que presenta y las facilidades de uso. La segunda, además, por incorporar métodos para la edición de datos en conjuntos con clases desbalanceadas y algoritmos de clasifcación para este tipo de conjuntos. Tanto el Weka como el Keel leen los ficheros en formato *.arff*.

3 Aplicación de técnicas de aprendizaje automático a la predicción desde la adolescencia de la hipertensión arterial

En este capítulo presentamos el diseño de los experimentos realizados para aplicar coherentemente las técnicas de aprendizaje automatizados en la predicción de la hipertensión arterial en la adultez a partir de los indicadores de los pacientes en la adolescencia. Se muestran los resultados de las principales técnicas aplicadas, tanto a los datos originales como después de normalizar, discretizar, seleccionar rasgos y editar. Atención especial tienen los resultados obtenidos cuando se aplican técnicas para el manejo de los conjuntos de datos con clases desbalanceadas.

3.1 Diseño de los experimentos

Existe una gran cantidad de técnicas para el preprocesamiento de los datos, así como algoritmos de clasificación. Por tal motivo, decidimos realizar experimentos comenzando por la aplicación de los métodos clásicos a la base de datos original, es decir, la variante más sencilla, hasta la aplicación de métodos de clasificación que logran manejar conjuntos con clases desbalanceadas, tal es el caso de la base de los 125 pacientes estudiados, ya que se conoce que 82 adolescentes (65,60 %) que se convirtieron en hipertensos establecidos en la adultez (clase mayoritaria); mientras que el 34,40 % permaneció en la categoría de preHTA (clase minoritaria).

A continuación describiremos cada uno de los experimentos realizados, aumentando el grado de complejidad de los métodos de clasificación a aplicar, así como las técnicas de preprocesamiento de los datos:

Experimento # 1:

1. Aplicar métodos clásicos de aprendizaje supervisado.
2. Aplicar medidas de validación de la clasificación

Experimento # 2:

1. Preprocesar los datos (selección de rasgos, normalización de atributos, discretización de atributos, edición)
2. Aplicar métodos clásicos de aprendizaje supervisado.

3. Aplicar medidas de validación de la clasificación.

Experimento # 3:

1. Aplicar métodos de clasificación para conjuntos con clases desbalanceadas.
2. Aplicar medidas de validación de la clasificación.

Experimento # 4:

1. Preprocesar los datos (selección de rasgos, normalización de atributos, discretización de atributos, edición)
2. Aplicar métodos de clasificación para conjuntos con clases desbalanceadas.
3. Aplicar medidas de validación de la clasificación.

Experimento # 5:

1. Aplicar métodos de edición para conjuntos de datos con clases desbalanceadas. En algunos casos preprocesar los datos además con técnicas de selección de rasgos, normalización de atributos y discretización de atributos, edición.
2. Aplicar métodos clásicos de aprendizaje supervisado.
3. Aplicar medidas de validación de la clasificación.

Los experimentos se realizaron utilizando las herramientos Weka y Keel. Para ellos se preparó el fichero con formato *.arff* con los datos obtenidos por los estudios realizados anteriormente, icnluyendo los 125 pacientes de la muestra. El fichero creado se nombró “Prehta.arff”.

3.2 Experimento # 1

Los resultados experimentales que se presentan en la Tabla A.1 del Anexo 1 muestran el desempeño de estos clasificadores que aparecen en Weka aplicados al conjunto de datos original. Los clasificadores están divididos en grupos según su implementación en Weka, de los métodos basados en redes bayesianas utilizamos el (BayesNet) y el (NaiveBayes), de las resdes neuronales usamos el (Multilayer Perceptron, MLP) que en la opción *hiddenLayers* se determina el número de neuronas ocultas, sus posibles valores son: a=(atributos+clases)/2, i=atributos, o=clases, t=atributos+clases. De los algoritmos basados en instancias usamos el (IBK), que tiene en cuenta únicamente el voto del vecino más cercano, es por eso que se especifica el valor de k, y el KStar, que si se activa la opción (entropicAutoBlend) se calcula el valor de los parámetros x_0 o (S) basándose en la entropía y sino se

calculan los parámetros de mezclado, expresados en tanto porciento; de los metaclasificadores usamos el (Attribute Select Classifier) que se especifica el algoritmo que utiliza con los atributos reducidos, también utilizamos el (*Bagging*) y debemos especificar el algoritmo que se combina para clasificar, al igual que con los restantes multiclasificadores como el (*Random Commitee*) y el (*Random Subspace*); de los métodos basados en reglas tomamos los que den mejores resultados, que mayormente es el JRip o el OneR, y de los árboles de decisión utilizamos el J48 (C4.5) y el (RandomForest), que frecuentemente tiene mejores resultados que sus semejantes. Ademas, aplicamos las máquinas de soporte vectorial implementadas en Keel sobre el conjunto de los datos originales. En la Tabla A.1 del Anexo 1 mostramos los resultados de los algoritmos que obtuvieron los mejores valores de clasificación por cada tipo de método, con los valores de los parámetros que permitieron los mejores resultados. Tanto en la Tabla A.1 del Anexo 1 como Figura 3.1 se observa que el mejor resultado obtenido es un 76,8% de casos correctamente clasificados, aplicando el metaclasificador *Random Committee (RandomForest)*. Este resultado está por debajo del resultado obtenido en investigaciones precedentes a esta.

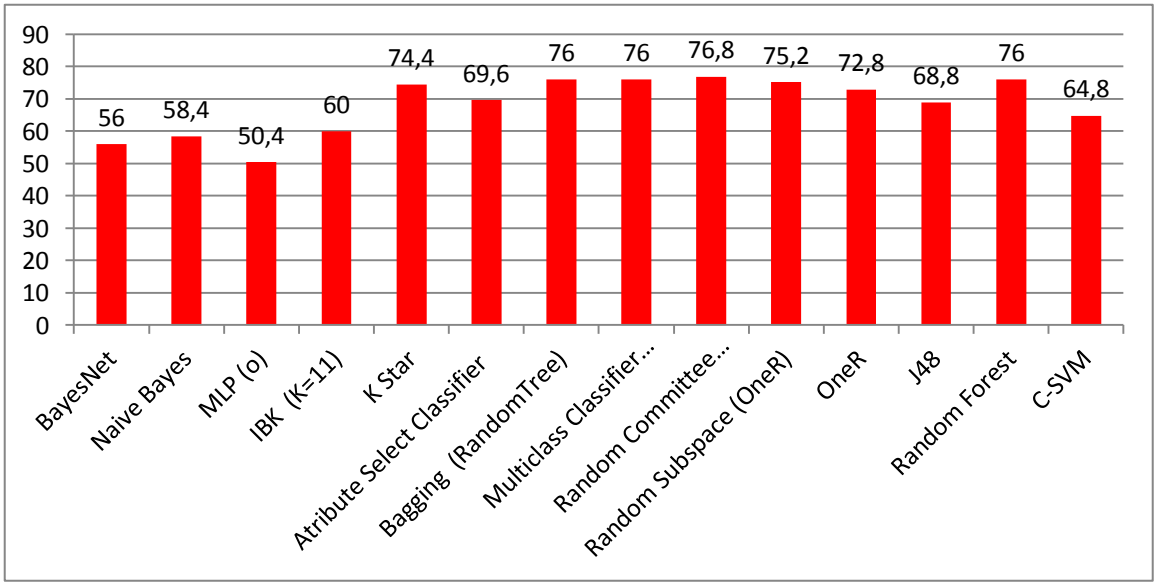

Figura 3.1 Resultados de algoritmos de clasificación aplicados al conjunto de datos sin preprocesar.

3.3 Experimento # 2

Ya analizamos que los resultados obtenidos en el Experimento # 2 no son buenos, por tanto se hace necesario aplicarle algún filtro que modifique los datos de forma tal que los clasificadores superen

estos valores, con filtros supervisados de selección de atributos o no supervisados, como la normalización y discretización que veremos a continuación.

Para preprocesar los datos, primeramente se aplica el filtro no supervisado para normalizar los atributos, de forma que se obtienen los siguientes resultados, con los de mejor desempeño en cada caso. Como se puede ver en la Tabla A.2 del Anexo 2, en cuanto a los resultados obtenidos en (Pérez Fernández 2012), no se mejoran los resultados, incluso son peores que al trabajar con los datos originales. Por tanto, trataremos de probar dicretizando los atributos continuos, aplicando un filtro no supervisado a los atributos originales. Los resultados se muestran en la Tabla A.3 del Anexo 2, donde se observa que el metaclasificador *Atribute Select Classifier (IBk)* obtuvo el 83,2% de instancias bien clasificadas. Ya en este experimentos se obtienen resultados que superan las investigaciones precedentes, pero aun pudieran ser superados.

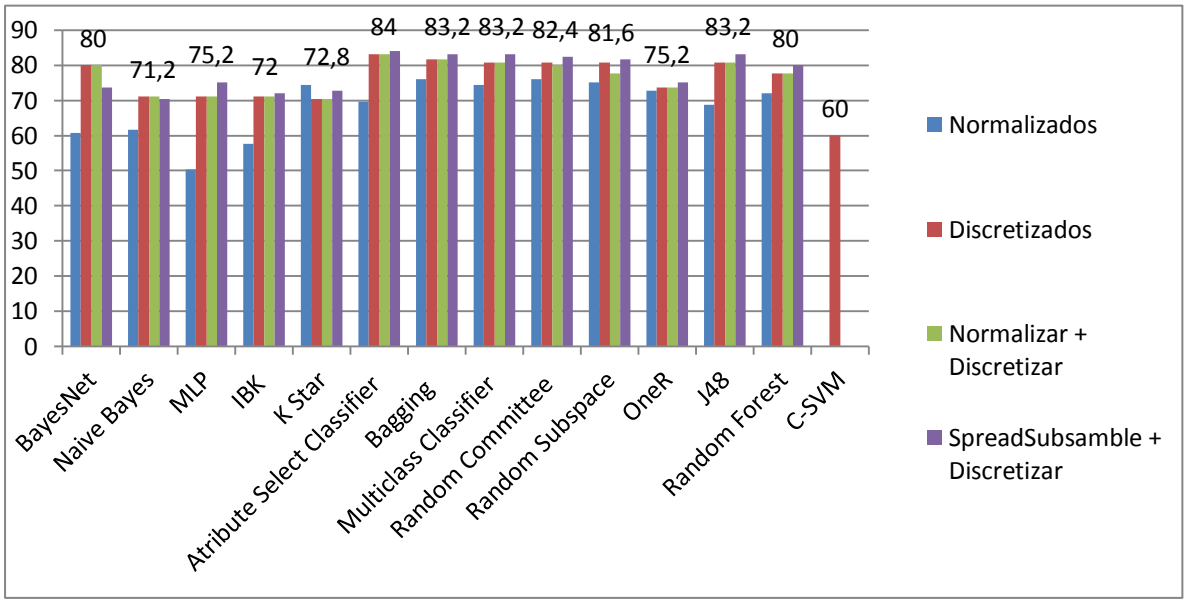

Figura 3.2 Resultados de algoritmos de clasificación aplicados al conjunto de datos preprocesar.

Ahora trataremos de probar normalizando y luego discretizando los atributos para ver si se logran obtener mejores resultados al aplicar estos clasificadores. Desafortunadamente, como se observa en la Tabla A.4 del Anexo 2, se mantiene el mismo porciento de clasificación correcta cuando solo se discretizaron los datos. El clasificador que produjo este resultado es el IBK con K=5. IBK obtuvo 104 instancias correctamente clasificadas que representa un 83.2%, un error medio absoluto de 0.2272, esto se puede seguir modificando con un filtro supervisado de instancias como el *Spread Subsample*

que permite editar el conjunto de entrenamiento. Los resultados al aplicar este filtro se muestran en la Tabla A.5 del Anexo 2. Aplicando el filtro supervisado por instancias *Spread Subsample* se mejoran ligeramente los resultados, obteniendo un 84% de instancias clasificadas correctamente al usar el multiclasificador que selecciona atributos usando el algoritmo *Random Forest*.

En la Figura 3.2 se muestran los porcientos de clasificación correcta al aplicar los clasificadores clásicos a los datos preprocesados.

3.4 Experimento # 3

El KEEL en la sección de Módulos contiene una serie de algoritmos de clasificación para clases desbalanceadas, que cuando se aplicaron a los datos originales, que presentan este tipo de problemas, se obtuvieron los resultados que se muestran en la Tabla A.6 del Anexo 3.

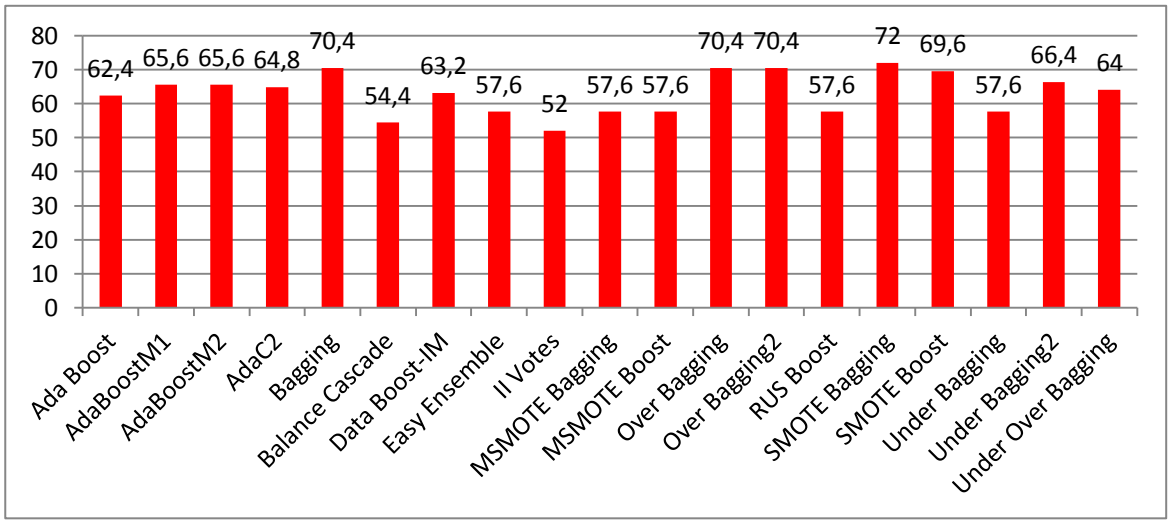

Figura 3.3 Resultados de algoritmos de clasificación que manejan datos con clases desbalanceadas aplicados al conjunto de datos original.

En la Tabla A.6 se muestran los resultados divididos por pruebas (test) y entrenamiento (training), para cada clasificador y los resultados se miden en cuanto a la exactitud de clasificación, desviación estándar, error esperado, error medio y las dos últimas columnas de la tabla muestran las matrices de confusión (de prueba y de entrenamiento), que se dividen en los valores que toma la matriz 2x2 y los valores de las clases, el valor 2 son los pacientes saludables y el valor 1 son los pacientes que llegaron a ser hipertensos en la adultez, y como se puede apreciar los resultados no son muy buenos, ya que

no superan el 0.8 de exactitud en las particiones de pruebas y los valores de error son mayores que 0.2 que representa un 20%. En la Figura 3.3 se muestran los porcientos de clasificación correcta por algoritmos que manejan datos con clases desbalanceadas.

3.5 Experimento # 4

Al obtener malos resultados en el Experimento # 3 con los datos sin preprocesar, decidimos en este experimento aplicar un filtro para discretizar los datos, buscando un buen desempeño de estos algoritmos. Los resultados se muestran en la Tabla A.7 del Anexo 4, donde es posible apreciar que al aplicar los algoritmos para trabajar con clases desbalanceadas después de discretizar los datos con un filtro de discretización de igual frecuencia, los resultados no mejoraron su desempeño, ya que el de mayor exactitud correcta fue el *bagging* con un 75,2% de clasificación correcta considerando el conjunto de prueba.

En la Figura 3.4 se muestran los porcientos de clasificación correcta por algoritmos que manejan datos con clases desbalanceadas aplicados al conjunto de datos discretizado.

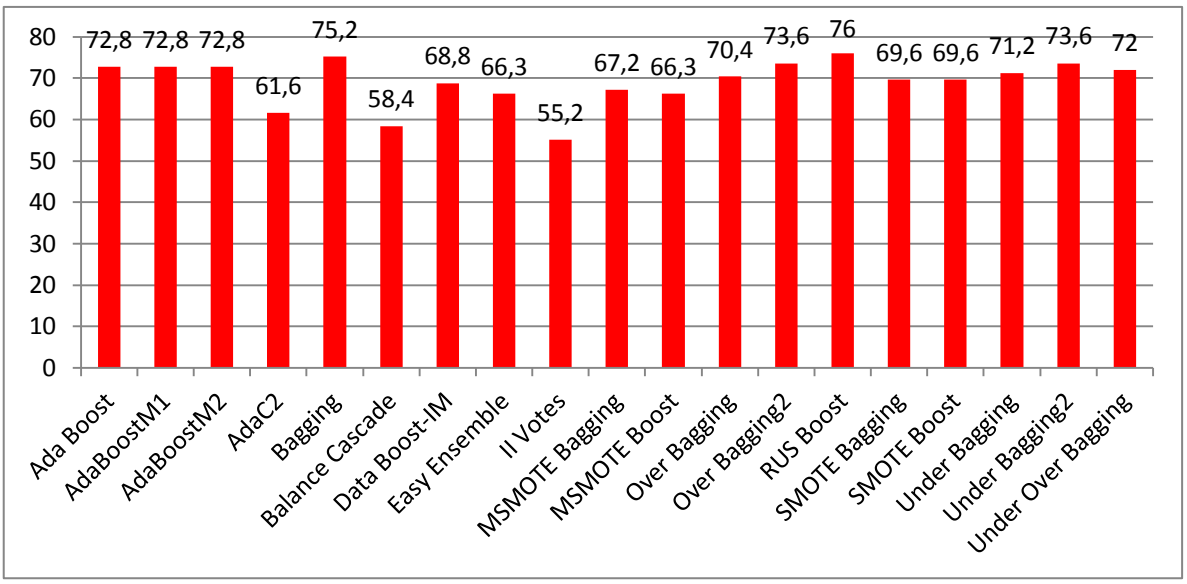

Figura 3.4 Resultados de algoritmos de clasificación que manejan datos con clases desbalanceadas aplicados al conjunto de datos discretizado.

3.6 Experimento # 5

Para resolver el problema del conjunto de datos se debe tratar de balancear las clases y para eso se utilizan algunos métodos de edición que presenta el KEEL como el Borderline_SMOTE, SPIDER2, Safe_Level_SMOTE, SMOTE_ENN, SMOTE_RSB, SMOTE_TL y el SMOTE, que eliminan e incrementan instancias a las clases, para equilibrar ambas clases. A continuación usamos estos métodos de edición y seleccionamos rasgos relevantes para evaluar y buscar mejores resultados. Los métodos de edición se aplicaron en el KEEL y los métodos de clasificación aplicados fueron los del Weka. Primeramente se aplicaron los clasificadores sin preprocesar los datos obtenidos por los métodos de edición del KEEL, y luego se le aplicaron filtros de selección de rasgos y discretización, los conjuntos de datos editados que mejores resultados dieron.

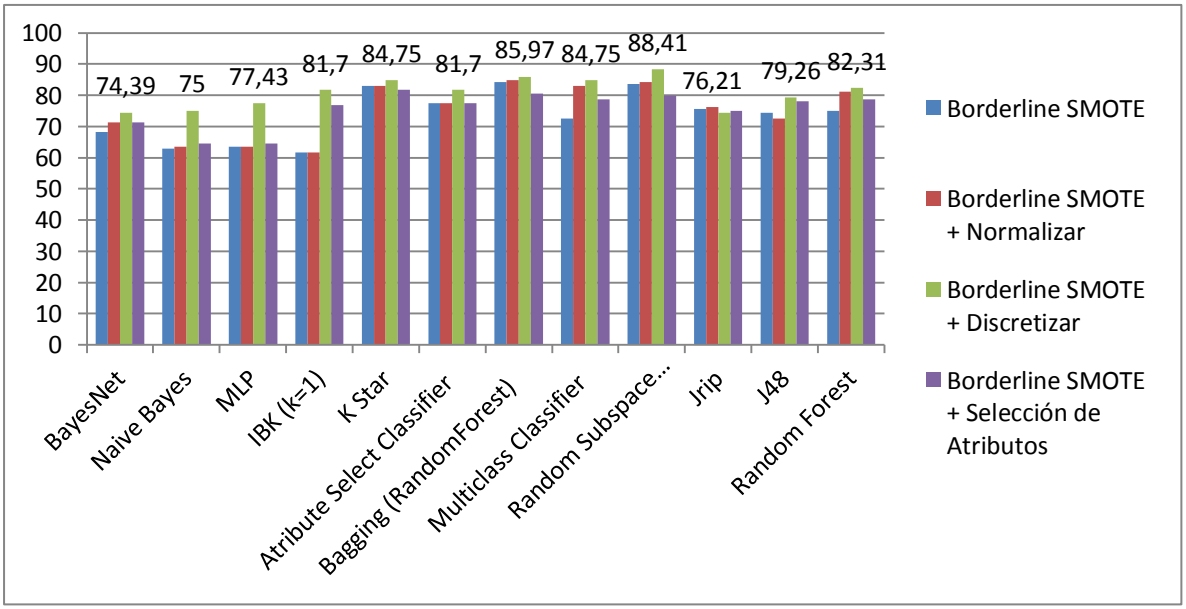

Figura 3.5 Resultados de algoritmos de clasificación después de editar con Borderline SMOTE.

La Tabla A.8 del Anexo 5 y en la Figura 3.5 se muestran los resultados obtenidos por el método de edición para clases desbalanceadas Borderline_SMOTE del KEEL, incrementando la cantidad de instancias para equilibrar las clases, que mejora los resultados, ya que se obtuvo con el multiclasificador Bagging y el árbol de decisión Random_Forest, un 84.14% de instancias bien clasificadas, con el multiclasificador Random_Subspace y el árbol de decisiones Random_Forest se obtiene un 83.53% de clasificación correcta y 0.909 de área bajo la curva ROC y con el método basado en instancias (K*) se obtiene un error medio absoluto de 0.1908, y un TPRate de 0.939 para

una clase y una precisión de 0.922 para la otra, con un área bajo la curva ROC de 0.928 muy cercano a 1, con el clasificador de redes bayesianas Naive Bayes se obtiene un TPRate de 0.927, siendo estos los mejores resultado obtenidos hasta el momento, a continuación se muestran los resultados obtenidos aplicando otros tipos de pre-procesamiento de los datos para balancear las clases.

En la Tabla A.9 del Anexo 5 se muestran los resultados de aplicar el método *Borderline_SMOTE1* al conjunto de datos normalizados y se obtienen resultados similares al aplicarlo a los datos originales. En la Tabla A.10 se muestran los resultados de aplicar el método *Borderline_SMOTE1* para balancear las clases y luego discretizando, con 164 instancias y 17 atributos. El mejor resultado lo obtuvo el multiclasificador Random Subspace usando el algoritmo Random Forest, con un porciento de instancias correctamente clasificadas de 88.41%, siendo este el mejor hasta el momento de los casos anteriores, también se puede señalar que otros algoritmos de clasificación mejoraron sus resultados, como los perezosos que obtuvieron un Error Medio Absoluto de 0.18 y un área bajo la curva ROC de 0.9 en el caso del K Star, pero si aplicamos otros filtros podríamos mejorar estos resultados.

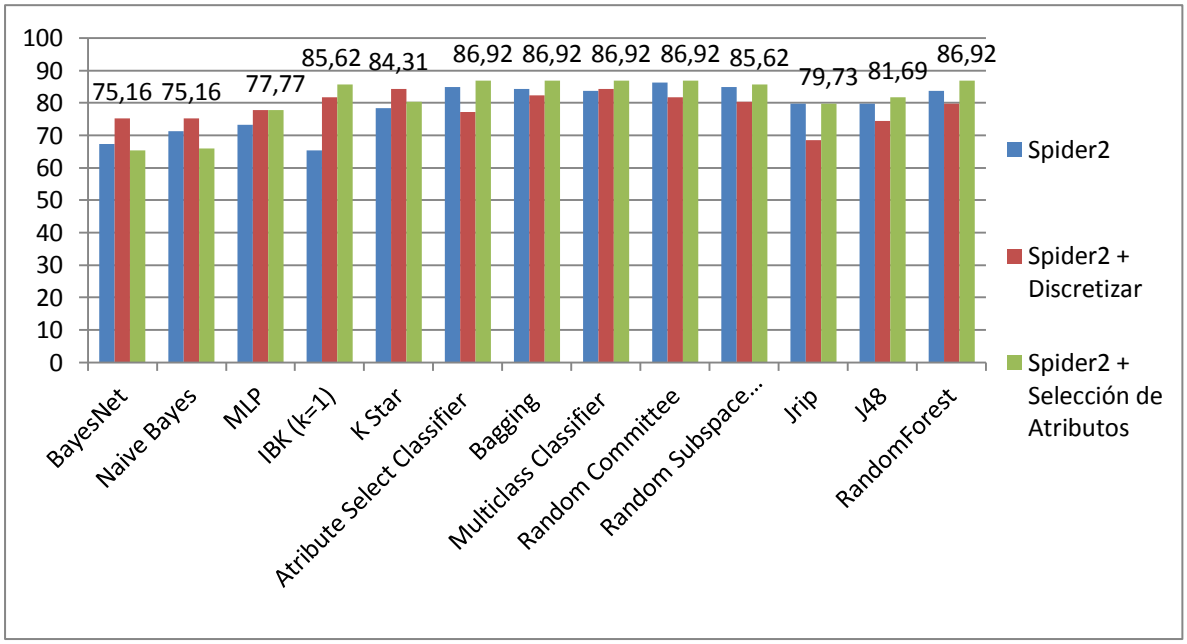

Figura 3.6 Resultados de algoritmos de clasificación después de editar con Spider2.

En las tablas de la A.10 a la A.26 del Anexo 5 se muestran los resultados de aplicar los restantes métodos de edición para clases desbalanceadas al conjunto de datos original y después aplicando

filtros como: discretizadores, normalizadores y selección de rasgos. Desafortunadamente, ninguno de estos experimentos logró superar los resultados alcanzados por el multiclasificador Random Subspace usando el algoritmo Random Forest al editar el conjunto de datos con el método *Borderline_SMOTE1.*

Los resultados obtenidos de los datos editados por *SPIDER2-I*, con los mismos algoritmos de clasificación, sin aplicar ningún otro filtro y con 164 instancias en total, fueron muy buenos y el mejor resultado se encontró al aplicar el multiclasificador Random Committee utilizando el árbol de decisión Random Tree que clasificó correctamente 132 instancias representando un 86.27%, y como se puede apreciar en otros clasificadores se obtuvieron buenos resultados atendiendo al TPRate con valores mayores que 0.9, por clases y los niveles de precisión también están por encima de 0.8, o sea, que con esta modificación a la base de casos se logra una mejor predicción de la hipertensión arterial a partir de adolescentes pre-hipertensos.

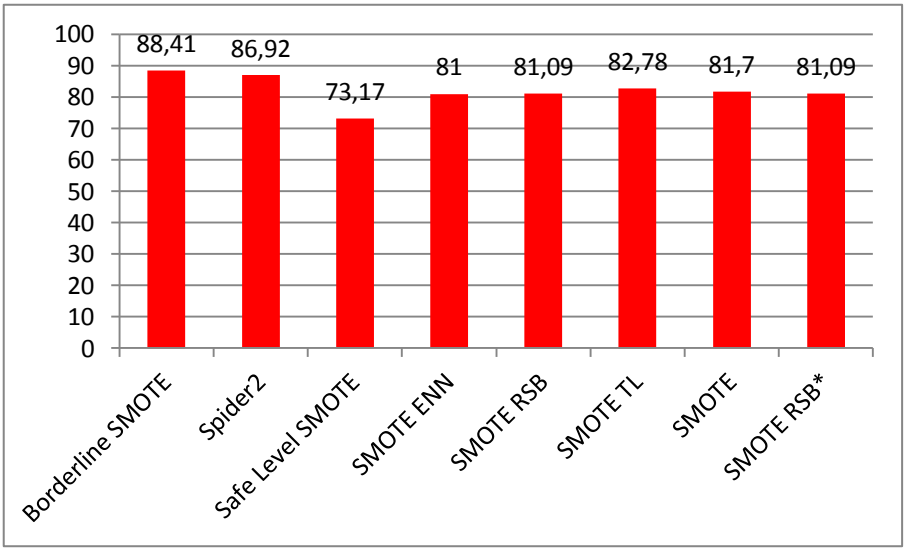

Figura 3.7 Los mejores resultados de algoritmos de clasificación después de aplicar métodos de edición para problemas con clases desbalanceadas.

De manera general, al aplicar la selección de rasgos después de balancear las clases, los clasificadores no logran mejorar los resultados, incluso algunas veces empeoran, aunque se mantienen algunos sobre el 80% de instancias correctas, como es el caso del algoritmo perezoso *K Star* y el multiclasificador *Bagging* usando el árbol *Random Forest*. Este filtro aplicado no es conveniente, ya que elimina atributos que son importantes al predecir la hipertensión arterial en la adultez, y como se

puede apreciar quedan muy pocos atributos, lo que provoca una pérdida de información necesaria. Es necesario señalar que la selección de rasgos si resultó acertada al aplicarla después de editar el conjunto de datos con el método *SPIDER2.* En este caso si mejoraron considerablemente los valores de algunos algoritmos de clasificación superando incluso el 85% de instancias correctamente clasificadas y el TPRate también se muestra bastante alto con valores superiores a los 0.9 por clases y el error medio absoluto alcanza los 0.14.

En la Figura 3.7 se muestran los mejores resultados de algoritmos de clasificación después de aplicar métodos de edición para problemas con clases desbalanceadas, evidenciándose que el mejor resultado se obtiene al editar el conjunto de datos con *Borderline SMOTE*.

3.7 Conclusiones parciales

Se diseñaron cinco experimentos para aplicar las técnicas de aprendizaje automatizado supervisado al conjunto de datos de estudio con los 125 pacientes. Los mejores resultados fueron obtenidos al realizar el Experimento # 5, cuando se aplicaron métodos de edición que son capaces de manejar problemas con clases desbalanceadas (estos métodos aparecen implementados en el Keel), técnicas para discretizar los datos y después se aplicaron los algoritmos clásicos que incorpora el Weka para problemas de clasificación supervisada. De todos los métodos de edición del Keel con tales propósitos, fue el *Borderline_SMOTE1* el que logró transformar la base de 125 casos a 164, garantizando de esta forma balancear las clases. Posteriormente se discretizaron los atributos continuos y el mejor resultado lo obtuvo el multiclasificador Random Subspace usando el algoritmo Random Forest, con un porciento de instancias correctamente clasificadas de 88.41%.

Conclusiones y recomendaciones

Como resultado de la investigación, se aplicaron las técnicas de aprendizaje automatizado que permitieron pronosticar la conversión de un adolescente prehipertenso en un adulto hipertenso en Cuba, elevando los índices de exactitud previamente obtenidos; cumpliéndose de esta forma el objetivo general planteado ya que:

- Se identificaron los algoritmos de clasificación clásicos basados en redes bayesianas, redes neuronales, basados en instancias, basados en reglas, árboles de decisión, máquinas de soporte vectorial y metaclasificadores que combinan estos métodos; así como las herramientas Keel y Weka que los implementan y que permitieron su efectiva aplicación en la predicción de adultos hipertensos a partir de adolescentes prehipertensos.
- Se comprobó que es imprescindible realizar una correcta ingeniería de los datos, donde algunos métodos aplicados fueron la normalización, discretización, selección de rasgos y edición del conjunto de entrenamiento, sobre todo aquellos métodos que son capaces de manipular conjuntos de datos con clases desbalanceadas, lográndose con esto mejores resultados en la clasificación que al trabajar directamente con los datos originales.
- Los mejores resultados fueron obtenidos al realizar el Experimento # 5, cuando se aplicaron métodos de edición que son capaces de manejar problemas con clases desbalanceadas (estos métodos aparecen implementados en el Keel), técnicas para discretizar los datos y después se aplicaron los algoritmos clásicos que incorpora el Weka para problemas de clasificación supervisada. De todos los métodos de edición del Keel con tales propósitos, fue el *Borderline_SMOTE1* el que logró transformar la base de 125 casos a 164, garantizando de esta forma balancear las clases. Posteriormente se discretizaron los atributos continuos y el mejor resultado lo obtuvo el multiclasificador Random Subspace usando el algoritmo Random Forest, con un porciento de instancias correctamente clasificadas de 88.41%.

Derivadas del estudio realizado, así como de las conclusiones generales emanadas del mismo, se recomienda:

- Aplicar los resultados presentados en (González Castellanos 2010) para modelar los atributos antecedente familiar de HTA, antecedente familiar de obesidad, antecedente familiar de

diabetes mellitus, antecedente familiar de cardiopatía isquémica, antecedente familiar de insuficiencia cardíaca y antecedente familiar de accidente vascular encefálico, como un único atributo tipo conjunto que reúna todos los posibles antecedentes que pueda presentar el paciente. De esta forma se podrá experimentar y ver las ventajas o no de la modelación de los datos con el atributo tipo conjunto, a partir del análisis de la eficacia en la clasificación utilizando árboles de decisión, enfoque basado en instancias y el probabilístico.

- Incrementar el conjunto de datos de estudio incluyéndole nuevos pacientes estudiados por especialistas de cardiología en el Hospital Universitario "Celestino Hernández Robau" de Santa Clara, Villa Clara.
- Valorar la posibilidad de coleccionar los datos de los pacientes considerando mayor información de cada rasgo y no sólo utilizando variables dicotómicas como está ahora en la mayoría de los rasgos. De esta forma se deben obtener mejores resultados de los clasificadores y por tanto superar aún más los indicadores de clasificación correcta.

Referencias bibliográficas

Aha, D. (1992). "Tolerating noisy, irrelevant and novel attributes in instancebased learning algorithms." Computational Journal of Man-Machine Studies **36**: 267-287.

Aha, D. and D. Kibler (1991). "InstanceBased Learning Algorithms.".

Arboláez Malboa, A. (2008). "Extensiones al ambiente de aprendizaje automatizado Weka-parallel con distintos algoritmos de aprendizaje en redes bayesianas. Aplicaciones Bioinformáticas." Trabajo de diploma en Ciencia de la Computación. Universidad Central "Marta Abreu" de Las Villas.

Barandela, R. (2003). "Strategies for learning in class imbalance problems." Pattern Recognition. **36**(3): 849-851.

Barandela, R. and E. Gasca (2002). "Correcting the Training Data." Pattern Recognition and String Matching.

Batista, G. E. A. P. A. (2004). "A study of the behaviour of several methods for balancing machine learning training data." SIGKDD Explorations . **6**(1): 20-29.

Berenson, G. S. (2002). "Childhood risk factors predict adult risk associated with subclinical cardiovascular disease: the Bogalusa Heart Study. ." Am J Cardiol. **90**: 3L–7L.

Bezdek, J. C. and L. I. Kuncheva (2004). "Nearest Prototype classifiers design: an experimental study.".

Bonet Cruz, I. (2009). "Modelo para la clasificación de secuencias, en problemas de la Bioinformática, usando técnicas de Inteligencia Artificial." Tesis de doctorado en Ciencias Técnicas. Universidad Central "Marta Abreu" de Las Villas.

Bonet, I. (2008). Modelo para la clasificación de secuencias, en problemas de la bioinformática, usando técnicas de inteligencia artificial. . Bioinformática. Villa Clara, Universidad Central "Martha Abreu" de las Villas.

Bradley, A. P. (1997). "The use of the area under the ROC curve in the evaluation of machine learning algorithms." Pattern Recognition **30**(7): 1145-1159.

Breiman, L. (1996). "Bagging predictors." Machine Learning **24**: 123-140.

Breiman, L. (2001). "Random Forests." Machine Learning **45**: 5–32.

Bunkhumpornpat, C. (2009). "Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem." Pacific-Asia Conference on Knowledge. Discovery and Data Mining (PAKDD09), Springer.

Caballero Mota, Y. (2008). "Aplicación de la Teoría de los conjuntos aproximados en el preprocesamiento de los conjuntos de entrenamiento para los algoritmos de aprendizaje automatizado. ." Tesis de doctorado en Ciencias Técnicas. Universidad Central "Marta Abreu" de Las Villas.

Catlett, J. (1991). "On changing continuous attributes into ordered discrete attributes." In Y. Kodratoff, ed., Proceedings of the European Working Session on Learning, Berlin, Germany. Springer-Verlag.

Chávez Cárdenas, M. C. (2009). "Modelos de redes bayesianas en el estudio de secuencias genómicas y otros problemas biomédicos." Tesis de doctorado en Ciencias Técnicas. Universidad Central "Marta Abreu" de Las Villas.

Chawla, N. V. (2002). "SMOTE: Synthetic minority over-sampling technique." Journal of Artificial Intelligent Research. **16**: 321-357.

Chawla, N. V. (2004). "Editorial: special issue on learning from imbalanced data sets." SIGKDD Explorations **6**(1): 1-6.

Dougherty, J., R. Kohavi, et al. (1995). "Supervised and unsupervised iscretization of continuous features." In Machine Learning: Proceedings of the twelfth International Conference. Morgan kaufmann.

Estabrooks, A. (2004). A multiple resampling method for learning from imbalanced data sets. Computational Intelligence. **20:** 18-36.

Fayyad, U. M., G. Piatetsky-Shapiro, et al. (1996). Advances in knowledge discovery and data mining.

Fernández, A. (2010). "On the 2-tuples based genetic tuning performance for fuzzy rule based classification systems in imbalanced data-sets.". Information Sciences **180:** 1268-1291.

Filiberto Cabrera, Y. (2011). "Métodos de aprendizaje para dominios con datos mezclados basados en la teoría de los conjuntos aproximados extendida. ." Tesis de doctorado en Ciencias Técnicas. Universidad Central "Marta Abreu" de Las Villas.

Freund, Y. and R. E. Schapire (1997). "Decision-theoretic generalization of on-line learning and an application to boosting." Journal of Computer and System Sciences **55**: 119–139.

Freund, Y. and R. E. Schapire (1997). "Decision-theoretic generalization of on-line learning and an application to boosting." Journal of Computer and System Sciences **55**: 119–139.

García, M. and Y. Villuendas (2005). "Métodos de selección y construcción de objetos para el mejoramiento de un clasificador supervisado: estado del arte.".

Gómez Díaz, Y. (2010). "Algoritmos que combinan conjuntos aproximados y optimización basada en colonias de hormigas para la selección de rasgos. Extensión a múltiples fuentes de datos. ." Tesis de doctorado en Ciencias Técnicas. Universidad Central "Marta Abreu" de Las Villas.

González Castellanos, M. (2010). "Extensión de algoritmos representativos del aprendizaje automático al trabajo con datos tipo conjunto. ." Tesis de Maestría en Ciencia de la Computación. Universidad Central "Marta Abreu" de Las Villas.

González Castellanos, M. (2010). "Extensión de algoritmos representativos del aprendizaje automático al trabajo con datos tipo conjunto." Tesis de Maestría en Ciencia de la Computación. Universidad Central "Marta Abreu" de Las Villas.

Guevara Yanes, L. E. (2007). "Extensión del sistema Weka con la incorporación de Redes Neuronales Recurrentes." Trabajo de diploma en Ciencia de la Computación. Universidad Central "Marta Abreu" de Las Villas.

Hamamoto, Y. and S. Uchimura (1997). "A bootstrap technique for nearest neighbor classifier design." IEEE transactions on pattern analysis and Machine intelligence **19**(1): 73-79.

Han, H. (2009). "Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning." International Conference on Intelligent Computing (ICIC05) LNCS.

Ho, T. K. (1998). "The random space method for constructing decision forests." IEEE Transactions on Pattern Analysis and Machine Intelligence **20**: 832-844.

Huang, J. and C. X. Ling (2005). "Using AUC and accuracy in evaluating learning algorithms." IEEE Transactions on Knowledge and Data Engineering **17**(3): 299-310.

Huang, Y. M. (2006). "Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem." Nonlinear Analysis: Real World Applications **7**(4): 720-747.

Kibler, D. and D. Aha (1987). "Learning representative exemplars of concepts: An initial case study." Fourth International Workshop on Machine Learning, Irvine, CA.

Kilic, K. (2007). "Comparison of different strategies of utilizing fuzzy clustering in structure identification." Information Sciences **177**(23): 5153-5162.

Koplowitz, J. and T. A. Brown (1978). "On the relation of performance to editing in nearest neighbor rule." 4th International Joint Conference on Pattern Recognition, Japan.

Kuncheva, L. I. (2004). Combining Pattern Classifiers, Methods and Algorithms. New York, NY, Wiley Interscience.

Larrañaga, P., B. Calvo, et al. (2006). "Machine learning in bioinformatics." Briefings in Bioinformatics **7**(1): 86-112.

Lurbe, E. T., M.I. (2010). "Early vascular phenotypes in the genesis of hypertension." Pediatr Nephrol. **25(4)**: 763-767.

Mancia, G. D. B., G.; Dominiczak, A.; Cifkova, R.; Fagard, R.; Germano, G. (2007). "ESH-ESC Guidelines for the management of arterial hypertension: the task force for the management of arterial hypertension of the European Society of Hypertension (ESH) and of the European Society of Cardiology (ESC)." J Hypertens. **25(6)**: 1105-1187.

Mazurowski, M., et al. (2008). "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance." Neural Networks **21**(2-3): 427-436.

McNiece, K. L. P., T.S.; Turner, J.L.; Franco, K.D.; Sorof, J.M.; Portman, R.J. (2007). "Prevalence of hypertension and prehypertension among adolescents." J Pediatr. **150**: 640-644.

Mitchel, T. M. (1997). Machine learning.

Orriols-Puig, A. and E. Bernad´o-Mansilla (2009). "Evolutionary rule–based systems for imbalanced datasets." Soft Computing **13**(1): 21.

Peng, X., et al. (2008). "Robust BMPM training based on second-order cone programming and its application in medical diagnosis." Neural Networks **21**(2-3): 450-457.

Pérez Fernández, G. A. (2011). "ESTUDIOS CLÍNICO-EPIDEMIOLÓGICOS DE LA PRESIÓN ARTERIAL SISTÉMICA EN ADOLESCENTES DEL MUNICIPIO SANTA CLARA, 2001-2009." Tesis presentada en opción al Grado Científico de

Doctor en Ciencias Médicas. UNIVERSIDAD DE CIENCIAS MÉDICAS DE VILLA CLARA.

Pérez Fernández, G. A. G. A., R. (2012). "Del adolescente prehipertenso al adulto hipertenso. ¿Es posible pronosticar la conversión?" Archivos de cardiología de México. Investigación clínica. Elsevier. **82(2)**: 112-119.

Pérez Fernández, G. A. G. A., R. (2012). "Predicción de la evolución hacia la hipertensión arterial en la adultez desde la adolescencia." Revista Cubana de Informática Médica. Número 1 **Año 12**.

Pérez Fernández, G. A. M. E., Y.; Aladro, I.; Santana Santana, C. (2005). "Afectación estructural cardiovascular precoz detectada mediante ecocardiografía bidimensional y doppler en el adolescente hipertenso." Revista Mapfre Medicina. **16**: 159-166.

Polikar, R. (2006). Ensemble based systems in decision making. IEEE Circuits and Systems Magazine**:** 21-44.

Provost, F., et al. (2001). Robust classification for imprecise environments. Machine Learning Research **42:** 203-231.

Ramentol, E., R. Bello, et al. (2010). "SMOTE-RSB*: A Hybrid Preprocessing Approach based on Oversampling and Undersampling for High Imbalanced Data-Sets using SMOTE and Rough Sets Theory." Tesis de doctorado Iberoamericano en Soft Computing.

Rodríguez Sarabia, Y. (2009). "Generalización de la métrica basada en la diferencia de valores (VDM) para variables lingüísticas y su aplicación en sistemas basados en el conocimiento. ." Tesis de doctorado en Ciencias Técnicas. Universidad Central "Marta Abreu" de Las Villas.

Saeys, Y., I. Inza, et al. (2007). "A review of feature selection techniques in bioinformatics." Bioinformatics **23**(19): 2507–2517.

Schapire, R. E. (1990). "The strength of weak learnability." Machine Learning **5**(2): 197-227.

Skurichina, M. (2001). Stabilizing weak classifiers. Delft, The Netherlands., Delft University of Technology. **PhD**.

Sun, Y., et al. (2007). "Cost-sensitive boosting for classification of imbalanced data." Pattern Recognition **40**: 3358-3378.

Suresh, S. e. a. (2008). "Risk-sensitive loss functions for sparse multi-category classification problems." Information Sciences **178**(12): 2621-2638.

Tomek, I. (1976). "An Experiment with the edited nearest neighbor rule." IEEE Transactions on Systems, Man and Cybernetics **6**(6): 448-452.

Vapnik, V. (1995). The nature of statistical learning theory. New York, Springer-Verlag.

Visa, S. and A. Ralescu (2005). "Issues in Mining Imbalanced Data Sets - A Review Paper." ECECS Department ML 0030 University of Cincinnati.

Wilson, D. L. (1972). "Asymptotic properties of Nearest Neighbor rules using edited data sets." IEEE Transactions on Systems, Man and Cybernetics, SMC **2**: 408-421.

Witten, I. and E. Frank (2005). Data Mining: Practical Machine Learning Tools and Techniques. San Francisco, Diane Cerra.

Witten, I. H. and E. Frank (2005). Data Mining, practical Machine Learning Tools and Techniques. USA, San Diego, Morgan Kaufmann Publishers.

Witten, I. H., E. Frank, et al. (2011). Data Mining., Morgan Kaufmann publishers.

Wu, G. and E. Chang (2005). "KBA: kernel boundary alignment considering imbalanced data distribution." IEEE Transactions on Knowledge Data Engineering **17**(6): 786-795.

Xu, L., et al. (2007). "Power distribution fault cause identification with imbalanced data using the data mining-based fuzzy classification EAlgorithm." IEEE Transactions on Power Systems **22**(1): 164-171.

Yang, Q. and X. Wu (2006). "10 challenging problems in data mining research." International Journal of Information Technology and Decision Making **5**(4): 597-604.

Anexos

Anexo 1. Resultados del Experimento # 1.

Tabla A.1 Resultados de algoritmos de clasificación aplicados al conjunto de datos sin preprocesar.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Bayes Net	56% (70)	44% (55)	0.4583	0.186	0.244	0.286	0.474
					0.756	0.814	0.639	
	Naive Bayes	58.4% (73)	41.6% (52)	0.4178	0.698	0.476	0.435	0.655
					0.524	0.302	0.768	
Redes neuronales	MLP (o)	50.4% (63)	49.6% (62)	0.4627	0.302	0.390	0.289	0.491
					0.610	0.698	0.625	
Métodos basados en instancias	IBK (K=11)	60% (75)	40% (50)	0.4742	0.233	0.207	0.370	0.464
					0.793	0.767	0.663	
	K Star	74.4% (93)	25.6% (32)	0.3033	0.581	0.171	0.641	0.739
					0.829	0.419	0.791	
Meta clasificadores	Atribute Select Classifier	69.6% (87)	30.4% (38)	0.304	0.465	0.183	0.571	0.641
					0.817	0.535	0.744	
	Bagging (RandomTree)	76% (95)	24% (30)	0.3336	0.581	0.146	0.676	0.759
					0.854	0.419	0.795	
	Multiclass Classifier (RandomForest)	76% (95)	24% (30)	0.3336	0.581	0.146	0.676	0.759
					0.854	0.419	0.795	
	Random Committee (RandomForest)	**76.8% (96)**	**23.2% (29)**	**0.3533**	**0.535**	**0.110**	**0.719**	**0.764**
					0.890	**0.465**	**0.785**	
	Random Subspace (OneR)	75.2% (94)	24.8% (31)	0.3040	0.465	0.098	0.714	0.700
					0.902	0.535	0.763	
Métodos basados en reglas	OneR	72.8% (91)	27.2% (34)	0.2720	0.465	0.134	0.645	0.665
					0.866	0.535	0.755	
Árboles de decisión	J48	68.8% (86)	31.2% (39)	0.3471	0.535	0.232	0.548	0.666
					0.768	0.465	0.759	
	Random Forest	76% (95)	24% (30)	0.3336	0.581	0.146	0.676	0.759
					0.854	0.419	0.795	
Support Vector Machine	C-SVM	64.8%	35.2%					

Anexo 2. Resultados del Experimento # 2.

Tabla A.2 Resultados de algoritmos de clasificación aplicados al conjunto de datos normalizados.

Algoritmos de Clasificación		**Instancias Correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	60.8% (76)	39.2% (49)	0.4512	0.047	0.098	0.200	0.495
					0.902	0.953	0.643	
	Naive Bayes	61.6% (77)	38.4% (48)	0.4214	0.791	0.476	0.466	0.659
					0.524	0.209	0.827	
Redes neuronales	MLP (o)	50.4% (63)	49.6% (62)	0.4627	0.302	0.390	0.289	0.491
					0.610	0.698	0.625	
Métodos basados en instancias	IBK (K=13)	57.6% (72)	42.4% (53)	0.4763	0.163	0.207	0.292	0.451
					0.793	0.837	0.644	
	K Star	74.4% (93)	25.6% (32)	0.3033	0.581	0.171	0.641	0.739
					0.829	0.419	0.791	
Meta clasificadores	Atribute Select Classifier	69.6% (87)	30.4% (38)	0.3040	0.465	0.183	0.571	0.641
					0.817	0.535	0.744	
	Bagging (RForest)	**76% (95)**	**24% (30)**	**0.3618**	**0.512**	**0.110**	**0.710**	**0.763**
					0.890	**0.488**	**0.777**	
	Multiclass Classifier (KStar)	74.4% (93)	25.6% (32)	0.3033	0.581	0.171	0.641	0.739
					0.829	0.419	0.791	
	Random Committee (RForest)	**76% (95)**	**24% (30)**	**0.3662**	**0.512**	**0.110**	**0.710**	**0.741**
					0.890	**0.488**	**0.777**	
	Random Subspace (RIPTree)	75.2% (94)	24.8% (31)	0.3871	0.372	0.049	0.800	0.768
					0.951	0.628	0.743	
Métodos basados en reglas	OneR	72.8% (91)	27.2% (34)	0.2720	0.465	0.134	0.645	0.665
					0.866	0.535	0.755	
Árboles de decisión	J48	68.8% (86)	31.2% (39)	0.3471	0.535	0.232	0.548	0.666
					0.768	0.465	0.759	
	Random Forest	72% (90)	28% (35)	0.3512	0.628	0.232	0.587	0.754
					0.768	0.372	0.797	

Tabla A.3 Resultados de algoritmos de clasificación aplicados al conjunto de datos discretizados.

Algoritmos de Clasificación		**Instancias Correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	80% (100)	20% (25)	0.3079	0.674	0.134	0.725	0.771
					0.866	0.326	0.835	
	Naive Bayes	71.2% (89)	28.8% (36)	0.3386	0.581	0.220	0.581	0.733
					0.780	0.419	0.780	
Redes neuronales	MLP (o)	71.2% (89)	28.8% (36)	0.3003	0.558	0.207	0.585	0.718
					0.793	0.442	0.774	
Métodos basados en instancias	IBK (K=5)	71.2% (89)	28.8% (36)	0.4059	0.372	0.110	0.640	0.655
					0.890	0.628	0.730	
	K Star	70.4% (88)	29.6% (37)	0.3119	0.488	0.183	0.583	0.748
					0.817	0.512	0.753	
Meta clasificadores	**Atribute Select Classifier (IBk)**	**83.2% (104)**	**16.8% (21)**	**0.2272**	**0.767**	**0.134**	**0.750**	**0.851**
					0.866	**0.233**	**0.877**	
	Bagging (J48)	81.6% (102)	18.4% (23)	0.2854	0.628	0.085	0.794	0.829
					0.915	0.372	0.824	
	Multiclass Classifier (J48)	80.8% (101)	19.2% (24)	0.2497	0.698	0.134	0.732	0.788
					0.866	0.302	0.845	
	Random Committee (RForest)	80.8% (101)	19.2% (24)	0.3351	0.605	0.085	0.788	0.825
					0.915	0.395	0.815	
	Random Subspace (J48)	80.8% (101)	19.2% (24)	0.3618	0.558	0.061	0.828	0.796
					0.939	0.442	0.802	
Métodos basados en reglas	OneR	73.6% (92)	26.4% (33)	0.2640	0.442	0.110	0.679	0.666
					0.890	0.558	0.753	
Árboles de decisión	J48	80.8% (101)	19.2% (24)	0.2497	0.698	0.134	0.732	0.788
					0.866	0.302	0.845	
	REPTree	77.6% (97)	22.4% (28)	0.3018	0.535	0.098	0.742	0.768
					0.902	0.465	0.787	
Support Vector Machine	C-SVM	60%	40%					

Tabla A.4 Resultados de algoritmos de clasificación aplicados al conjunto de datos normalizados y discretizados.

Algoritmos de Clasificación		**Instancias Correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	80% (100)	20% (25)	0.3079	0.674	0.134	0.725	0.771
					0.866	0.326	0.835	
	Naive Bayes	71.2% (89)	28.8% (36)	0.3386	0.581	0.220	0.581	0.733
					0.780	0.419	0.780	
Redes neuronales	MLP (o)	71.2% (89)	28.8% (36)	0.3003	0.558	0.207	0.585	0.718
					0.793	0.442	0.774	
Métodos basados en instancias	IBK (K=5)	71.2% (89)	28.8% (36)	0.4059	0.372	0.110	0.640	0.655
					0.890	0.628	0.730	
	K Star	70.4% (88)	29.6% (37)	0.3119	0.488	0.183	0.583	0.748
					0.817	0.512	0.753	
Meta clasificadores	**Atribute Select Classifier (IBk)**	**83.2% (104)**	**16.8% (21)**	**0.2272**	**0.767**	**0.134**	**0.750**	**0.851**
					0.866	**0.233**	**0.877**	
	Bagging (J48)	81.6% (102)	18.4% (23)	0.2854	0.628	0.085	0.794	0.829
					0.915	0.372	0.824	
	Multiclass Classifier (J48)	80.8% (101)	19.2% (24)	0.2497	0.698	0.134	0.732	0.788
					0.866	0.302	0.845	
	Random Committee (RForest)	80% (100)	20% (25)	0.3286	0.581	0.085	0.781	0.829
					0.915	0.419	0.806	
	Random Subspace (J48)	77.6% (97)	22.4% (28)	0.3608	0.488	0.073	0.778	0.813
					0.927	0.512	0.776	
Métodos basados en reglas	OneR	73.6% (92)	26.4% (33)	0.2640	0.442	0.110	0.679	0.666
					0.890	0.558	0.753	
Árboles de decisión	J48	80.8% (101)	19.2% (24)	0.2497	0.698	0.134	0.732	0.788
					0.866	0.302	0.845	
	REPTree	77.6% (97)	22.4% (28)	0.3018	0.535	0.098	0.742	0.768
					0.902	0.465	0.787	

Tabla A.5 Resultados de algoritmos de clasificación aplicados al conjunto de datos con *Spread Subsample* y discretizando.

Algoritmos de Clasificación		**Instancias Correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	73.6 % (92)	26.4% (33)	0.3071	0.651	0.220	0.609	0.786
					0.780	0.349	0.810	
	Naive Bayes	70.4% (88)	29.6% (37)	0.3386	0.581	0.232	0.568	0.739
					0.768	0.419	0.778	
Redes neuronales	MLP (i)	75.2% (94)	24.8% (31)	0.2699	0.651	0.195	0.636	0.752
					0.805	0.349	0.815	
Métodos basados en instancias	IBK (K=1)	72% (90)	28% (35)	0.2878	0.628	0.232	0.587	0.740
					0.768	0.372	0.797	
	K Star	72.8% (91)	27.2% (34)	0.2912	0.558	0.183	0.615	0.775
					0.817	0.442	0.779	
Meta clasificadores	**Atribute Select Classifier (RForest)**	**84% (105)**	**16% (20)**	**0.2336**	**0.767**	**0.122**	**0.767**	**0.848**
					0.878	**0.233**	**0.878**	
	Bagging (J48)	83.2% (104)	16.8% (21)	0.2836	0.628	0.061	0.844	0.839
					0.939	0.372	0.828	
	Multiclass Classifier (J48)	83.2% (104)	16.8% (21)	0.2474	0.698	0.098	0.789	0.791
					0.902	0.302	0.851	
	Random Committee (RForest)	82.4% (103)	17.6% (22)	0.3266	0.651	0.085	0.800	0.861
					0.915	0.349	0.833	
	RandomSubspace (RForest)	81.6% (102)	18.4% (23)	0.3396	0.605	0.073	0.813	0.849
					0.927	0.395	0.817	
Métodos basados en reglas	PART	75.2% (94)	24.8% (31)	0.3361	0.512	0.122	0.688	0.693
					0.878	0.488	0.774	
Árboles de decisión	J48	83.2% (104)	16.8% (21)	0.2474	0.698	0.098	0.789	0.791
					0.902	0.302	0.851	
	LMT	80% (100)	20% (25)	0.2956	0.651	0.122	0.737	0.812
					0.878	0.349	0.828	

Anexo 3. Resultados del Experimento # 3.

Tabla A.6 Resultados de algoritmos de clasificación que manejan datos con clases desbalanceadas aplicados al conjunto de datos en estudio.

Algoritmos de Clasificación		**Exactitud**	**stddev**	**Error esperado**	**Error Medio**	**Matriz de Confusión (test)**	**Matriz de Confusión (training)**
Ada Boost	Test	0.624	0.0407	0.3760	0.3600	17 26 2 21 61 1	172 0 2 0 328 1
	Training	1.0	0.0				
AdaBoostM1	Test	0.656	0.0598	0.3440	0.3600	23 20 2 23 59 1	172 0 2 0 328 1
	Training	1.0	0.0				
AdaBoostM2	Test	0.656	0.0598	0.3440	0.3600	23 20 2 23 59 1	172 0 2 0 328 1
	Training	1.0	0.0				
AdaC2	Test	0.648	0.0640	0.3520	0.3600	35 8 2 36 46 1	172 0 2 4 324 1
	Training	0.992	0.0160				
Bagging	Test	0.704	0.0897	0.2960	0.2400	21 22 2 15 67 1	139 33 2 10 318 1
	Training	0.914	0.0241				
Balance Cascade	Test	0.544	0.1175	0.4560	0.4800	37 6 2 51 31 1	171 1 2 164 164 1
	Training	0.670	0.1046				
Data Boost-IM	Test	0.632	0.0159	0.3680	0.3600	24 19 2 27 55 1	172 0 2 0 328 1
	Training	1.0	0.0				
Easy Ensemble	Test	0.576	0.0649	0.4240	0.4000	31 12 2 41 41 1	164 8 2 74 254 1
	Training	0.836	0.0407				
II Votes	Test	0.520	0.1213	0.4800	0.5200	32 11 2 49 33 1	153 19 2 152 176 1
	Training	0.658	0.1718				
MSMOTE Bagging	Test	0.576	0.0697	0.4240	0.4000	32 11 2 42 40 1	166 6 2 99 229 1
	Training	0.790	0.0404				
MSMOTE Boost	Test	0.576	0.0542	0.4240	0.4000	24 19 2 34 48 1	165 7 2 62 266 1
	Training	0.862	0.0331				
Over Bagging	Test	0.704	0.0741	0.2960	0.2800	25 18 2 19 63 1	158 14 2 17 311 1
	Training	0.938	0.0213				
Over Bagging2	Test	0.704	0.0407	0.2960	0.2800	27 16 2 21 61 1	161 11 2 25 303 1
	Training	0.928	0.0097				
RUS Boost	Test	0.576	0.1175	0.4240	0.4000	29 14 2 39 43 1	172 0 2 35 293 1
	Training	0.930	0.0275				
SMOTE Bagging	Test	0.720	0.0715	0.2800	0.2400	27 16 2 19 63 1	157 15 2 33 295 1
	Training	0.903	0.0101				
SMOTE Boost	Test	0.696	0.0542	0,3040	0,2800	24 19 2 19 63 1	172 0 2 0 328 1
	Training	1.0	0.0				
Under Bagging	Test	0.576	0.0697	0.4240	0.4000	31 12 2 41 41 1	166 6 2 69 259 1
	Training	0.850	0.0109				
Under Bagging2	Test	0.664	0.0542	0.3360	0.3200	29 14 2 28 54 1	158 14 2 45 283 1
	Training	0.882	0.0248				
Under Over Bagging	Test	0.640	0.0669	0.3600	0.3600	28 15 2 30 52 1	161 11 2 41 287 1
	Training	0.895	0.0135				

Anexo 4. Resultados del Experimento # 4.

Tabla A.7 Resultados de algoritmos de clasificación que manejan datos con clases desbalanceadas aplicados al conjunto de datos discretizado.

Clasificadores		**Exactitud**	**stddev**	**Error esperado**	**Error Medio**	**Matriz de Confusión (test)**	**Matriz de Confusión (training)**
Ada Boost	Test	0.728	0.0733	0.2720	0.2400	28 15 2	172 0 2
	Training	1.0	0.0			19 63 1	0 328 1
AdaBoostM1	Test	0.728	0.0733	0.2720	0.2400	28 15 2	172 0 2
	Training	1.0	0.0			19 63 1	0 328 1
AdaBoostM2	Test	0.728	0.0733	0.2720	0.2400	28 15 2	172 0 2
	Training	1.0	0.0			19 63 1	0 328 1
AdaC2	Test	0.616	0.0741	0.3840	0.4000	36 7 2	172 0 2
	Training	0.950	0.0352			41 41 1	25 303 1
Bagging	Test	0.752	0.0587	0.2480	0.2400	23 20 2	120 52 2
	Training	0.874	0.0101			11 71 1	11 317 1
Balance Cascade	Test	0.584	0.0741	0.4160	0.4000	33 10 2	163 9 2
	Training	0.750	0.0389			42 40 1	116 212 1
Data Boost-IM	Test	0.688	0.1348	0.3120	0.3200	29 14 2	172 0 2
	Training	1.0	0.0			25 57 1	0 328 1
Easy Ensemble	Test	0.663	0.0897	0.3360	0.3600	32 11 2	153 19 2
	Training	0.811	0.0292			31 51 1	75 253 1
II Votes	Test	0.552	0.1274	0.4480	0.4000	26 17 2	137 35 2
	Training	0.708	0.1807			39 43 1	111 217 1
MSMOTE Bagging	Test	0.672	0.0926	0.3280	0.3600	29 14 2	130 42 2
	Training	0.784	0.0656			27 55 1	66 262 1
MSMOTE Boost	Test	0.663	0.1399	0.3360	0.2800	31 12 2	166 6 2
	Training	0.880	0.0517			30 52 1	54 274 1
Over Bagging	Test	0.704	0.0407	0.2960	0.2800	30 13 2	164 8 2
	Training	0.935	0.0135			15 67 1	24 304 1
Over Bagging2	Test	0.736	0.1148	0.2640	0.2800	27 16 2	161 11 2
	Training	0.916	0.0338			21 61 1	25 303 1
RUS Boost	Test	0.760	0.0669	0.2400	0.2400	32 11 2	166 6 2
	Training	0.920	0.0357			19 63 1	34 294 1
SMOTE Bagging	Test	0.696	0.0649	0.3040	0.3200	28 15 2	127 45 2
	Training	0.774	0.0300			23 59 1	68 260 1
SMOTE Boost	Test	0.696	0.0542	0,3040	0,2800	24 19 2	172 0 2
	Training	1.0	0.0			19 63 1	0 328 1
Under Bagging	Test	0.712	0.0530	0.2880	0.2800	30 13 2	134 38 2
	Training	0.822	0.0426			23 59 1	51 277 1
Under Bagging2	Test	0.736	0.0783	0.2640	0.2000	32 11 2	155 17 2
	Training	0.894	0.0287			22 60 1	36 292 1
Under Over Bagging	Test	0.720	0.0619	0.2800	0.2400	30 13 2	158 14 2
	Training	0.884	0.0257			22 60 1	44 284 1

Anexo 5. Resultados del Experimento # 5.

Tabla A.8 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por Borderline_SMOTE1

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	68.29% (112)	31.70% (52)	0.3388	0.707	0.341	0.674	0.761
					0.659	0.293	0.692	
	Naive Bayes	62.80% (103)	37.19% (61)	0.3659	0.927	0.671	0.580	0.756
					0.329	0.073	0.818	
Redes neuronales	MLP	63.41% (104)	36.58% (60)	0.3847	0.646	0.378	0.631	0.657
					0.622	0.354	0.638	
Métodos basados en instancias	IBK (k=1)	61.58% (101)	38.41% (63)	0.3857	0.695	0.463	0.600	0.621
					0.537	0.305	0.638	
	K Star	82.92% (136)	17.07% (28)	0.1908	0.939	0.280	0.770	0.928
					0.720	0.061	0.922	
Meta clasificadores	Atribute Select Classifier	77.43% (127)	22.56% (37)	0.304	0.793	0.244	0.765	0.759
					0.756	0.207	0.785	
	Bagging (RandomForest)	**84.14% (138)**	**15.85% (26)**	**0.2929**	**0.829**	**0.146**	**0.850**	**0.895**
					0.854	**0.171**	**0.833**	
	Filtered Classifier	72.56% (119)	27.43% (45)	0.3223	0.756	0.305	0.713	0.768
					0.695	0.244	0.740	
	Multiclass Classifier	62.19% (102)	37.80% (62)	0.4558	0.671	0.427	0.611	0.608
					0.573	0.329	0.635	
	Random Subspace (RandomForest)	83.53% (137)	16.46% (27)	0.3034	0.854	0.183	0.824	0.909
					0.817	0.146	0.848	
Métodos basados en reglas	JRip	75.60% (124)	75.60% (40)	0.3293	0.732	0.220	0.769	0.779
					0.780	0.268	0.744	
Árboles de decisión	J48	74.39% (122)	25.60% (42)	0.3246	0.732	0.244	0.750	0.736
					0.756	0.268	0.738	
	Random Forest	75% (123)	25% (41)	0.3037	0.780	0.280	0.736	0.838
					0.720	0.220	0.766	

Tabla A.9 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por Borderline_SMOTE-I y luego normalizados.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	71.34% (117)	28.65% (47)	0.3315	0.707	0.280	0.716	0.760
					0.720	0.293	0.711	
	Naive Bayes	63.41% (104)	36.58% (60)	0.3678	0.927	0.659	0.585	0.757
					0.341	0.073	0.824	
Redes neuronales	MLP (a)	63.41% (104)	36.58% (60)	0.3847	0.646	0.378	0.631	0.657
					0.622	0.354	0.638	
Métodos basados en instancias	IBK (k=1)	61.58% (101)	38.41% (63)	0.3857	0.695	0.463	0.600	0.621
					0.537	0.305	0.638	
	K Star	82.92% (136)	17.07% (28)	0.1908	0.939	0.280	0.770	0.928
					0.720	0.061	0.922	
Multi clasificadores	Atribute Select Classifier (RandomForest)	77.43% (127)	22.56% (37)	0.2649	0.817	0.268	0.753	0.834
					0.732	0.183	0.800	
	Bagging (RandomForest)	**84.75% (139)**	**15.24% (25)**	**0.2918**	**0.841**	**0.146**	**0.852**	**0.889**
					0.854	**0.159**	**0.843**	
	Multiclass Classifier (KStar)	82.92% (136)	17.07% (28)	0.1908	0.939	0.280	0.770	0.928
					0.720	0.061	0.922	
	Random Subspace (KStar)	84.14% (138)	15.85% (26)	0.3217	0.915	0.232	0.798	0.919
					0.768	0.085	0.900	
Métodos basados en reglas	JRip	76.21% (125)	23.78% (39)	0.3191	0.695	0.171	0.803	0.779
					0.829	0.305	0.731	
Árboles de decisión	J48	72.56% (119)	27.43% (45)	0.3173	0.756	0.305	0.713	0.717
					0.695	0.244	0.740	
	Random Forest	81.09% (133)	18.90% (31)	0.2829	0.866	0.244	0.780	0.876
					0.756	0.134	0.849	

Tabla A.10 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por Borderline_SMOTE-I y luego discretizados.

Algoritmos de Clasificación		Instancias correctas	Instancias incorrectas	Error M. absoluto	TPRate	FPRate	Precisión	ROC Área
Redes bayesianas	Redes Bayesianas	74.39% (122)	25.60% (42)	0.2611	0.732	0.244	0.750	0.854
					0.756	0.268	0.738	
	Naive Bayes	75% (123)	25% (41)	0.2695	0.732	0.232	0.759	0.849
					0.768	0.268	0.741	
Redes neuronales	MLP (o)	77.43% (127)	22.56% (37)	0.2548	0.768	0.220	0.778	0.819
					0.780	0.232	0.771	
Métodos basados en instancias	IBK (k=1)	81.70% (134)	18.29% (30)	0.1834	0.878	0.244	0.783	0.864
					0.756	0.122	0.861	
	K Star	84.75% (139)	15.24% (25)	0.1868	0.878	0.183	0.828	0.933
					0.817	0.122	0.870	
Meta clasificadores	Atribute Select Classifier (RForest)	81.70% (134)	18.29% (30)	0.2608	0.854	0.220	0.795	0.846
					0.780	0.146	0.842	
	Bagging (RForest)	85.97% (141)	14.02% (23)	0.2989	0.866	0.146	0.855	0.924
					0.854	0.134	0.864	
	Multiclass Classifier (KStar)	84.75% (139)	15.24% (25)	0.1868	0.878	0.183	0.828	0.933
					0.817	0.122	0.870	
	RSubspace (RForest)	**88.41% (145)**	11.58% (19)	0.2998	0.866	0.098	0.899	0.925
					0.902	0.134	0.871	
Métodos basados en reglas	JRip	74.39% (122)	25.60% (42)	0.2561	0.780	0.293	0.727	0.744
					0.707	0.220	0.763	
Árboles de decisión	J48	79.26% (130)	20.73% (34)	0.2881	0.793	0.207	0.793	0.816
	Random Forest	82.31% (135)	17.68% (29)	0.2817	0.817	0.171	0.827	0.912
					0.829	0.183	0.819	

Tabla A.11 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SPIDER2.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	67.32% (103)	67.32% (50)	0.3868	0.780	0.484	0.703	0.672
					0.516	0.220	0.615	
	Naive Bayes	71.24% (109)	28.75% (44)	0.3408	0.912	0.581	0.697	0.725
					0.419	0.088	0.765	
Redes neuronales	MLP	73.20% (112)	26.79% (41)	0.3103	0.802	0.371	0.760	0.696
					0.629	0.198	0.684	
Métodos basados en instancias	IBK (K=8)	65.35% (100)	34.64% (53)	0.4229	0.934	0.758	0.644	0.617
					0.242	0.066	0.714	
	K Star	78.43% (120)	21.56% (33)	0.2387	0.868	0.339	0.790	0.867
					0.661	0.132	0.774	
Meta clasificadores	Atribute Select Classifier (RForest)	84.96% (130)	15.03% (23)	0.1911	0.912	0.242	0.847	0.899
					0.758	0.088	0.855	
	Bagging (RForest)	84.31% (129)	15.68% (24)	0.2669	0.901	0.242	0.845	0.894
					0.758	0.099	0.839	
	Multiclass Classifier	83.66% (128)	16.33% (25)	0.2373	0.934	0.306	0.817	0.903
					0.694	0.066	0.878	
	Random Committee (RTree)	**86.27% (132)**	**13.72% (21)**	**0.2229**	**0.890**	**0.177**	**0.880**	**0.899**
					0.823	**0.110**	**0.836**	
	Random Subspace	84.96% (130)	15.03% (23)	0.2437	0.901	0.226	0.854	0.925
					0.774	0.099	0.842	
Métodos basados en reglas	JRip	79.73% (122)	20.26% (31)	0.2677	0.890	0.339	0.794	0.769
					0.661	0.110	0.804	
Árboles de decisión	J48	79.73% (122)	20.26% (31)	0.2623	0.868	0.306	0.806	0.762
					0.694	0.132	0.782	
	Random Forest	83.66% (128)	16.33% (25)	0.2373	0.934	0.306	0.817	0.903
					0.694	0.066	0.878	

Tabla A.12 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SPIDER2 y luego discretizados.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	75.16% (115)	24.83% (38)	0.2849	0.824	0.355	0.773	0.800
					0.645	0.176	0.714	
	Naive Bayes	75.16% (115)	24.83% (38)	0.2907	0.824	0.355	0.773	0.801
					0.645	0.176	0.714	
Redes neuronales	MLP (o)	77.77% (119)	22.22% (34)	0.2350	0.879	0.371	0.777	0.800
					0.629	0.121	0.780	
Métodos basados en instancias	IBK (K=1)	81.69% (125)	18.30% (28)	0.2028	0.923	0.339	0.800	0.834
					0.661	0.077	0.854	0.833
	K Star	**84.31% (129)**	**15.68% (24)**	**0.2256**	**0.923**	**0.274**	**0.832**	**0.880**
					0.726	0.077	0.865	
Meta clasificadores	Atribute Select Classifier (IBK)	77.12% (118)	22.87% (35)	0.2769	0.879	0.387	0.769	0.795
					0.613	0.121	0.776	
	Bagging (KStar)	82.35% (126)	17.64% (27)	0.2535	0.912	0.306	0.814	0.864
					0.694	0.088	0.843	
	Multiclass Classifier (KStar)	**84.31% (129)**	**15.68% (24)**	**0.2256**	**0.923**	**0.274**	**0.832**	**0.880**
					0.726	**0.077**	**0.865**	
	R Committee (RTree)	81.69% (125)	18.30% (28)	0.2807	0.901	0.306	0.812	0.896
					0.694	0.099	0.827	
	Random Subspace	80.39% (123)	19.60% (30)	0.3007	0.901	0.339	0.796	0.897
					0.661	0.099	0.820	
Métodos basados en reglas	PART	68.62% (105)	31.37% (48)	0.3597	0.714	0.355	0.747	0.652
					0.645	0.286	0.606	
Árboles de decisión	J48	74.50% (114)	25.49% (39)	0.2829	0.846	0.403	0.755	0.801
					0.597	0.154	0.725	
	Random Forest	79.73% (122)	20.26% (31)	0.2797	0.901	0.355	0.788	0.881
					0.645	0.099	0.816	

Tabla A.13 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por Safe_Level_SMOTE-I.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	53.04% (87)	46.95% (77)	0.4872	0.756	0.695	0.521	0.542
					0.305	0.244	0.556	
	Naive Bayes	62.19 (102)	37.80% (62)	0.4010	0.915	0.671	0.577	0.691
					0.329	0.085	0.794	
Redes neuronales	MLP	59.14% (97)	40.85% (67)	0.4266	0.634	0.451	0.584	0.589
					0.549	0.366	0.600	
Métodos basados en instancias	IBK (K=2)	55.48% (91)	44.51% (73)	0.4818	0.866	0.756	0.534	0.549
					0.244	0.134	0.645	
	K Star	63.41% (104)	36.58% (60)	0.3900	0.671	0.402	0.625	0.648
					0.598	0.329	0.645	
Meta clasificadores	Atribute Select Classifier	65.85% (108)	34.14% (56)	0.3756	0.683	0.366	0.651	0.71
					0.634	0.317	0.667	
	Bagging (J48)	70.73% (116)	29.26% (48)	0.3698	0.720	0.305	0.702	0.695
					0.695	0.280	0.713	
	Multiclass Classifier	72.56% (119)	27.43% (45)	0.3370	0.695	0.244	0.740	0.726
					0.756	0.305	0.713	
	R Committee (Random Forest)	70.73% (116)	29.26% (48)	0.3835	0.683	0.268	0.718	0.695
					0.732	0.317	0.698	
	Random Subspace	**73.17% (120)**	**26.82% (44)**	**0.3991**	**0.695**	**0.232**	**0.750**	**0.753**
					0.768	**0.305**	**0.716**	
Métodos basados en reglas	JRip	64.02% (105)	35.97% (59)	0.4365	0.659	0.378	0.635	0.622
					0.622	0.341	0.646	
Árboles de decisión	J48	72.56% (119)	27.43% (45)	0.3370	0.695	0.244	0.740	0.726
					0.756	0.305	0.713	
	Random Forest	64.63% (106)	35.36% (58)	0.3884	0.671	0.378	0.640	0.691
					0.622	0.329	0.654	

Tabla A.14 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SMOTE_ENN.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	69%	31%	0.4022	0.708	0.327	0.667	0.650
					0.673	0.292	0.714	
	Naive Bayes	70%	30%	0.3770	0.708	0.308	0.680	0.709
					0.692	0.292	0.720	0.622
Redes neuronales	MLP	61%	39%	0.4078	0.583	0.365	0.596	0.640
					0.635	0.417	0.623	
Métodos basados en instancias	IBK (K=1)	65%	35%	0.3533	0.646	0.346	0.633	0.650
					0.654	0.354	0.667	
	K Star	68%	32%	0.3512	0.688	0.327	0.660	0.740
					0.673	0.313	0.700	
Meta clasificadores	Atribute Select Classifier	76%	24%	0.2923	0.771	0.250	0.740	0.792
					0.750	0.229	0.780	
	Bagging (J48)	**80%**	**20%**	**0.2967**	**0.771**	**0.173**	**0.804**	**0.845**
					0.827	**0.229**	**0.796**	
	Multiclass Classifier	78%	22%	0.2200	0.708	0.154	0.810	0.777
					0.846	0.292	0.759	
	Random Committee (RTree)	78%	22%	0.2720	0.771	0.212	0.771	0.854
					0.788	0.229	0.788	
	Random Subspace	78%	22%	0.3488	0.792	0.231	0.760	0.847
					0.769	0.208	0.800	
Métodos basados en reglas	JRip	71%	29%	0.3864	0.750	0.327	0.679	0.648
					0.673	0.250	0.745	
Árboles de decisión	J48	74%	26%	0.2906	0.688	0.212	0.750	0.722
					0.788	0.313	0.732	
	Random Tree	78%	22%	0.22	0.708	0.154	0.810	0.777
					0.846	0.292	0.759	

Tabla A.15 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SMOTE_RSB.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	64.63% (106)	35.36% (58)	0.3995	0.695	0.402	0.633	0.683
					0.598	0.305	0.662	
	Naive Bayes	61.58% (101)	38.41% (63)	0.4129	0.902	0.671	0.574	0.656
					0.329	0.098	0.771	
Redes neuronales	MLP	59.75% (98)	40.24% (66)	0.4047	0.622	0.427	0.593	0.626
					0.573	0.378	0.603	
Métodos basados en instancias	IBK (K=1)	57.92% (95)	42.07% (69)	0.4606	0.866	0.707	0.550	0.549
					0.293	0.134	0.686	
	K Star	79.26% (130)	20.73% (34)	0.2374	0.866	0.280	0.755	0.861
					0.720	0.134	0.843	
Meta clasificadores	Atribute Select Classifier	79.26% (130)	20.73% (34)	0.2679	0.805	0.220	0.786	0.841
					0.780	0.195	0.800	
	Bagging (RandomForest)	80.48% (132)	19.51% (32)	0.3334	0.780	0.171	0.821	0.844
					0.829	0.220	0.791	
	Multiclass Classifier	79.26% (130)	20.73% (34)	0.2374	0.866	0.280	0.755	0.861
					0.720	0.134	0.843	
	Random Committee (RTree)	80.48% (132)	19.51% (32)	0.2817	0.841	0.232	0.784	0.867
					0.768	0.159	0.829	
	Random Subspace	**81.09% (133)**	**18.90% (31)**	**0.3190**	**0.817**	**0.195**	**0.807**	**0.863**
					0.805	**0.183**	**0.815**	
Métodos basados en reglas	JRip	69.51% (114)	30.48% (50)	0.3616	0.695	0.305	0.695	0.754
Árboles de decisión	J48	75.60% (124)	24.39% (40)	0.2976	0.756	0.244	0.756	0.732
	Random Forest	77.43% (127)	22.56% (37)	0.3079	0.817	0.268	0.753	0.732
					0.732	0.183	0.800	

Tabla A.16 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SMOTE_TL con 122 instancias en total.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	68.85% (84)	31.14% (38)	0.3751	0.768	0.475	0.768	0.655
					0.525	0.232	0.525	
	Naive Bayes	74.59% (91)	25.40% (31)	0.2767	0.939	0.650	0.748	0.790
					0.350	0.061	0.737	
Redes neuronales	MLP	72.13% (88)	27.86% (34)	0.2806	0.805	0.450	0.786	0.746
					0.550	0.195	0.579	
Métodos basados en instancias	IBK (K=1)	77.86% (95)	22.13% (27)	0.2263	0.841	0.350	0.831	0.729
					0.650	0.159	0.667	
	K Star	76.22% (93)	23.77% (29)	0.2358	0.939	0.600	0.762	0.848
					0.400	0.061		
Meta clasificadores	Atribute Select Classifier	76.22% (93)	23.77% (29)	0.2676	0.854	0.425	0.805	0.765
					0.575	0.146	0.657	
	Bagging (RandomForest)	81.14% (99)	18.85% (23)	0.2875	0.915	0.400	0.824	0.870
					0.600	0.085	0.774	
	Multiclass Classifier	79.50% (97)	20.49% (25)	0.2795	0.927	0.475	0.800	0.847
					0.525	0.073	0.778	
	Random Committee (RandomForest)	81.96% (100)	18.03% (22)	0.2700	0.902	0.350	0.841	0.884
					0.650	0.098	0.765	
	Random Subspace	**82.78% (101)**	**17.21% (21)**	**0.2633**	**0.951**	**0.425**	**0.821**	**0.892**
					0.575	**0.049**	**0.852**	
Métodos basados en reglas	JRip	76.22% (93)	23.77% (29)	0.3002	0.841	0.400	0.812	0.711
					0.600	0.159	0.649	
Árboles de decisión	J48	77.04% (94)	22.95% (28)	0.2651	0.890	0.475	0.793	0.687
					0.525	0.110	0.700	
	Random Forest	79.50% (97)	20.49% (25)	0.2795	0.927	0.475	0.800	0.847
					0.525	0.073	0.778	

Tabla A.17 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SMOTE con 164 instancias en total.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	68.29% (112)	31.70% (52)	0.4033	0.659	0.293	0.692	0.691
					0.707	0.341	0.674	0.692
	Naive Bayes	67.07% (110)	31.92% (54)	0.4065	0.659	0.317	0.675	0.685
					0.683	0.341	0.667	
Redes neuronales	MLP (t)	57.92% (95)	42.07% (69)	0.4317	0.634	0.476	0.571	0.580
					0.524	0.366	0.589	
Métodos basados en instancias	IBK (K=2)	56.70% (93)	43.29% (71)	0.4546	0.890	0.756	0.541	0.577
					0.244	0.110	0.690	
	K Star	70.73% (116)	29.26% (48)	0.2999	0.793	0.378	0.677	0.797
					0.622	0.207	0.750	
Meta clasificadores	Atribute Select Classifier	75.60% (124)	24.39% (40)	0.2756	0.805	0.293	0.733	0.834
					0.707	0.195	0.784	
	Bagging (RandomForest)	**80.48% (132)**	**19.51% (32)**	**0.3294**	**0.805**	**0.195**	**0.805**	**0.858**
	Multiclass Classifier	78.65% (129)	21.34% (35)	0.2686	0.829	0.256	0.764	0.751
					0.744	0.171	0.813	
	Random Committee (RandomTree)	**80.48% (132)**	**19.51% (32)**	**0.2841**	**0.841**	**0.232**	**0.784**	**0.848**
					0.768	**0.159**	**0.829**	
	Random Subspace	79.87% (131)	20.12% (33)	0.3472	0.793	0.195	0.802	0.869
					0.805	0.207	0.795	
Métodos basados en reglas	JRip	70.12% (115)	29.87% (49)	0.3638	0.720	0.317	0.694	0.739
					0.683	0.280	0.709	
Árboles de decisión	J48	78.65% (129)	21.34% (35)	0.2686	0.829	0.256	0.764	0.751
					0.744	0.171	0.813	
	Random Forest	78.04% (128)	21.95% (36)	0.3207	0.805	0.244	0.767	0.819
					0.756	0.195	0.795	

Tabla A.18 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SMOTE_RSB* y RST.

Algoritmos de Clasificación		**Instancias Correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	71.95% (118)	28.04% (46)	0.3292	0.707	0.268	0.725	0.77
					0.732	0.293	0.714	
	Naive Bayes	60.97% (100)	39.02% (64)	0.3893	0.683	0.463	0.596	0.695
					0.537	0.317	0.629	
Redes neuronales	MLP (o)	67.07% (110)	32.92% (54)	0.3527	0.634	0.293	0.684	0.704
					0.707	0.366	0.659	
Métodos basados en instancias	IBK (K=7)	66.46% (109)	33.53% (55)	0.3887	0.756	0.427	0.639	0.731
					0.573	0.244	0.701	
	K Star	78.65% (129)	21.34% (35)	0.2288	0.768	0.195	0.797	0.867
					0.805	0.232	0.776	
Meta clasificadores	Atribute Select Classifier (RTree)	78.65% (129)	21.34% (35)	0.2155	0.805	0.232	0.776	0.786
					0.768	0.195	0.797	
	Bagging (RForest)	77.43% (127)	22.56% (37)	0.3071	0.756	0.207	0.785	0.861
					0.793	0.244	0.765	
	Multiclass Classifier (J48)	79.87% (131)	20.12% (33)	0.2522	0.841	0.244	0.775	0.824
					0.756	0.159	0.827	
	Random Committee (RForest)	79.87% (131)	20.12% (33)	0.2898	0.780	0.183	0.810	0.876
					0.817	0.220	0.788	
	Random Subspace (KStar)	**80.48% (132)**	**19.51% (32)**	**0.2844**	**0.793**	**0.183**	**0.813**	**0.883**
					0.817	**0.207**	**0.798**	
Métodos basados en reglas	JRip	75% (123)	25% (41)	0.3226	0.732	0.232	0.759	0.785
					0.768	0.268	0.741	
Árboles de decisión	J48	79.87% (131)	20.12% (33)	0.2522	0.841	0.244	0.775	0.824
					0.756	0.159	0.827	
	Random Forest	73.78% (121)	26.21% (43)	0.3000	0.780	0.305	0.719	0.833
					0.695	0.220	0.760	

Tabla A.19 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SMOTE_RSB* y RST y luego discretizados.

Algoritmos de Clasificación		**Instancias Correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	76.82% (126)	23.17% (38)	0.2636	0.634	0.098	0.867	0.815
					0.902	0.366	0.712	
	Naive Bayes	76.21% (125)	23.78% (39)	0.2658	0.610	0.085	0.877	0.812
					0.915	0.390	0.701	
Redes neuronales	MLP (o)	74.39% (122)	25.60% (42)	0.2686	0.732	0.244	0.750	0.798
					0.756	0.268	0.738	
Métodos basados en instancias	IBK (K=7)	77.43% (127)	22.56% (37)	0.3400	0.756	0.207	0.785	0.815
					0.793	0.244	0.765	
	K Star	77.43% (127)	22.56% (37)	0.2733	0.756	0.207	0.785	0.815
					0.793	0.244	0.765	
Meta clasificadores	Atribute Select Classifier (IBK)	74.39% (122)	25.60% (42)	0.3578	0.793	0.305	0.722	0.792
					0.695	0.207	0.770	
	Bagging (RForest)	**78.65% (129)**	**21.34% (35)**	**0.3174**	**0.720**	**0.146**	**0.831**	**0.854**
					0.854	**0.280**	**0.753**	
	Multiclass Classifier (J48)	78.04% (128)	21.95% (36)	0.3153	0.732	0.171	0.811	0.819
					0.829	0.268	0.756	
	Random Committee (RForest)	76.82% (126)	23.17% (38)	0.3113	0.695	0.159	0.814	0.844
					0.841	0.305	0.734	
	Random Subspace (RForest)	76.82% (126)	23.17% (38)	0.3189	0.707	0.171	0.806	0.843
					0.829	0.293	0.739	
Métodos basados en reglas	OneR	69.51% (114)	30.48% (50)	0.3049	0.537	0.146	0.786	0.695
					0.854	0.463	0.648	
Árboles de decisión	J48	65.85% (108)	34.14% (56)	0.3658	0.524	0.207	0.717	0.683
					0.793	0.476	0.625	
	Random Forest	78.04% (128)	21.95% (36)	0.3153	0.732	0.171	0.811	0.819
					0.829	0.268	0.756	

Tabla A.20 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por Borderline_SMOTE y el filtro supervisado Atribute Selection.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	71.34% (117)	28.65% (47)	0.3305	0.707	0.280	0.716	0.762
					0.720	0.293	0.711	
	Naive Bayes	64.63% (106)	35.36% (58)	0.3622	0.927	0.634	0.594	0.792
					0.366	0.073	0.833	
Redes neuronales	MLP (a)	64.63% (126)	23.17% (38)	0.2970	0.829	0.293	0.739	0.792
					0.707	0.171	0.806	
Métodos basados en instancias	IBK (K=1)	76.82% (126)	23.17% (38)	0.2352	0.866	0.329	0.724	0.772
					0.671	0.134	0.833	
	K Star	**81.70% (134)**	**18.29% (30)**	**0.2459**	**0.878**	**0.244**	**0.783**	**0.900**
					0.756	**0.122**	**0.861**	
Meta clasificadores	Atribute Select Classifier	77.43% (127)	22.56% (37)	0.2998	0.793	0.244	0.765	0.763
					0.756	0.207	0.785	
	Bagging (RandomForest)	80.48% (132)	19.51% (32)	0.2569	0.829	0.220	0.791	0.887
					0.780	0.171	0.821	
	Multiclass Classifier	78.65% (129)	21.34% (35)	0.2134	0.793	0.220	0.783	0.787
					0.780	0.207	0.790	
	Random Committee (RandomTree)	79.26% (130)	20.73% (34)	0.2412	0.817	0.232	0.779	0.894
					0.768	0.183	0.808	
	Random Subspace	79.87% (131)	20.12% (33)	0.2771	0.817	0.220	0.788	0.891
					0.780	0.183	0.810	
Métodos basados en reglas	JRip	75% (123)	25% (41)	0.3404	0.768	0.268	0.741	0.772
					0.732	0.232	0.759	
Árboles de decisión	J48	78.04% (128)	21.95% (36)	0.2949	0.793	0.232	0.774	0.775
					0.768	0.207	0.788	
	Random Tree	78.65% (129)	21.65% (35)	0.2134	0.793	0.220	0.783	0.787
					0.780	0.207	0.790	

Tabla A.21 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SPIDER2 y el filtro supervisado Atribute Selection.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	65.35% (100)	34.64% (53)	0.4105	0.758	0.500	0.690	0.641
					0.500	0.242	0.585	
	Naive Bayes	66.01% (101)	33.98% (52)	0.3840	0.956	0.774	0.644	0.643
					0.226	0.044	0.778	
Redes neuronales	MLP (t)	77.77% (119)	22.22% (34)	0.3245	0.890	0.387	0.771	0.764
					0.613	0.110	0.792	
Métodos basados en instancias	IBK (K=1)	85.62% (131)	14.37% (22)	0.1550	0.934	0.258	0.842	0.892
					0.742	0.066	0.885	
	K Star	80.39% (123)	19.60% (30)	0.3254	0.934	0.387	0.780	0.840
					0.613	0.066	0.864	
Meta clasificadores	**Atribute Select Classifier**	**86.92% (133)**	**13.07% (20)**	**0.1373**	**0.912**	**0.194**	**0.874**	**0.869**
					0.806	**0.088**	**0.862**	
	Bagging (RandomTree)	**86.92% (133)**	**13.07% (20)**	**0.1518**	**0.923**	**0.210**	**0.866**	**0.912**
					0.790	**0.077**	**0.875**	
	Multiclass Classifier	**86.92% (133)**	**13.07% (20)**	**0.1373**	**0.912**	**0.194**	**0.874**	**0.869**
					0.806	**0.088**	**0.862**	
	Random Committee (RTree)	86.92% (133)	13.07% (20)	0.1373	0.912	0.194	0.874	0.869
					0.806	0.088	0.862	
	Random Subspace	85.62% (131)	14.37% (22)	0.2740	0.945	0.274	0.835	0.861
					0.726	0.055	0.900	
Métodos basados en reglas	JRip	79.73% (122)	20.26% (31)	0.2907	0.846	0.274	0.819	0.776
					0.726	0.154	0.763	
Árboles de decisión	J48	81.69% (125)	18.30% (28)	0.2532	0.934	0.355	0.794	0.812
					0.645	0.066	0.870	
	Random Tree	**86.92% (133)**	**13.07% (20)**	**0.1373**	**0.912**	**0.194**	**0.874**	**0.869**
					0.806	**0.088**	**0.862**	

Tabla A.22 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por Safe_Level_SMOTE y el filtro supervisado Atribute Selection.

Clasificadores		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	BayesNet	54.87% (90)	45.12% (74)	0.4892	0.744	0.646	0.535	0.534
					0.354	0.256	0.580	
	Naive Bayes	66.46% (109)	33.53% (55)	0.4057	0.939	0.610	0.606	0.731
					0.390	0.061	0.865	
Redes neuronales	MLP (t)	64.63% (106)	35.36% (58)	0.4260	0.756	0.463	0.620	0.666
					0.537	0.244	0.688	
Métodos basados en instancias	IBK (K=13)	64.63% (106)	35.36% (58)	0.4548	0.902	0.610	0.597	0.646
					0.390	0.098	0.800	
	K Star	60.36% (99)	39.63% (65)	0.4387	0.756	0.549	0.579	0.636
					0.451	0.244	0.649	
Meta clasificadores	Atribute Select Classifier	64.63% (106)	35.36% (58)	0.4275	0.756	0.463	0.620	0.649
					0.537	0.244	0.688	
	Bagging (RIPTree)	67.68% (111)	32.31% (53)	0.4189	0.720	0.366	0.663	0.717
					0.634	0.280	0.693	
	Multiclass Classifier	65.24% (107)	34.75% (57)	0.3784	0.659	0.354	0.651	0.695
					0.646	0.341	0.654	
	Random Committee (RTree)	63.41% (104)	36.58% (60)	0.3849	0.683	0.415	0.622	0.659
					0.585	0.317	0.649	
	Random Subspace	**67.68% (111)**	**32.31% (53)**	**0.4575**	**0.793**	**0.439**	**0.644**	**0.693**
					0.561	**0.207**	**0.730**	
Métodos basados en reglas	**JRip**	**67.68% (111)**	**32.31% (53)**	**0.4197**	**0.695**	**0.341**	**0.671**	**0.672**
					0.659	**0.305**	**0.684**	
Árboles de decisión	J48	64.63% (106)	35.36% (58)	0.4247	0.756	0.463	0.620	0.656
					0.537	0.244	0.688	
	Random Forest	65.24% (107)	34.75% (57)	0.3784	0.659	0.354	0.651	0.696
					0.646	0.341	0.654	

Tabla A.23 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SMOTE_ENN y el filtro supervisado Atribute Selection.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	68%	32%	0.3897	0.688	0.327	0.660	0.701
					0.673	0.313	0.700	
	Naive Bayes	69%	31%	0.3661	0.813	0.423	0.639	0.798
					0.577	0.188	0.769	
Redes neuronales	MLP (o)	71%	29%	0.3549	0.750	0.327	0.679	0.725
					0.673	0.250	0.745	
Métodos basados en instancias	IBK (K=3)	71%	29%	0.3511	0.729	0.308	0.686	0.723
					0.692	0.271	0.735	
	K Star	74%	26%	0.3275	0.750	0.269	0.720	0.771
					0.731	0.250	0.760	
Meta clasificadores	**Atribute Select Classifier**	**81%**	**19%**	**0.1958**	**0.813**	**0.192**	**0.796**	**0.806**
					0.808	**0.188**	**0.824**	
	Bagging (RIPTree)	79%	21%	0.2983	0.792	0.212	0.776	0.835
					0.788	0.208	0.804	
	Multiclass Classifier	78%	22%	0.2200	0.771	0.212	0.771	0.780
					0.788	0.229	0.788	
	Random Committee (RandomForest)	**81%**	**19%**	**0.2854**	**0.792**	**0.173**	**0.809**	**0.829**
					0.827	**0.208**	**0.811**	
	Random Subspace	78%	22%	0.3407	0.688	0.135	0.825	0.822
					0.865	0.313	0.750	
Métodos basados en reglas	JRip	76%	24%	0.3560	0.792	0.269	0.731	0.691
					0.731	0.208	0.792	
Árboles de decisión	J48	77%	23%	0.2796	0.750	0.212	0.766	0.744
					0.788	0.250	0.774	
	Random Tree	78%	22%	0.2200	0.771	0.212	0.771	0.780
					0.788	0.229	0.788	

Tabla A.24 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SMOTE_RSB y el filtro supervisado Atribute Selection.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	64.63% (106)	35.36% (58)	0.4157	0.683	0.390	0.636	0.679
					0.610	0.317	0.658	
	Naive Bayes	59.14% (97)	40.85% (67)	0.4121	0.902	0.720	0.556	0.726
					0.280	0.098	0.742	
Redes neuronales	MLP (i)	68.29% (112)	31.70% (52)	0.3797	0.744	0.378	0.663	0.714
					0.622	0.256	0.708	
Métodos basados en instancias	IBK (K=1)	75% (123)	25% (41)	0.2533	0.793	0.293	0.730	0.747
					0.707	0.207	0.773	
	K Star	77.43% (127)	22.56% (37)	0.3038	0.866	0.317	0.732	0.855
					0.683	0.134	0.836	
Meta clasificadores	Atribute Select Classifier	79.87% (131)	20.12% (33)	0.2516	0.793	0.195	0.802	0.845
					0.805	0.207	0.795	
	Bagging (RandomTree)	79.87% (131)	20.12% (33)	0.2622	0.854	0.256	0.769	0.855
					0.744	0.146	0.836	
	Multiclass Classifier	79.87% (131)	20.12% (33)	0.2622	0.854	0.256	0.769	0.855
					0.744	0.146	0.836	
	Random Committee (RandomForest)	**80.48% (132)**	**19.51% (32)**	**0.2640**	**0.829**	**0.220**	**0.791**	**0.861**
					0.780	**0.171**	**0.821**	
	Random Subspace	77.43% (127)	22.56% (37)	0.3168	0.756	0.207	0.785	0.879
					0.793	0.244	0.765	
Métodos basados en reglas	JRip	70.12% (115)	29.87% (49)	0.3569	0.768	0.366	0.677	0.766
					0.634	0.232	0.732	
Árboles de decisión	J48	73.17% (120)	26.82% (44)	0.3125	0.793	0.329	0.707	0.741
					0.671	0.207	0.764	
	Random Forest	79.87% (131)	20.12% (33)	0.2622	0.854	0.256	0.769	0.855
					0.744	0.146	0.836	

Tabla A.25 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SMOTE_TL y el filtro supervisado Atribute Selection.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	71.31% (87)	28.68% (35)	0.3511	0.817	0.500	0.770	0.701
					0.500	0.183	0.571	
	Naive Bayes	72.95% (89)	27.04% (33)	0.2781	0.927	0.675	0.738	0.824
					0.325	0.073	0.684	
Redes neuronales	MLP (o)	75.40% (92)	24.59% (30)	0.3064	0.878	0.500	0.783	0.706
					0.500	0.122	0.667	
Métodos basados en instancias	IBK (K=1)	75.40% (92)	24.59% (30)	0.2504	0.854	0.450	0.795	0.721
					0.550	0.146	0.647	
	K Star	76.22% (93)	23.77% (29)	0.2697	0.890	0.500	0.785	0.794
					0.500	0.110	0.690	
Meta clasificadores	Atribute Select Classifier	76.22% (93)	23.77% (29)	0.2676	0.854	0.425	0.805	0.765
					0.575	0.146	0.657	
	Bagging (RandomForest)	77.86% (95)	22.13% (27)	0.2812	0.854	0.375	0.824	0.824
					0.625	0.146	0.676	
	Multiclass Classifier	77.04% (94)	22.95% (28)	0.3323	0.890	0.475	0.793	0.733
					0.525	0.110	0.700	
	Random Committee (RandomForest)	75.40% (92)	24.59% (30)	0.2704	0.829	0.400	0.810	0.795
					0.600	0.171	0.632	
	Random Subspace	**78.68% (96)**	**21.31% (26)**	**0.2826**	**0.890**	**0.425**	**0.811**	**0.814**
					0.575	**0.110**	**0.719**	
Métodos basados en reglas	JRip	73.77% (90)	26.22% (32)	0.3192	0.841	0.475	0.784	0.691
					0.525	0.159	0.618	
Árboles de decisión	J48	76.22% (93)	23.77% (29)	0.3037	0.866	0.450	0.798	0.695
					0.550	0.134	0.667	
	REPTree	77.04% (94)	22.95% (28)	0.3323	0.890	0.475	0.793	0.733
					0.525	0.110	0.700	

Tabla A.26 Resultados de algoritmos de clasificación aplicados al conjunto de datos editados por SMOTE y el filtro supervisado Atribute Selection.

Algoritmos de Clasificación		**Instancias correctas**	**Instancias incorrectas**	**Error M. absoluto**	**TPRate**	**FPRate**	**Precisión**	**ROC Área**
Redes bayesianas	Redes Bayesianas	68.90% (113)	31.09% (51)	0.3959	0.671	0.293	0.696	0.729
					0.707	0.329	0.682	
	Naive Bayes	62.19% (102)	37.80% (62)	0.3897	0.902	0.659	0.578	0.761
					0.341	0.098	0.778	
Redes neuronales	MLP (a)	71.34% (117)	28.65% (47)	0.3482	0.780	0.354	0.688	0.745
					0.646	0.220	0.746	
Métodos basados en instancias	IBK (K=1)	67.07% (110)	32.92% (54)	0.3315	0.720	0.378	0.656	0.690
					0.622	0.280	0.689	
	K Star	73.17% (120)	26.82% (44)	0.3393	0.817	0.354	0.698	0.776
					0.646	0.183	0.779	
Meta clasificadores	Atribute Select Classifier	75% (123)	25% (41)	0.2958	0.793	0.293	0.730	0.800
					0.707	0.207	0.773	
	Bagging (RandomForest)	78.04% (128)	21.95 (36)	0.3034	0.805	0.244	0.767	0.842
					0.756	0.195	0.795	
	Multiclass Classifier	75.60% (124)	24.39% (40)	0.2701	0.805	0.293	0.733	0.852
					0.707	0.195	0.784	
	R Committee (RandomForest)	78.04% (128)	21.95% (36)	0.2802	0.793	0.232	0.774	0.851
					0.768	0.207	0.788	
	Random Subspace	**81.70% (134)**	**18.29% (30)**	**0.3181**	**0.805**	**0.171**	**0.825**	**0.872**
					0.829	**0.195**	**0.810**	
Métodos basados en reglas	JRip	73.17% (120)	26.82% (44)	0.3366	0.732	0.268	0.732	0.764
Árboles de decisión	J48	71.95% (118)	28.04% (46)	0.3430	0.756	0.317	0.705	0.741
					0.683	0.244	0.737	
	Random Forest	75.60% (124)	24.39% (40)	0.2701	0.805	0.293	0.733	0.852
					0.707	0.195	0.784	