Universidad Central “Marta Abreu” de Las Villas

Facultad de Matemática, Física y Computación

MEJORAS DE LA CLASIFICACIÓN EN INTERACCIONES DE PROTEÍNAS DE LA *ARABIDOPSIS THALIANA* UTILIZANDO TÉCNICAS PARA CONJUNTOS DE DATOS DESBALANCEADOS.

Tesis en opción al título de Máster en Bioinformática y Biología Computacional

Autor

Lic. Ana C. Castillo Sánchez

Tutores

Dra. María del Carmen Chávez Cárdenas

Dra. María Matilde García Lorenzo

MSc. Leidys Cabrera Hernández

Santa Clara

2016

Agradecimientos

Dedicatoria

Resumen

En la actualidad, constituye un reto lograr la correcta clasificación de grandes conjuntos de datos usando aprendizaje automatizado. En problemas de Bioinformática es muy común tener grandes bases de casos, las cuales en algunas ocasiones son desbalanceadas, siendo la clase minoritaria casi siempre la de principal interés de investigación. Este trabajo se enmarca en la predicción de interacciones de proteínas, donde el conjunto de datos pertenece a la interacción de proteínas en la *Arabidopsis Thaliana* y precisamente la clase minoritaria es aquella que representa a las proteínas que interactúan. Como resultado de este trabajo se presenta la aplicación de diferentes técnicas para tratar el desbalance existente en los datos, tanto dirigidas a los atributos como a las instancias, además de los resultados que se alcanzan al aplicar diversos métodos de clasificación. La experimentación se realiza utilizando dos herramientas: WEKA (*Waikato Environment for Knowledge Analysis*) y KEEL (*Knowledge Extraction based on Evolutionary Learning*). Para hacer la selección del mejor clasificador se aplican las medidas más conocidas basadas en la matriz de confusión: razón de Verdaderos Positivos (VP), área bajo la Curva de Operación del Receptor (ROC), la Exactitud (Accuracy, en inglés), la medida-F (F-Measure, en inglés) y las curvas Precision_Recall (PRC).

Finalmente se demuestra que utilizando las técnicas de desbalance como pre-procesamiento de los datos, se logra mejorar los resultados de la clasificación en la base *Arabidopsis Thaliana,* respecto a los ya alcanzados hasta el momento.

Abstract

At present, it is challenging to correctly classify large datasets using automated learning. In Bioinformatics problems it is very common to have large case bases, which are sometimes unbalanced, with the minority class almost always being the main research interest. This work is framed in the prediction of protein interactions, where the dataset belongs to the interaction of proteins in Arabidopsis Thaliana and precisely the minority class is that which represents the interacting proteins. As a result of this work we present the application of different techniques to deal with the imbalance in the data, both for the attributes and the instances, as well as the results achieved by applying different classification methods. The experimentation is carried out using two tools: WEKA (Waikato Environment for Knowledge Analysis) and KEEL (Knowledge Extraction based on Evolutionary Learning). In order to make the selection of the best classifier, the most known measures based on the confusion matrix are applied: True Positive Ratio (VP), area under the Receiver Operating Curve (ROC), Accuracy, (F-Measure) and Precision_Recall (PRC) curves. Finally, it is shown that using the unbalance techniques as pre-processing of the data, it is possible to improve the results of the classification in the base Arabidopsis Thaliana, compared to those already achieved so far.

Tabla de Contenidos

INTRODUCCIÓN

La interacción de proteínas se refiere a la asociación de proteínas y el estudio de esas asociaciones desde las perspectivas de la bioquímica, transducción de señales y redes de interacción de proteínas.

Las interacciones de proteínas operan en casi todos los niveles de las funciones celulares; y la mayoría de las proteínas forman parten de algún tipo de complejo proteico en algún momento particular de la vida de una célula. No sería posible mencionar todos los casos conocidos en los cuales dos o más proteínas interaccionan entre sí; y mucho menos los casos particulares caracterizados y reportados hasta la fecha. Solo para generar una idea de cuán magníficas y complejas pueden ser dichas interacciones, se puede mencionar que las proteínas forman redes y complejas interacciones en procesos celulares como replicación, transcripción, y traducción del DNA; metabolismo del RNA; control del ciclo celular; metabolismo energético; transducción de señales; transporte de metabolitos, macromoléculas, y otras sustancias tanto en el interior, como en el intercambio con el medio externo; detoxificación, supervivencia frente a condiciones del entorno desfavorables; respuesta inmune; funciones estructurales como andamiaje celular (citoesqueleto y estructura de organelas), contracción muscular, ensamblado de membranas; etc (Phizicky and Fields, 1995).

El estudio de las interacciones entre proteínas requiere de la utilización de conocimientos y técnicas de diversas áreas como biología, genética, bioquímica, biofísica, etc. Comprender completamente la naturaleza de estos procesos es uno de los principales objetivos en la era de la proteómica. El conocimiento completo de las redes de interacción proteica no solo es importante para comprender los procesos celulares y sus implicancias desde un punto de vista biológico; sino también por los fines aplicados que de él se desprenden, como son el tratamiento de muchas enfermedades y desórdenes que tienen base en defectos de la interacción de proteínas, algunos de esos ejemplos son: diseño de drogas, tratamientos anti-virales y tratamientos contra otros patógenos, etc (Nooren and Thornton, 2003)

Las implicaciones que se desprenden de la interacción de proteínas no solo se aplican a las del cuerpo humano, sino que también cobran gran importancia aquellas que se realizan en

otros organismos como las plantas, ya que estas son capaces también de curar grandes enfermedades del cuerpo humano.

Junto a la identificación de nuevas proteínas en los vegetales, una tarea importante en esta rama concierne a la organización de los datos generados en bases de datos. Hace algunos años atrás se habían introducido dos bases de datos dedicadas a las proteínas vegetales, una de ellas el conjunto de datos de la *Arabidopsis Thailana*.

En el año 1907 el Dr. F. Laibach descubrió el número de cromosomas de la *Arabidopsis Thaliana*; sugiriendo el potencial para la experimentación genética, entre otras razones por la brevedad de su ciclo vital.

En el año 2000 se presentó por vez primera el mapa genético de la planta, convirtiéndola así en la primera planta cuyo genoma ha sido secuenciado y de gran interés en el estudio de la comunidad científica.

En las figuras correspondientes al Anexo 11 se pueden observar imagines de la planta en su habitad natural, estructura, así como su ubicación geográfica.

Por otro lado, en la actualidad constituye un reto lograr la correcta clasificación de grandes conjuntos de datos usando aprendizaje automatizado. En problemas de Bioinformática es muy común tener grandes bases de casos, en algunas ocasiones estas son desbalanceadas, siendo la clase minoritaria casi siempre la de principal interés de investigación.

En la clasificación supervisada frecuentemente aparecen problemas donde la cantidad de objetos de una clase es significativamente mayor que la cantidad de objetos de otra clase. A este tipo de problemas se le llaman problemas con clases desbalanceadas. Comúnmente, la clase minoritaria representa el concepto más importante que hay que aprender y es difícil identificarlo, ya que podría estar asociado a casos excepcionales pero significativos, o porque la adquisición de estos datos es muy difícil.

El problema de desbalance es complejo, y no solamente depende de la proporción que existe entre el número de instancias de cada clase, dicho problema es conocido también como “desbalance entre clases”. La complejidad de los datos juega un papel importante en este tipo de problemas, la falta de datos representativos en algunas regiones del espacio de entrada o la existencia de subconceptos. Cuando dentro de un problema de clasificación existen

subconceptos que contienen pocas instancias, se presenta lo que se conoce como el "desbalance al interior de las clases.

Problema de investigación

El conjunto de datos de interacciones de proteínas de la *Arabidopsis Thaliana* no presenta casos muy bien distinguidos en cada clase, por lo que el clasificador aprende con dudas. A su vez, el conjunto de datos utilizado es desbalanceado por lo que los resultados que se alcanzan en las clasificaciones de investigaciones actuales no son buenos.

Objetivo general

Mejorar los resultados de la clasificación en la predicción de las interacciones de proteínas en un caso de estudio de la *Arabidopsis Thaliana,* mediante la aplicación de técnicas para mejorar el desbalance.

El objetivo general se puede desglosar en los siguientes objetivos específicos.

Objetivos específicos

1. Identificar técnicas de pre-procesamiento de datos para mejorar los resultados obtenidos al aplicar métodos de clasificación en conjuntos de datos desbalanceados.
2. Analizar métodos de clasificación a emplear y medidas para evaluar los resultados de la clasificación en bases de datos desbalanceadas.
3. Evaluar el desempeño de los clasificadores luego de aplicar las técnicas de pre-procesamiento de datos empleadas.
4. Comparar la influencia de distintas técnicas de pre-procesado en las métricas sensibles al desbalance de clases.

A continuación, se muestran las preguntas de investigación.

Preguntas de investigación

1. ¿Qué técnicas de pre-procesamiento de datos utilizar cuando el conjunto de datos es desbalanceado?
2. ¿Qué medidas emplear para evaluar los resultados de un clasificador luego de aplicar técnicas de pre-procesamiento para el desbalance?
3. ¿Cómo diseñar correctamente los experimentos?

Por todo lo anterior se planteó la siguiente hipótesis de investigación:

Hipótesis de investigación

La aplicación de técnicas para el desbalance de los datos mejora los resultados en la predicción de las interacciones de proteínas en un caso de estudio de la *Arabidopsis Thaliana*.

El presente trabajo se encuentra estructurado de la siguiente manera: introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos.

Capítulo 1. Técnicas de pre-procesamiento para el desbalance en los datos y algoritmos de clasificación.

En este capítulo se realiza una presentación detallada de las técnicas de pre-procesamiento de datos dirigidas a los atributos e instancias. 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.

Capítulo 2. Interacción de proteínas en la *Arabidopsis thaliana* y herramientas que permiten aplicar técnicas para el desbalance y la clasificación.

En este capítulo se abordan elementos de las interacciones de proteínas y las herramientas que permiten aplicar técnicas de pre-procesamiento a los datos, así como su clasificación. Se describe el conjunto de datos a estudiar y los resultados alcanzados hasta ahora en el mismo. Además, se muestran los detalles de las principales herramientas que permiten trabajar con métodos de aprendizaje automatizado.

Capítulo 3. Aplicación de las técnicas para el desbalance y resultados de la clasificación.

El tercer capítulo muestra el diseño de todos los experimentos realizados y sus resultados, incluyéndose las técnicas para mejorar el desbalance, así como los resultados finales de la clasificación en la base *Arabidopsis thaliana*.

1. TÉCNICAS DE PRE-PROCESAMIENTO PARA EL DESBALANCE EN LOS DATOS Y ALGORITMOS DE CLASIFICACIÓN.

CAPÍTULO 1. TÉCNICAS DE PRE-PROCESAMIENTO PARA EL DESBALACE EN LOS DATOS Y ALGORITMOS DE CLASIFICACIÓN.

En este capítulo se analizan las diferentes técnicas de pre-procesamiento para el desbalance en los datos, así como los diferentes algoritmos de clasificación que serán aplicados. Además, se abordan las medidas que permiten la validación de los algoritmos de clasificación.

1.1 Desbalance en los datos

El problema de desbalance es complejo, y no solamente depende de la proporción que existe entre el número de instancias de cada clase, dicho problema es conocido como "desbalance entre clases", como se mencionaba anteriormente en la introducción. La complejidad de los datos juega un papel importante en este tipo de problemas, la falta de datos representativos en algunas regiones del espacio de entrada o la existencia de subconceptos. Cuando dentro de un problema de clasificación existen subconceptos que contienen pocas instancias, se presenta lo que se conoce como el "desbalance al interior de las clases" (Barandela et al., 2003), (Goya et al., 2015).

La comunidad científica internacional ha trazado tres estrategias fundamentales para mitigar las dificultades que aparecen en la clasificación supervisada al trabajar con bases de datos con clases desbalanceadas.(López et al., 2014a), (López et al., 2013), (Krawczyk et al., 2014), (González et al., 2014). Estas estrategias se agrupan en las siguientes categorías:

Nivel de Datos. Re-muestreo de la base de datos para balancear las clases. Consiste en alcanzar un balance entre las clases mediante la eliminación de objetos de la clase mayoritaria (sub-muestreo)(López et al., 2013), (Albisua et al., 2013), (Li et al., 2010); o la inclusión de objetos en la clase minoritaria (sobre-muestreo) (Menardi and Torelli, 2014) (López et al., 2014b), (Soda, 2011), (Soda, 2011), (Weiss et al., 2007), (Luengo et al., 2011), (Chawla, 2009), (Chawla et al., 2002a). El sub-muestreo puede excluir algunos objetos representativos o valiosos para entrenar el clasificador. En cambio, el sobre-muestreo incluye objetos artificiales que pueden sobre-entrenar al clasificador.

Modificación de Algoritmos. Los clasificadores existentes son modificados para fortalecer su predicción con respecto a la clase minoritaria. Depende mucho de la

naturaleza del clasificador y la mayoría son modificados para resolver un problema específico (Rodda and Mogalla, 2011). (Liu et al., 2010), (Lenca et al., 2008) 15, , 17, 18

Matrices de costo. Éstas permiten asignarle diferentes costos a los distintos tipos de errores que comete un clasificador. De esta forma, estos pesos pueden utilizarse para priorizar la clase minoritaria. Desafortunadamente, es difícil para un especialista determinar el costo de los diferentes errores de clasificación. Por ello, la matriz de costo en la mayoría de las bases de datos es desconocida (Krawczyk et al., 2014), (Lomax and Vadera, 2013), (Jackowski et al., 2012), (Wei et al., 2013), (Min and Zhu, 2012), (Freitas, 2011), (Sun et al., 2007).

1.1.1 Técnicas de pre-procesamiento de datos dirigidas a los atributos

"El Pre-procesamiento de Datos" / "La Preparación de Datos" engloba a todas aquellas técnicas de análisis de datos que permite mejorar la calidad de un conjunto de datos de modo que las técnicas de extracción de conocimiento puedan obtener mayor y mejor información (mejor porcentaje de clasificación, reglas con más completitud, etc.) (Zhang et al., 2003).

El propósito del pre-procesamiento de datos es principalmente corregir las inconsistencias de los datos que serán la base de análisis en procesos de minería de datos. Con el pre-procesamiento de datos se pretende conservar la coherencia de los datos que van a ser utilizados en tareas de análisis o descubrimiento de conocimiento (Hernández and Rodríguez, 2013).

El objetivo principal de cualquier análisis de datos, antes que el descubrimiento de conocimiento, es utilizar los resultados para resolver problemas o para la toma de decisiones.

En la mayoría de los casos, las imperfecciones con los datos sólo son notadas cuando se inicia el análisis de los datos. Para disminuir tiempo y costos es importante preparar los datos para dicho análisis; en esta línea, ya existen diversas técnicas, las cuales pueden mejorar la calidad de los datos, ayudando a mejorar la precisión y eficiencia de los procesos de análisis de datos, de ahí que el pre-procesamiento de datos se convierta en un paso preliminar importante. Detectando anomalías, corrigiéndolas a tiempo y

reduciendo los datos que serán analizados conlleva a que la toma de decisiones sea mucho más eficaz.

El pre-procesamiento es una tarea necesaria para la preparación de los datos que serán utilizados en análisis de datos. La justificación de este proceso preliminar al análisis de datos, generalmente, radica en que los datos vienen con ruido por diferentes razones, entre las cuales se encuentran (Han et al., 2011):

- Datos incompletos: valores faltantes para algunos atributos o sólo se tienen los datos agregados y no se cuenta con el detalle.
- Ruido: errores en los datos, por ejemplo, manejar valores negativos para un atributo que maneja salarios.
- Inconsistencias: contiene discrepancias en los datos.

En el mejor de los casos, el algoritmo va a funcionar, pero los resultados ofrecidos no tendrán sentido o no será considerado como un conocimiento preciso. Por lo tanto, ¿cuáles son las cuestiones básicas que deben ser resueltas en la preparación de datos? Aquí, proporcionamos una lista de preguntas acompañado de las respuestas correctas que implica cada tipo de proceso que pertenece a la familia de las técnicas de preparación de datos:

- ¿Cómo se limpia la seguridad de los datos? – Limpieza de Datos.
- ¿Cómo hacer que se proporcionen datos precisos? Transformación de los datos.
- ¿Cómo incorporar y ajustar los datos? – Integración de los Datos.
- ¿Cómo manejo su falta? –Imputación de valores perdidos.
- ¿Cómo detectar y tratar el ruido? - Identificación de Ruido.
- ¿Cómo unificar datos y tratar datos de escala? – Normalización de Datos.

A continuación, se detallan algunas de ellas.

Limpieza de Datos

La limpieza de datos incluye las operaciones que permiten corregir los datos erróneos, filtrar algunos datos incorrectos fuera del conjunto de datos y reducir el detalle

innecesario de datos. Es un concepto general que se comprende o que se solapa con otras técnicas de preparación de datos conocidas, mostradas en la Figura 1.1.

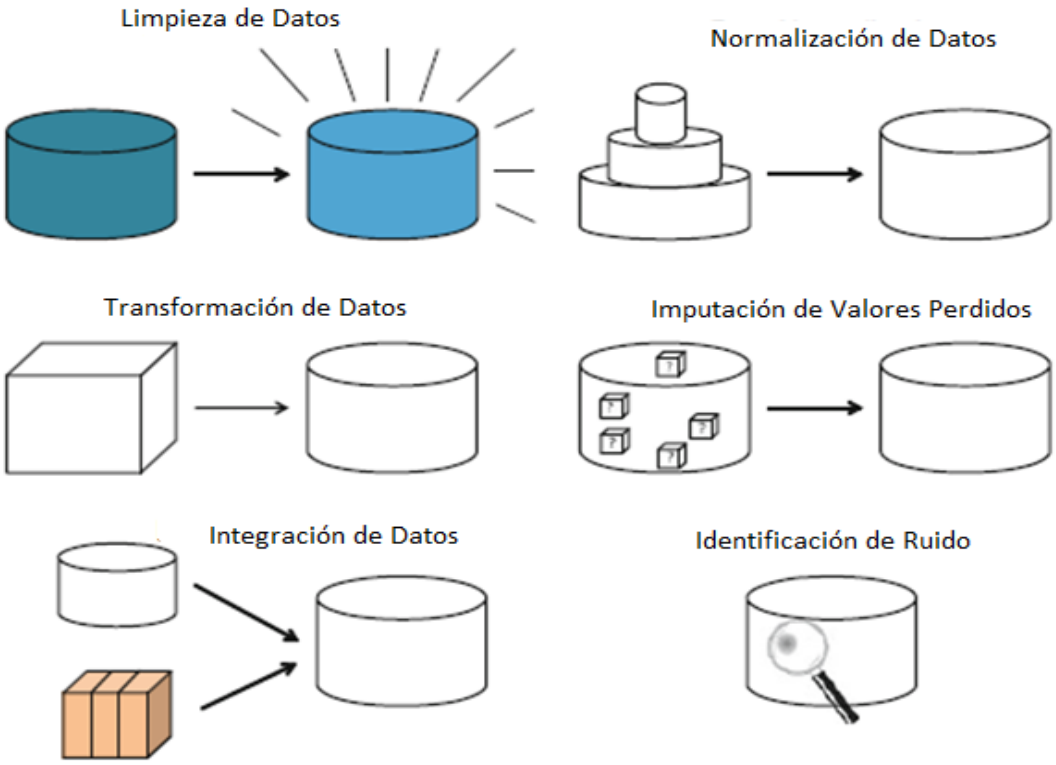


Figura 1.1 Técnicas de preparación de los datos (García et al., 2015).

Transformación de datos

En esta etapa de pre-procesamiento los datos se convierten o consolidan, de modo que el resultado del proceso de minería se podría aplicar o puede ser más eficiente. Las sub-tareas dentro de la transformación de datos son: el alisado, la construcción característica, agregación o resumen de los datos, la normalización, la discretización y generalización. La mayoría de ellas se segregarán como tareas independientes.

Integración de Datos

La integración de datos se compone de la fusión de datos de múltiples almacenes de datos. El empleo de este proceso debe ser cuidadoso con el fin de evitar redundancias e inconsistencias en el conjunto de datos resultante. Las operaciones típicas realizadas dentro de la integración de datos son la identificación y unificación de las variables y los

dominios, el análisis de correlación de atributo, la duplicación de tuplas y la detección de conflictos en los valores de datos de diferentes fuentes.

Imputación de valores perdidos

La imputación de valores perdidos es una forma de limpieza de datos, donde el propósito es llenar las variables que contienen valores perdidos con algunos datos intuitivos. En la mayoría de los casos, la adición de una estimación razonable de un valor de datos adecuado es mejor que dejarlo en blanco (Luengo et al., 2012).

Identificación de ruido

La identificación de ruido, incluido como un paso de limpieza de datos y también conocido como el suavizado en la transformación de datos, tiene como principal objetivo detectar los errores aleatorios o variaciones en una medida variable. Tenga en cuenta que se refiere a la detección de ruido en lugar de la eliminación de ruido, el cual está más relacionado con la tarea de la reducción de datos (Sáez et al., 2013).

Normalización

Una vez que las características han sido seleccionadas, estas pueden tener un rango diverso de valores. 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. Se pasa de la variable Xn a la variable $\hat{x}_n$y así transformando cada valor con las siguientes fórmulas:

$$\hat{x}_{i,n} = \frac{x_{in} - \overline{x_n}}{\sigma_n} \qquad 1.1$$

donde:

$$\hat{x} = \frac{1}{m}\sum_{i=1}^{m} x_{i,n} \qquad 1.2$$

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

$\hat{x}$ es la medida de cada variable x_n, σ_n es la desviación estándar y m es la talla del conjunto de datos (Valdovinos, 2006).

La normalización de los datos intenta dar un peso igual a todos los atributos y es particularmente útil en los métodos de aprendizaje estadísticos (García et al., 2015).

La normalización es empleada cuando se tienen atributos con órdenes de magnitud muy diferentes. Gracias a la normalización se evita que los atributos con valores más altos ganen un peso significativamente más importante en el modelo final que aquellos con valores más bajos.

Discretización

Este proceso transforma datos cuantitativos en datos cualitativos, es decir, atributos numérico en atributos discretos o nominales con un número finito de intervalos (Garcia et al., 2013). Es muy útil cuando se trabaja con atributos numéricos, puesto que muchas herramientas de análisis requieren datos simbólicos, y por tanto se necesita aplicar esta transformación antes. Es necesario cuando se quiere hacer una clasificación sobre un atributo numérico. Este proceso transforma los atributos numéricos seleccionados en atributos simbólicos, con una serie de etiquetas resultantes de dividir la amplitud total del atributo en intervalos, con diferentes opciones para seleccionar los límites. Por defecto, se divide la amplitud del intervalo en tantas "cajas" como se indique en bins (por defecto 10), todas ellas de la misma amplitud. El proceso de aprendizaje frecuentemente es menos eficiente y menos efectivo cuando los datos tienen solamente variables cuantitativas.

Al transformar los valores numéricos en valores categóricos se reduce el número de posibles valores. La discretización suaviza el efecto del ruido y permite modelos más simples; y, por lo tanto, menos propensos al sobreajuste.

Muchos algoritmos de clasificación requieren que las variables de clasificación sean discretizadas, ya sea porque necesitan valores nominales, o con el fin de reducir el costo computacional o para acelerar el proceso de inducción (Dougherty et al., 1995).

Hay que destacar que la discretización es en realidad una técnica de pre-procesamiento de datos híbrido, involucrando tanto a las tareas de preparación de datos como a las de reducción de datos. Algunas fuentes incluyen la discretización en la categoría de

transformación de datos y otras fuentes la consideran como un proceso de reducción de datos (Lin et al., 2004).

En (Dougherty et al., 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 o local, supervisado o no supervisado, y estático o 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.

Las técnicas aplicadas en nuestro trabajo fueron específicamente la normalización y la discretización debido a las características propias que ofrece el conjunto de datos de nuestro caso de estudio.

1.1.2 Técnicas de pre-procesamiento de datos dirigidas a las instancias

Todo aprendizaje supervisado es más eficiente y eficaz si la base de entrenamiento tiene casos muy bien distinguidos en cada clase, es decir, se conocen positivos sin duda y negativos sin duda.

La base de entrenamiento de interacciones de proteínas en la *Arabidopsis thaliana*, que es nuestro caso de estudio, no presenta estas características idóneas. De hecho, de los denominados negativos, no se está absolutamente seguros que lo sean, se conoce que no están reportados como positivos. Entonces el aprendizaje puede estar sesgado y hasta es natural que este grupo tenga el mejor porciento de clasificación, simplemente porque el clasificador aprende con dudas.

El aprendizaje a partir de datos no balanceados es uno de los desafíos que actualmente está enfrentando el aprendizaje automático, debido al mal funcionamiento de los algoritmos frente a conjuntos de éste tipo. La ocurrencia de sucesos poco frecuentes ha dado lugar a que exista

una desproporción considerable entre el número de ejemplos en cada clase, lo que se conoce como *clases no balanceadas* o *desbalanceadas*.

El problema del desbalanceo de los datos consiste en la predominancia de ciertos valores en los datos y la escasez o ausencia de otros que dificulta o impide la extracción de información.

Los clasificadores logran muy buenas precisiones con la clase más representada (mayoritaria), mientras que en la menos representada (minoritaria) ocurre todo lo contrario. En los no balanceados el conocimiento más novedoso suele residir en los datos menos representados, sin embargo, muchos clasificadores pueden considerarlos como rarezas o ruido, pues los mismos no tienen en cuenta la distribución de los datos, únicamente se centran en los resultados de las medidas globales.

Como resultado de las investigaciones, han surgido numerosas técnicas para tratar el desbalance, las mismas han sido agrupadas en 2 categorías:

1. Las que están al nivel de los algoritmos de aprendizaje, las cuales no modifican la distribución de los datos, se centran en el ajuste de costo por clase, ajuste de la estimación de probabilidad en las hojas del árbol de decisión, aprender de una sola clase, entre otros.
2. Las que modifican la distribución de los datos, conocidas también como re-muestreo, es decir, pueden reducir la clase mayoritaria eliminando ejemplos, a través de la técnica conocida como *under-sampling*, o pueden aumentar la clase minoritaria creando nuevos ejemplos con la técnica conocida como *over-sampling*.

A continuación, se hace énfasis en la segunda categoría que es la aplicada en nuestro trabajo.

1.1.2.1 Métodos de Re-muestreo

Los métodos de re-muestreo, también conocidos como métodos de pre-procesado de conjuntos de entrenamiento, pueden ser divididos en 3 grandes grupos: los que eliminan instancias de la clase mayoritaria (*under-sampling*), los que generan nuevas instancias de la clase minoritaria (*over-sampling*) o la hibridación de ambas. A continuación, son descritos algunos de los métodos más conocidos.

1.1.2.1.1 Métodos de Over-Sampling

Entre las estrategias más conocidas para la generación de nuevas instancias con el fin de balancear conjuntos de entrenamiento se encuentra SMOTE (*Synthetic Minority Over-Sampling TEchnique)*, propuesto por Chawla y colaboradores en (Chawla et al., 2002b), éste algoritmo para cada ejemplo de la clase minoritaria introduce ejemplos sintéticos en la línea que une al elemento con sus 5 vecinos más cercanos. Sin embargo, esta estrategia presenta el problema de que puede introducir ejemplos de la clase minoritaria en el área de la mayoritaria, es decir, crear malos ejemplos que posteriormente pudieran confundir a los clasificadores.

En el 2005 se realizaron dos nuevas propuestas de SMOTE en (Han et al., 2005), se presenta borderline-SMOTE1 y el borderline-SMOTE2, ambos generan instancias en la frontera entre las clases, es decir, son etiquetados como "peligrosos" los elementos de la minoritaria situados muy cercanos a la mayoritaria y a partir de ellos y sus vecinos se comienzan a generar las nuevas instancias, lográndose muy buenos resultados.

En el 2006 se propone el AHC (*Agglomerative Hierarchical Clustering Based)* en (Cohen et al., 2006), donde a partir de la creación de grupos enlazados usando un algoritmo jerárquico aglomerativo de agrupamiento, se seleccionan los centroides de cada grupo como un nuevo elemento sintético y finalmente son insertados en el conjunto original.

Otra de las estrategias reportadas en la literatura, perteneciente a los métodos de *Over-Sampling* es el *Resampling*, el cual duplica al azar instancias de la clase minoritaria (Moreno et al., 2009).

1.1.2.1.2 Métodos de Under-Sampling

Esta técnica corresponde a un método no heurístico que tiene como objetivo equilibrar la distribución de las clases a través de la "eliminación aleatoria" de ejemplos de la clase mayoritaria.

Dentro de los métodos más clásicos para realizar *under-sampling* se encuentran:

- El RU (*Random Under-Sampling*), donde se selecciona de manera aleatoria instancias de la clase mayoritaria para ser eliminados sin reemplazamiento hasta que ambas clases queden balanceadas.

- El NCR (*Neighborhood Cleaning Rule*), propuesto en (Laurikkala, 2001) , donde para cada elemento del conjunto de entrenamiento se buscan sus tres vecinos más cercanos, si el elemento seleccionado es de la clase mayoritaria y los tres vecinos son de la minoritaria, entonces se elimina el elemento; si el elemento pertenece a la clase minoritaria entonces se eliminan los vecinos que sean de la mayoritaria.
- El *Tomek Links* (Tomek, 1976), donde se eliminan sólo instancias de la clase mayoritaria que sean redundantes o que se encuentren muy cerca de instancias de la clase minoritaria.
- El *Wilson Editing* (Witten, 2005). También conocido como ENN (*Editing Nearest Neighbor*), elimina aquellas instancias donde la mayoría de sus vecinos pertenecen a otra clase.
- El *Condensed Nearest Neighbor Rule.* Regla condensada del vecino más cercano (CNN) (Hart, 1968), se utiliza para encontrar un subconjunto consistente de ejemplos. Un subconjunto $\hat{E} \subseteq E$ es consistente con E si se utiliza a 1- vecino más cercano (N*earest Neighbor* NN), $\hat{E}$ clasifica correctamente los ejemplos en E. El algoritmo para crear un subconjunto $\hat{E}$ de E , como un método *under-sampling,* es el siguiente (Kubat and Matwin, 1997): En primer lugar, aleatoriamente dibuja un ejemplo de la clase mayoritaria y pone todos los ejemplos de la clase minoritaria en $\hat{E}$. A continuación, utiliza un 1-NN en los ejemplos en $\hat{E}$ para clasificar los ejemplos de E. Todos los ejemplos mal clasificados de E se trasladan a $\hat{E}$. Es importante señalar que este procedimiento no encuentra el subconjunto más pequeño consistente de E . La idea detrás de esta implementación de un subconjunto coherente es eliminar los ejemplos de la clase mayoritaria que son distantes de la frontera de decisión, ya que este tipo de ejemplos pueden ser considerados menos relevantes para el aprendizaje.
- El *One-Sided Selection* (OSS) propuesto en (Kubat and Matwin, 1997), es un método *under-sampling* resultante de la aplicación de *Tomek links*, seguido de la aplicación de CNN. *Tomek links* se utiliza como un método de *under-sampling* y elimina ejemplos ruidosos y cercarnos a la frontera de la clase mayoritaria. Los ejemplos cercarnos a la frontera pueden ser considerados "inseguros", ya que una pequeña cantidad de ruido puede hacer caer en el lado equivocado de la frontera

de decisión. CNN tiene como objetivo eliminar ejemplos de la clase mayoritaria que están distantes de la frontera de decisión.

1.1.2.1.3 Métodos híbridos

A pesar de que tanto el *Over-Sampling* como el *Under-Sampling* logran buenos resultados por separado, muchos investigadores del área han obtenido magníficos resultados hibridando ambos métodos, de estos se pueden citar:

- SMOTE- TomekHybrid: inicialmente se realiza el *over-sampling* con la clase minoritaria y luego se aplica el Tomek Link a ambas clases (Batista et al., 2004b)
- CNN + Tomek links: es similar a la selección de un solo lado (one-sided selection), pero el método para encontrar el subconjunto consistente se aplica antes del Tomek Links. Nuestro objetivo es verificar su competitividad con el OSS. A medida que la búsqueda de Tomek Links es computacionalmente exigente, sería computacionalmente más barato si se realizó en unos datos de conjuntos reducidos.
- SMOTE + ENN, la motivación detrás de este método es similar al SMOTE + Tomek links. ENN tiende a eliminar más ejemplos que los que hace Tomek links, por lo que se espera que proporcionará una mayor depuración de los datos en profundidad. A diferencia de NCL, el cual es un método *under-sampling*, ENN se utiliza para eliminar tanto ejemplos como clases. Por tanto, ningún ejemplo que esté mal clasificado por sus tres vecinos más cercanos se retira del conjunto de entrenamiento (Batista et al., 2004b).
- Smote RSB propuesto por (Ramentol et al., 2012), es un nuevo método híbrido para procesamiento previo de conjuntos de datos desbalanceados a través de la construcción de nuevas muestras, utilizando SMOTE junto con la aplicación de una técnica de edición basada en el conjunto aproximado. La teoría y la aproximación inferior de un subconjunto.

1.2 Técnicas de clasificación

La clasificación es una tarea de la minería de datos que permite predecir el valor de una variable categórica (objetivo o clase) construyendo un modelo basado en uno o más variables numéricas o categóricas (predictores o atributos).

Mediante la clasificación se analiza un conjunto de datos y se construye un modelo de objetos para cada clase. Dicho modelo puede representarse con árboles de decisión o con reglas de clasificación que muestren las características de los datos. El modelo puede ser usado para mayor comprensión de los datos existentes y para la clasificación de datos futuros.

Los métodos matemáticos de clasificación pertenecen al llamado "aprendizaje supervisado". Ellos están caracterizados fundamentalmente porque se conoce la información acerca de la clase a la que pertenece cada uno de los objetos. Cuando la variable de decisión, función o hipótesis a predecir es continua, a los algoritmos relacionados con los problemas supervisados se les conoce como métodos de regresión. Si por el contrario la variable de decisión, función o hipótesis es discreta, ellos se conocen como métodos de clasificación o simplemente clasificadores. Este trabajo se centra en estos últimos.

En un problema de clasificación se tienen un conjunto de objetos, elementos, instancias u observaciones divididos en clases o etiquetados. Dado un elemento del conjunto, un especialista le asigna una clase de acuerdo a los rasgos, características o variables que lo describen. Esta relación entre los descriptores y la clase puede estar dada por un conjunto de reglas. La mayoría de las veces este conjunto de reglas no se conoce y la única información que se tiene es el conjunto de ejemplos etiquetados, de forma tal que las etiquetas representan las clases.

De manera general, se puede decir que 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 de gran cantidad de formas. Su definición dependerá del problema en particular (Cabrera et al., 2013).

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

Entre los métodos de clasificación más usados están los árboles de decisión, redes bayesianas, máquinas de soporte vectorial, redes neuronales artificiales, algoritmos

perezosos y multiclasificadores, pero estos no son los únicos. A continuación, se presenta una breve descripción de cada uno de ellos.

1.2.1 Algoritmos de árboles de decisión

Un árbol de decisión es un conjunto de condiciones o reglas organizadas en una estructura jerárquica, de tal manera que la decisión final se puede determinar siguiendo las condiciones que se cumplen desde la raíz hasta alguna de sus hojas. Es un modelo de predicción utilizado en el ámbito de la inteligencia artificial, dada una base de datos se construyen estos 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.

La elaboración de árboles de decisión se ha utilizado como método de minería de datos para representar los eventos que surgen a partir de una decisión. Los árboles de decisión han sido utilizados principalmente para explicar procedimientos médicos, legales, matemáticos, estratégicos, entre otros.

Un árbol de decisión tiene unas entradas las cuales pueden ser un objeto o una situación descrita por medio de un conjunto de atributos y a partir de esto devuelve una respuesta, la cual es una decisión que es tomada a partir de las entradas.

El aprendizaje usando árboles de decisión es un método para aproximar funciones. Los árboles de decisión pueden también representarse como conjuntos de reglas IF-THEN. Cada nodo interior del árbol especifica una prueba de algún atributo y las hojas son las clases en las cuales se clasifican las instancias, cada rama descendiente de un nodo interior corresponde a un valor posible del atributo probado en ese nodo. Un árbol de decisión representa una disyunción de conjunciones sobre los valores de los atributos. Así, cada rama, de la raíz a un nodo hoja, corresponde a una conjunción de atributos y el árbol en sí, a una disyunción de estas conjunciones.

J48 o C4.5

Este algoritmo es una versión optimizada del algoritmo de clasificación mediante árboles de decisión denominado C4.5 (Quinlan, 1993). Este algoritmo genera un árbol de

decisiones probabilístico que puede ser fácilmente interpretado por expertos y transformado en claras y comprensibles reglas.

Se trata de un clasificador sencillo, de bajo costo computacional y de fácil interpretabilidad. Los árboles de decisión parten de un nodo raíz donde se encuentran todos los patrones. Se selecciona la característica que maximiza el decremento de la impureza y a partir de dicha característica “abrir" el árbol generando nodos intermedios. Se repite el proceso hasta llegar a cumplir con el criterio de parada establecido. A su vez, el último nodo del árbol se denomina hoja. Al patrón que llegue a dicha hoja se le etiquetará con la etiqueta que corresponde a esa hoja.

Es un algoritmo de inducción que genera una estructura de reglas o árbol a partir de subconjuntos (ventanas) de casos extraídos del conjunto total de datos de “entrenamiento”. El algoritmo genera una estructura de reglas y evalúa su “bondad” usando criterios que miden la precisión en la clasificación de los casos. Emplea dos criterios principales para dirigir el proceso, dados por:

1. Calcular el valor de la información proporcionada por una regla candidata (o rama del árbol), con una rutina que se llama “info”.

2. Calcular la mejora global que proporciona una regla/rama usando una rutina que se llama gain (beneficio).

Con estos dos criterios se puede calcular una especie de valor de coste/beneficio en cada ciclo del proceso, que le sirve para decidir si crear, por ejemplo, dos nuevas reglas, o si es mejor agrupar los casos de una sola.

El algoritmo realiza el proceso de los datos en sucesivos ciclos. En cada ciclo se incrementa el tamaño de la “ventana” de proceso en un porcentaje determinado respecto al conjunto total. El objetivo es tener reglas a partir de la ventana que clasifiquen correctamente a un número cada vez mayor de casos en el conjunto total.

Cada ciclo de proceso emplea como punto de partida los resultados conseguidos por el ciclo anterior. En cada ciclo de proceso se ejecuta un sub-modelo contra los casos restantes que no están incluidos en la ventana. De esta forma se calcula la precisión del modelo respecto a la totalidad de datos.

Este algoritmo usa puntos de corte e introduce varias medidas para evitar el sobre entrenamiento, en particular los criterios de parada de la división y de poda del árbol. El algoritmo C4.5 se basa en la utilización del criterio radio de ganancia (gain ratio). De esta manera se consigue evitar que las variables con mayor número de posibles valores salgan beneficiadas en la selección. Además, el algoritmo C4.5 incorpora una poda del árbol de clasificación una vez que este ha sido inducido. La poda está basada en la aplicación de una prueba de hipótesis que trata de responder a la pregunta de si merece la pena expandir o no una determinada rama .(Quinlan, 1996).

1.2.2 Algoritmos basados en Redes Bayesianas

Una red bayesiana representa la distribución de probabilidades conjunta para un conjunto de variables (Mitchell, 1997). Las redes bayesianas son una representación compacta de una distribución de probabilidad multivariante. Formalmente, una red bayesiana es un grafo dirigido acíclico donde cada nodo representa una variable aleatoria y las dependencias entre las variables quedan codificadas en la propia estructura del grafo según el criterio de d-separación (Pearl, 2014). Asociada a cada nodo de la red hay una distribución de probabilidad condicionada a los padres de ese nodo, de manera que la distribución conjunta factoriza como el producto de las distribuciones condicionadas asociadas a los nodos de la red.

La inclusión de las relaciones de independencia en la propia estructura de la red, hace de las redes bayesianas una buena herramienta para representar conocimiento de forma compacta pues se reduce el número de parámetros necesarios. Estas relaciones simplifican la representación de la función de probabilidad conjunta como el producto de las funciones de probabilidad condicional de cada variable (Chávez, 2008)

La topología o estructura de la red brinda información sobre las dependencias probabilísticas entre las variables, pero también sobre las independencias condicionales de una variable (o conjunto de variables) dada otra u otras variables; dichas independencias simplifican la representación del conocimiento (menos parámetros) y el razonamiento (propagación de las probabilidades) (Felgaer, 2004).

El obtener una red bayesiana a partir de datos es un proceso de aprendizaje que se divide en dos etapas: el aprendizaje estructural y el aprendizaje paramétrico. La primera de ellas

consiste en obtener la estructura de la red bayesiana, es decir, las relaciones de dependencia e independencia entre las variables involucradas. La segunda etapa tiene como finalidad obtener las probabilidades a priori y condicionales requeridas a partir de una estructura dada.

Las redes bayesianas pueden realizar la tarea de clasificación -caso particular de predicción- que se caracteriza por tener una sola de las variables de la base de datos (clasificador) que se desea predecir, mientras que todas las otras son los datos propios del caso que se desea clasificar. Pueden existir una gran cantidad de variables en la base de datos, algunas de las cuales estén directamente relacionadas con la variable clasificadora pero también otras variables que tienen una influencia directa sobre dicha clase.

1.2.2.1 BayesNet

BayesNet es un método basado en "score" y búsqueda, en el cual la medida de bondad (verosimilitud, entropía, porcentaje de bien clasificados) de una estructura particular es definida, y ahora se lleva a cabo un procedimiento de búsqueda a lo largo del espacio de todas posibles estructuras de redes bayesianas (Pearl, 1998).

El clasificador usa diferentes tipos de algoritmos para el aprendizaje automatizado de la red bayesiana a usar en el proceso de clasificación, a los que denomina algoritmos de búsqueda, cuya tarea principal es definir la estructura de la red bayesiana (Chávez, 2008).

El aprendizaje Bayesiano se basa en que existe una distribución de probabilidad seguida por los datos. Este algoritmo tiene buena representación gráfica cuando la distribución de nodos es relativamente sencilla y puede sobrellevar fácilmente el problema de la replicación de nodos presente en los árboles de inducción. *BayesNet* es una extensión de *NaiveBayes* (López, 2007*)*.

Este algoritmo utiliza varios algoritmos de búsqueda y medidas cualitativas que determinan la red o estructura gráfica de las probabilidades de las instancias dadas. A continuación, se describen algunos de estos algoritmos:

Algoritmo BayesChaid

Según (Chávez, 2008) el algoritmo parte de ideas propias de la técnica de segmentación de CHAID (ChiSquare Automatic Interaction Detector) con adaptaciones para la generación de topologías más complejas que se ajustan a Redes Bayesianas.

La función principal del algoritmo es realizar una búsqueda completa de interacciones realizadas entre las variables, tanto a lo ancho como a profundidad en el árbol de posibles interacciones (Chávez, 2008).

El algoritmo se apoya en dos estructuras de datos que son fundamentales en su desempeño (Chávez et al., 2008):

- Una lista completa de todas las subpoblaciones que se generaron por interacciones de Chi-Cuadrado, estas subpoblaciones deben contener toda la información relacionada con los nodos padres.
- Una matriz en la cual se detalla la estructura de la red formada.

El algoritmo *BayesChaid* se basa, como su nombre lo indica, en ideas de la técnica de CHAID con adaptaciones. Estas consisten esencialmente en hacer la búsqueda de las dependencias entre las variables, no mediante los árboles de decisión completos, sino que busca a lo ancho y en profundidad en el árbol de interacciones posibles (Chávez, 2008).

Para comprender la idea del algoritmo *BayesChaid* se decidió utilizar en su cuerpo una función booleana "**Terminar**" que, dadas dos variables predictivas que se pasan como parámetros, devuelve "verdadero" en caso de que se cumpla una de las condiciones de terminación. Ellas son:

- Mínima cantidad de casos que debe tener una población para que el método considere su posible subdivisión (*MinCountOfInstancesToSplit*).
- Cota sobre la máxima cantidad de arcos que pudiese tener el camino más largo dentro de la topología generada (*MaxDepth*). Se utiliza para evitar caminos largos. Esto influye en la complejidad de los algoritmos de propagación.

En este algoritmo se incluye un nuevo parámetro, *MaxNrOfParents*, que es la cantidad de padres que podrán tener los nodos de la red a generar. Esto influye de forma especial en las tablas de probabilidades condicionadas generadas para la red (Chávez, 2008).

ByNet

ByNet muestra por pasos: la configuración del árbol de decisión que crea el software; la red Bayesiana; la tabla de probabilidades para cada nodo de la red y el resultado de la inferencia Bayesiana tomando como evidencia los atributos que el usuario seleccione. El sistema permite exportar los resultados utilizando formatos de ficheros de amplia utilización, que pueden ser cargados por otras aplicaciones.

El algoritmo ByNet parte de la construcción sucesiva de árboles de decisión utilizando la técnica CHAID (Chi-square Automatic Interaction Detector). El primer árbol obtenido romperá por la variable más significativa acorde al test Chi-cuadrado. Las siguientes variables que formen parte de ese árbol estarán en niveles inferiores, lo que significa que su posición dentro de la red estará más alejada de la clase. Como que se construye un árbol para cada variable que esté significativamente asociada a la clase y que no haya estado presente en árboles anteriores, la variable por la que rompe el último de ellos, puede tener menor importancia que otras ya incorporadas a otros árboles, y sin embargo, ella tendrá una dependencia directa de la clase.

Esta forma de proceder se realiza para evitar ciclos, pero de hecho trae consigo que el algoritmo ByNet no siempre ofrezca buenos resultados cuando la correlación entre las variables predictivas es muy elevada. En ese caso, los primeros árboles de decisión recogerán la información más importante contenida en los datos, mientras que los últimos ofrecerán una información mucho menor. La red conformada con la unión de todos los árboles no hace distinción entre unos y otros.

Por otra parte, originalmente la elección de cuántos árboles de decisión crear, es o bien puramente estadística, basada en la significación del Chi-cuadrado o se puede hacer que pertenezca por entero al usuario. Pudiera pensarse en el empleo de alguna heurística que lo ayudara a tomar una decisión en este sentido, pero ello hace más complejo el proceso de obtención de la estructura de la red (Chávez, 2008). Basados en las limitaciones aquí mencionadas, surgió la idea del siguiente algoritmo, también para el aprendizaje estructural en RB.

Algoritmo k2

El algoritmo K2 es propuesto por Cooper y Herskovits (Cooper and Herskovits, 1992). Este método comienza con la red más simple posible, es decir, una red sin arcos, y supone

que los nodos están ordenados. Para cada variable X_i, el algoritmo añade a su conjunto de padres Π_i el nodo con número menor de X_i que conduce a un máximo incremento de la calidad correspondiente a la medida de calidad elegida para el proceso de búsqueda. El proceso se repite hasta que, o bien no se incrementa la calidad, o se llega a una red completa.

Este algoritmo supone que un pedido ha sido establecido para las variables de manera que se reduce el espacio de búsqueda. El hecho de que X_1, X_2, …, X_n estén ordenadas implica que sólo los predecesores de X_k en la lista pueden ser sus nodos principales de la red. El algoritmo también asume que todas las redes son igualmente probables, porque es un algoritmo codicioso que no puede asegurar que la red resultante del proceso de aprendizaje sea la más probable dado los datos (Lazkano and Sierra, 2003).

El algoritmo K2 está basado en la optimización de una medida. Esa medida se usa para explorar, mediante un algoritmo de ascensión de colinas, el espacio de búsqueda formado por todas las redes que contienen las variables de la base de datos. Se parte de una red inicial y ésta se va modificando (añadiendo arcos, borrándolos o cambiándolos de dirección) obteniendo una nueva red con mejor medida. En concreto, la medida K2 (Cooper and Herskovits, 1992) para una red G y una base de datos D es la siguiente:

$$f(G:D) = \log P(G) + \sum_{i=1}^{n}\left[\sum_{k=1}^{s_i}\left[\log\frac{\Gamma(\eta_{ik})}{\Gamma(N_{ik}+\eta_{ik})} + \sum_{j=1}^{r_i}\log\frac{\Gamma(N_{ijk}+\eta_{ijk})}{\Gamma(\eta_{ijk})}\right]\right] \qquad 1.4$$

Donde N_{ijk} es la frecuencia de las configuraciones encontradas en la base de datos D de las variables x_i, donde *n* es el número de variables, tomando su j-ésimo valor y sus padres en G tomando su k-ésima configuración, donde S_i es el número de configuraciones posibles del conjunto de padres y r_i es el número de valores que puede tomar la variable x_i. Además, la función de Gamma es:

$$N_{ik} = \sum_{j=1}^{r_i} N_{ijk} \qquad 1.5$$

Es cierto que el algoritmo K2 es uno delos más rápidos para aprendizaje en redes bayesianas y puede utilizarse para problemas supervisados y no supervisados, pero depende del orden que se establece entre las variables.

Este algoritmo no garantiza obtener la red con mayor valor de probabilidad (Chávez,

2008).

Además, permite realizar una búsqueda heurística, para encontrar la red bayesiana con la mayor probabilidad y así construirla.

Algoritmo NaiveBayes

El clasificador *Naive Bayes* (NB) es un caso especial de una red bayesiana, en el que se asume que los atributos son condicionalmente independientes dado un atributo clase (Singh and Provan, 1995).

Este clasificador estadístico se considera, en general, como uno de los más básicos (John and Langley, 1995), (Màrquez, 2002), pero los autores coinciden en que, aunque sencillo, este clasificador sigue mostrando un buen desempeño en una gran variedad de problemas y por ello se considera vigente. Fue presentado por Duda & Hart (Duda and Hart, 1973) y utiliza como base la regla de Bayes (Bayes and Price, 1763) que se aplica cuando se desea calcular la probabilidad condicional de la ocurrencia de un evento que sucedió primero, dadas las condiciones históricas de los eventos que ocurrieron después.

Naive Bayes asume independencia entre sus características. Se puede notar observando la ecuación 1.6 que la probabilidad a posteriori $\mathrm{P}(\omega_i/x)$ depende tanto de las probabilidad a priori $\mathrm{P}(\omega_i)$ como de la verosimilitud $\mathrm{P}(x/\omega_i)$ y es por eso que este criterio tiene en cuenta ambas probabilidades a la hora de reducir el error (Urcelay and Bentos, 2014).

$$\mathrm{P}(\omega_i/x) = \frac{\mathrm{P}(x/\omega_i)P(\omega_i)}{P(x)} \qquad 1.6$$

NB es un clasificador que usa un estimador de clases, el estimador numérico y la precisión de valores son escogidos basados en un análisis de los datos de entrenamiento.

A este algoritmo, ampliamente usado en procesos de clasificación, se le considera como una forma especial, o como el modelo más simple de clasificación basado en una Red Bayesiana (Orallo et al., 2004), (Lowd and Domingos, 2005) y dentro del campo de aprendizaje automatizado y minería de datos, es conocido como uno de los algoritmos más eficientes y efectivos del aprendizaje inductivo (Zhang, 2004).

El presente algoritmo centra su fundamento en la hipótesis de que todos los atributos son independientes entre sí, conocido el valor de la variable clase. El algoritmo representa

una distribución de una mezcla de componentes, donde cada componente dentro de todas las variables se asume independiente. Esta hipótesis de independencia da lugar a un modelo de un único nodo raíz, correspondiente a la clase, y en el que todos los atributos son nodos hoja que tienen como único origen a la variable clase (Orallo et al., 2004), (Lowd and Domingos, 2005).

En varias situaciones se ha demostrado que el algoritmo en cuestión, trabaja mejor en dos casos:

- Cuando los atributos son completamente independientes, como es lógico esperar dada su premisa.
- Cuando los atributos son funcionalmente dependientes, lo que ya es menos evidente; y llegando a presentar sus peores resultados en situaciones intermedias entre estos dos extremos (Rish, 2001).

Naive Bayes se ha utilizado como un clasificador eficaz por muchos años. Tiene dos ventajas sobre otros muchos clasificadores. En primer lugar, es fácil de construir, ya que la estructura se da a priori. En segundo lugar, el proceso de clasificación es muy eficiente. Ambas ventajas son debido a su suposición de que todas las características son independientes. Aunque este supuesto de independencia es obviamente problemático. Tiene mejores resultados, sorprendentemente, que muchos clasificadores sofisticados a través de un gran número de conjuntos de datos, especialmente cuando las características no se correlacionan fuertemente (Langley et al., 1992).

Este algoritmo es uno de los clasificadores más utilizados por su simplicidad y rapidez. Se trata de una técnica de clasificación y predicción supervisada que construye modelos que predicen la probabilidad de posibles resultados. Constituye una técnica supervisada porque necesita tener ejemplos clasificados para que funcione.

1.2.3 Algoritmos basados en redes neuronales artificiales

Una red neuronal es un modelo computacional que pretende simular el funcionamiento del cerebro a partir del desarrollo de una arquitectura que toma rasgos del funcionamiento de este órgano sin llegar a desarrollar una réplica del mismo (Bello et al., 2001).

Una red neuronal es un procesador masivo, paralelo, distribuido, compuesto por unidades procesadoras simples, que tienen la capacidad de guardar conocimiento experimental y de hacerla útil para su uso (Haykin, 2004).

Las Redes Neuronales Artificiales (RNA) se consideran métodos alternativos para los procesos de clasificación al superar inconvenientes de los algoritmos convencionales de clasificación. Las redes neuronales son algoritmos de procesamiento que permiten reconocer patrones en los datos, a partir de modelos que simulan los sistemas biológicos de aprendizaje. De manera general, una red neuronal está compuesta por unidades de procesamiento llamadas neuronas, distribuidas en diferentes capas, conectadas entre sí por una serie de pesos que establecen las relaciones entre ellas.

La capacidad de procesamiento de la red se almacena en las fuerzas de conexión entre las unidades, o pesos, obtenidos por un proceso de aprendizaje a partir de un conjunto de patrones de entrenamiento. El objetivo de las RNA es conseguir que la red aprenda automáticamente las propiedades deseadas a partir de un conjunto de datos de entrada (suficientemente significativo) (Hassinger Rodriguez, 2016)).

Los datos a clasificar son ingresados en la capa de entrada, y se transmiten a las siguientes capas a través de las conexiones y sus pesos, mediante una serie de reglas y funciones matemáticas. El entrenamiento consiste en encontrar la mejor configuración de pesos que permita asociar los datos de la capa de entrada con los datos de la capa de salida obteniendo el mínimo error posible de reconocimiento.

El modelo de la neurona define el comportamiento de la misma al recibir una entrada para producir una respuesta. La topología no es más que la organización o arquitectura del conjunto de neuronas que la forman; esta organización comprende la distribución espacial de las mismas y los enlaces entre ellas. Los algoritmos de entrenamiento constituyen métodos que se aplican sobre los modelos de red para ajustar sus pesos y obtener un comportamiento determinado. Con frecuencia los algoritmos de entrenamiento son caracterizados por la clase de topologías sobre las que se aplica, los tipos de parámetros libres que afecta (pesos de las conexiones entre neuronas, parámetros del algoritmo de entrenamiento, la topología misma de la red, etc.) y la regla de modificación de los mismos (Bonet, 2008).

En los últimos años se han producido una amplia variedad de arquitecturas de redes neuronales, encontrándose entre las más utilizadas, las redes multicapa de alimentación hacia adelante (*Feed-Forward Neuronal Networks*, FFN), las cuales se distinguen porque sus neuronas están conectadas a manera de grafo acíclico dirigido (todos los arcos hacia adelante). Las redes *MultiLayer Perceptron* (MLP) constituyen un ejemplo genérico de las redes FFN, y se encuentran formadas por un conjunto de capas de neuronas ordenadas secuencialmente. Primero una capa de entrada, luego un conjunto de capas intermedias denominadas capas ocultas y por último una capa de salida. Las MLP usando neuronas ocultas con funciones no lineales, son capaces de aproximar cualquier tipo de función continua y brindar excelentes resultados en las tareas de clasificación (Salazar, 2005)

Algoritmo Perceptrón Multicapa (*MultiLayer Perceptron* MLP)

Este algoritmo, también llamado perceptrón multicapa (*Multi Layer Perceptron*), consta de varias capas de unidades computacionales interconectadas entre sí; cada neurona en una capa se encuentra directamente conectada a las neuronas de la capa anterior. El modelo se encuentra basado en funciones, ya que cada unidad de las redes mencionadas aplica una función de activación. Se utiliza para resolver problemas de asociación de patrones, segmentación de imágenes, compresión de datos, etc.

Tiene como objetivo la categorización o clasificación de forma supervisada de los datos, siendo una de las redes más utilizadas para la clasificación.

El modelo MLP se compone de capas de entrada, capas ocultas y capa de salida (ver Figura 2.1) las cuales están compuestas por una serie de neuronas que se encargan de recibir, procesar y enviar datos hacia otras neuronas procesando la información mediante distintas funciones matemáticas (Quero and Moreno, 2001) (Mather and Tso, 2016).

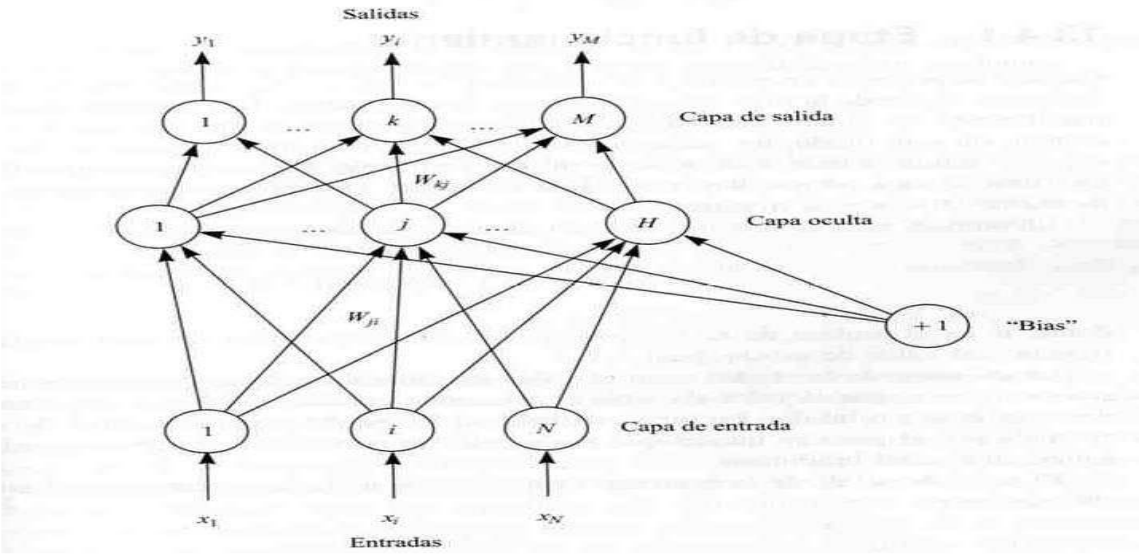


Figura 2.1 Arquitectura de una red neuronal Perceptrón Multicapa.

En (Rumelhart et al., 1985) se formaliza un método para que una red del tipo perceptrón multicapa aprendiera la asociación que existe entre un conjunto de patrones de entrada y sus salidas correspondientes: método *backpropagation error* (propagación del error hacia atrás).

Esta red tiene la capacidad de generalización: facilidad de dar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento.

1.2.4 Algoritmos perezosos (lazy)

Los algoritmos perezosos son métodos basados en instancias que utilizan enfoques conceptualmente sencillos para las aproximaciones de valores reales o discretos de las funciones de salida. Aprender en estos modelos consiste en almacenar los datos de entrenamiento presentados y, cuando una nueva instancia es encontrada, un grupo de ejemplos similares relacionados son recuperados de memoria y usados para clasificar la nueva instancia consultada. Una diferencia clave en estos enfoques, con respecto a otros métodos, es que pueden construir una aproximación diferente de la función de salida para cada ejemplo que debe ser clasificado. De hecho, muchas técnicas construyen solo una aproximación local de la función de salida que se aplica en la vecindad de una nueva instancia y nunca construyen una aproximación diseñada para tener un buen rendimiento sobre todo el espacio de instancias de entrada.

El razonamiento basado en casos se basa en el principio de usar experiencias viejas para resolver problemas nuevos. Muchos algoritmos usan este razonamiento para resolver los problemas y entre los más comunes están los de clasificación. Aunque todos los métodos de clasificación se basan en casos, existe un conjunto que se conoce como algoritmos basados en casos, o también como métodos de aprendizaje perezoso (Rodríguez, 2011).

Una nueva instancia se compara con el resto de la base de casos a través de una medida de similitud o de distancia. La clase de la nueva instancia será la misma que la del caso que más cercano esté a la nueva instancia (Díazs et al., 2015).

Algoritmo k vecinos más cercanos (IBK K-NN. K *NearestNeighbours*)

La clasificación mediante este algoritmo, publicado en 1951 por Fix y Hodges (Fix and Hodges, 1951), es una de las primeras investigaciones que proporcionan reglas basadas en métodos no-paramétricos para la manipulación de un conjunto de datos.

Este método de aprendizaje, también conocido como el método K-NN (K *Nearest Neighbours*), se basa en que los módulos de clasificación mantienen en memoria una selección de ejemplos sin crear ningún tipo de abstracción en forma de reglas o de árboles de decisión (de ahí su nombre, *lazy*, perezosos). Cada vez que una nueva instancia es encontrada, se calcula su relación con los ejemplos previamente guardados con el propósito de asignar un valor de la función objetivo para la nueva instancia.

El proceso de funcionamiento del algoritmo de k vecinos más cercanos es el siguiente: Un nuevo par (x, y) se da, en donde es observable sólo la medición X, y se desea estimar Y mediante la utilización de la información contenida en el conjunto de puntos correctamente clasificados (Lazkano and Sierra, 2003).

Las reglas de clasificación por vecindad están basadas en la búsqueda en un conjunto de prototipos de los k prototipos más cercanos al patrón a clasificar. No hay un modelo global asociado a los conceptos a aprender. Las predicciones se realizan basándose en los ejemplos más parecidos al que hay que predecir. El coste del aprendizaje es 0, todo el coste pasa al cálculo de la predicción. Se conoce como mecanismo de aprendizaje perezoso (*lazy learning*).

La idea básica sobre la que se fundamenta un clasificador KNN es que un nuevo patrón x se va a asignar a la clase más frecuente a la que pertenecen sus K_c vecinos más cercanos. Se usa K_c para referir a los K vecinos más cercanos usados para clasificación. En caso de que se produzca un empate entre dos o más clases, conviene tener una regla heurística para su ruptura como son: seleccionar la clase que contenta al vecino más próximo, seleccionar la clase con distancia media menor, etc. En la versión estándar de un clasificador KNN, los K_c vecinos más cercanos tienen implícitamente igual importancia en la decisión, sin considerarlas respectivas distancias a x. De esta forma, también es conveniente considerar una ponderación de cada uno de los K_c vecinos (Barker, 1997), de tal forma que se tenga en cuenta la distancia de cada vecino a x. En particular, se ha ponderado cada vecino de manera inversamente proporcional al cuadrado de la distancia del mismo a x, y se asigna a aquella clase cuya suma de ponderaciones sea menor (Barker, 1997).

Durante el entrenamiento, sólo guarda las instancias, no construye ningún modelo (a diferencia de, por ejemplo, los árboles de decisión), es no paramétrico (no hace suposiciones sobre la distribución que siguen los datos, a diferencia de, por ejemplo, un modelo lineal). Este algoritmo no necesita adaptación para más de dos clases (a diferencia de clasificadores lineales).

Nótese que el cómputo de la clasificación se lleva a cabo enteramente en la etapa de clasificación, dado que el entrenamiento solo se utiliza para la recolección de conjunto de muestras válidas.

1.2.5 Máquinas de Soporte Vectorial (Support Vector Machine SVM)

Las máquinas de soporte vectorial, también conocidas como máquinas de vectores de soporte (*Support Vector Machine*, SVM), son una técnica de aprendizaje supervisado muy interesante que se basa en la teoría del aprendizaje estadístico y fue presentado por Vladimir N. Vapnik en 1995 (Vapnik 1995), (Burges, 1998), partiendo de la teoría de aprendizaje estadístico y basada en el principio de minimización de riesgo estructural. Se usa mucho tanto para resolver problemas de clasificación, como para regresión.

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". Puede dividirse en SVM lineal y no lineal, este último 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 (2.4) y (2.5), respectivamente (Alfonso, 2013).

Polinomial: $k(x, x') = \langle x * x' \rangle^d$ 1.7

Gaussiana de base radial: $k(x, x') = exp\left(\frac{\|x-x\prime\|}{2\sigma^2}\right)$ 1.8

Los SVM son ampliamente utilizados en problemas de clasificación. Estos modelos crean un modelo probabilístico lineal no-binario construyendo un hiperplano, un hiperplano es una generalización de la noción de línea en geometría plana Euclidiana, o conjunto de hiperplanos óptimos en un espacio de alta dimensión que linealmente separa los patrones de clasificación. Generalmente, el hiperplano aprendido por una máquina de soporte vectorial provee una frontera de decisión para la clasificación. Los SVM maximizan el margen alrededor del hiperplano de separación. La función de decisión se específica mediante un subconjunto de muestras de entrenamiento (llamados vectores de soporte).

En muchas aplicaciones la utilización del algoritmo SVM ha mostrado tener un gran rendimiento, en parte por permitir fronteras de decisión flexibles y también por su buena capacidad de generalización.

SVM usualmente no requieren la generación de la interacción entre las variables, como hacen los métodos de regresión. Este hecho debe ahorrar algunos pasos de pre-procesamiento de datos (García et al., 2015).

1.2.6 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).

A continuación, se muestran las características de Random Forest, como uno de los multiclasificadores que mejores resultados ofrece en muchas aplicaciones de la actualidad (Breiman, 2001).

Random Forest

Es un multiclasificador que consiste en una combinación o mezcla de árboles de decisión no-podados con una selección aleatoria de las variables en cada división de cada árbol. El resultado de la clasificación es la clase mayoritaria de los resultados de los árboles que pertenecen al multiclasificador. Este algoritmo mejora la exactitud de la clasificación por la construcción estocástica de cada árbol de clasificación individual.

Orozco y otros, definen a Random Forest (RF) como un algoritmo compuesto por numerosos árboles de clasificación, en donde se definen una cantidad de árboles a desarrollar y una cantidad de atributos m, tal que sea menor a la cantidad total de atributos. Entre los árboles se reparten k patrones con reemplazo y se desarrollan los árboles, el resto de los patrones son usados para la prueba. Al desarrollar cada nodo se eligen m atributos y se determina el mejor atributo para desarrollar el nodo. Para el entrenamiento los patrones son repartidos aleatoriamente con repetición entre cada árbol.

El método RF está siendo utilizado de una manera extensiva en múltiples campos de investigación, tanto para seleccionar de un conjunto, aquellas variables con mayor poder clasificador, como para clasificar conjuntos de datos. En RF cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos.

En RF cada árbol individual se explora de una manera particular (Guillén et al., 2010):

1. Dado un conjunto de datos de entrenamiento N, se toman N muestras aleatorias con repetición como conjunto de entrenamiento.
2. Para cada nodo del árbol, se determinan M variables de entrada, y se determina "m" << M, para cada nodo, seleccionando m variables aleatorias. La variable más relevante elegida al azar se usa en el nodo. El valor de m se mantiene constante durante la expansión del bosque.
3. Cada árbol es desarrollado hasta su expansión máxima, nunca se poda.

Este algoritmo es una variante propuesta por (Breiman, 2001) y constituye una combinación de predictores de árboles de tal forma que cada árbol depende de los valores de un vector aleatorio muestreado independientemente y con la misma distribución para todos los árboles del bosque. La generalización de los errores converge hasta un límite en correspondencia con el crecimiento del número de árboles.

La generalización de un error en RF depende de la fuerza de cada árbol perteneciente al bosque y de la correlación entre ellos. Utiliza la selección aleatoria de rasgos para separar el rendimiento de cada nodo. Los estimados internos de errores de monitoreo, fuerza, y correlación, son usados para mostrar la respuesta al incrementar el número de rasgos usados en la separación. Los estimados internos también son usados para medir la importancia de la variable. Estas ideas son también aplicables a la regresión (Pérez, 2013).

1.3 Medidas para evaluar la efectividad de la clasificación

Las medidas de evaluación juegan un papel crucial tanto en la evaluación del rendimiento de clasificación y guiar el modelado clasificador.

Las medidas más conocidas para evaluar la clasificación están basadas en la matriz de confusión que se obtiene cuando se prueba el clasificador en un conjunto de datos que no intervienen en el entrenamiento, y las más usadas para base desbalanceadas según (Sun et al., 2007),(Chen et al., 2008), (Chawla et al., 2008), (Alfonso, 2013), (Pérez, 2013), (Galpert, 2016), (Arco, 2009), (García et al., 2012) , (Chávez, 2008), (Chávez et al.), (Batista et al., 2004a), (Chawla, 2009) son las que se muestran a continuación.

Las medidas más conocidas para evaluar la clasificación están basadas en la matriz de confusión que se obtiene cuando se prueba el clasificador en un conjunto de datos que no intervienen en el entrenamiento.

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 $TVP = \frac{\text{VP}}{(VP+FN)}$: es el porcentaje de casos positivos correctamente clasificados como pertenecientes a la clase positiva.
- Tasa de verdaderos negativos $TVN = \frac{\text{VN}}{(FP+VP)}$: es el porcentaje de casos negativos correctamente clasificados como pertenecientes a la clase negativa.
- Tasa de falsos positivos $TFP = \frac{\text{FP}}{(FP+VN)}$: es el porcentaje de casos negativos mal clasificados como pertenecientes a la clase positiva.
- Tasa de falsos negativos $TFN = \frac{\text{FN}}{(VP+FN)}$: 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 (*ACC*), según ecuación 1.9 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 (Padmaja et al., 2008), (García et al., 2012). 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 and Bernadó, 2009), podría no ser válida si no se clasifica correctamente cualquier instancia de la clase minoritaria.

$$ACC = \frac{VP+VN}{(VP+FN+FP+VN)} \quad 1.9$$

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 (acrónimo de *Receiver Operating Characteristic*) (Hanley and McNeil, 1983), (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. Además, proporciona herramientas para seleccionar los modelos posiblemente óptimos y descartar modelos subóptimos independientemente de (y antes de especificar) el coste de la distribución de las dos clases sobre las que se decide. La curva ROC es también independiente de la distribución de las clases en la población. Para calcular AUC sólo hay que obtener el área bajo la curva como se muestra en la ecuación 1.10.

$$AUC = \frac{1+TVP-TFP}{2} \quad 1.10$$

Por otra parte, se plantea que la curva ROC puede brindar una visión sobre-optimista del comportamiento de los algoritmos ante el desbalance. No puede capturar el efecto de aumento en el número de falsos positivos (FP), ya que este cambio no cambia significativamente el *False Positive Rate* ($\mathrm{FPR} = \mathrm{FP}/(\mathrm{FP} + \mathrm{TN})$) si el número de ejemplos negativos es muy grande. Dadas tales situaciones, la curva *Precision_Recall* (PRC) puede brindar una representación más informativa del comportamiento pues se define ploteando la razón de *precisión* sobre la razón de *recall* y la medida *precisión* considera el radio de TP con relación a TP+FP. Por estas razones, en la experimentación de este trabajo se incluyó el uso de la curva PRC para medir el comportamiento de algoritmos que manejan el desbalance (Goya et al., 2015), (Davis and Goadrich, 2006).

La precisión (*precisión*) es la medida más utilizada para la evaluación de los modelos de predicción y por tanto también la más criticada. Esta se define como el porcentaje de aciertos en la predicción sobre el total de predicciones realizadas. Una de las ventajas que tiene es su simplicidad, aunque también resulta un inconveniente por no tener en cuenta otros factores además

del número de aciertos. Esta medida no tiene en cuenta hechos tales como una distribución no balanceada de las clases, es decir, el entrenamiento del modelo con más muestras de una clase que de otras.

$$\text{Precision} = \frac{\text{VP}}{(VP+FN)} \qquad 1.11$$

Tradicionalmente, la precisión es la medida más comúnmente utilizada para estos fines. Sin embargo, para la clasificación con el problema de desbalance de clases, la precisión no es una medida adecuada, ya que la clase rara tiene muy poco impacto en la precisión en comparación con la clase prevalente (Sun et al., 2007).

Entre las medidas más utilizadas para la validación externa se mencionan en la *precisión* y el *cubrimiento* (*recall*), siendo la *precisión* el porcentaje de predicciones positivas que son correctas, y el *cubrimiento*, el porcentaje de verdaderos positivos que son correctamente detectados, o razón de verdaderos positivos. El *cubrimiento* es conocido también como *exactitud de la clase positiva* o *sensibilidad* (*sensitivity*).

Si sólo la actuación de la clase positiva es considerada, dos medidas son importantes: la verdadera Proporción Positiva (TPrate) y Predictive Value positivo (PPvalue). En la recuperación de información, la verdadera Proporción Positiva se define como *cubrimiento* (*recall*) que denota el porcentaje de objetos recuperados que son pertinentes:

$$Recall = TPrate = \frac{TP}{TP+FN} \qquad 1.12$$

Los autores en (Yoon and Kwek, 2005) se refieren a que tratar de obtener valores altos para el *recall* y la *precisión*, a la misma vez, resulta frecuentemente un objetivo contradictorio por lo que es aconsejable utilizar medidas como la *medida-F* (*F-Measure*).

El F- *Measure* (García et al., 2012) se utiliza para integrar la *precisión* y *Recall* dentro de una métrica, lo que representa una media armónica ponderada entre estos dos parámetros:

$$F = \frac{(1+\beta^2)(Precision*Recall)}{(\beta^2*Recall+Precision)} \qquad 1.13$$

1.4 Consideraciones finales del capítulo

En este capítulo se presentan las técnicas de pre-procesamiento que se pueden aplicar en bases de datos desbalanceadas, ya que nuestro caso de estudio presenta estas características. Además, se identificaron los algoritmos de clasificación clásicos basados

en redes bayesianas, redes neuronales, árboles de decisión, máquinas de soporte vectorial, algoritmos perezosos y multiclasificadores.

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 considerándose principalmente el área bajo la curva ROC (AUC), el área bajo la curva *Presicion_Recall (PRC)* y el *F-Measure,* ya que son las medidas que se van a aplicar.

2. INTERACCION DE PROTEINAS EN LA ARABIDOPSIS THALIANA Y HERRAMIENTAS QUE PERMITEN APLICAR TECNICAS PARA EL DESBALANCE Y CLASIFICACION.

CAPÍTULO 2. INTERACCION DE PROTEINAS EN LA ARABIDOPSIS THALIANA Y HERRAMIENTAS QUE PERMITEN APLICAR TECNICAS PARA EL DESBALANCE Y CLASIFICACION.

En este capítulo se presentan las características generales de un problema perteneciente al campo de la Bioinformática: predicción en las interacciones de proteínas desde una base de datos de la *Arabidopsis thaliana.* Se muestran además los resultados de algunos trabajos precedentes que permiten predecir las interacciones de proteínas, así como los resultados alcanzados hasta el momento en nuestro caso de estudio.

Se describe el conjunto de datos que constituye a la base en este proceso predictivo. Finalmente, se comentan los elementos fundamentales de las herramientas que permiten aplicar las diferentes técnicas mencionadas en el capítulo anterior y que son útiles para realizar los experimentos.

Como se mencionó en la introducción, hace algunos años atrás se habían introducido dos bases de datos dedicadas a las proteínas vegetales, una de ellas el conjunto de datos de la *Arabidopsis Thailana*.

En el año 1907 el Dr. F. Laibach descubrió el número de cromosomas de la *Arabidopsis Thaliana*; sugiriendo el potencial para la experimentación genética, entre otras razones por la brevedad de su ciclo vital.

2.1 Interacciones de proteínas

Las proteínas transmiten señales reguladoras en toda la célula, catalizan un gran número de reacciones químicas, y son cruciales para la estabilidad de numerosas estructuras celulares. Las interacciones entre las proteínas son esenciales para el funcionamiento de la célula y la identificación de tales interacciones es crucial para descifrar los mecanismos moleculares fundamentales de la célula.

La mayoría de las proteínas se unen unas con otras para llevar a cabo diferentes funciones. Estas asociaciones entre proteínas están involucradas en casi todas las funciones biológicas y los principios que gobiernan éstas interacciones entre proteínas no han sido esclarecidos del todo aún.

En las plantas las proteínas se encuentran entre las moléculas más estudiadas, incluyendo determinación de estructura, propiedades y función biológica. Existen varias razones por las que el estudio de proteínas aplicado a tejidos y organismos vegetales se está convirtiendo en un área de estudio en expansión en comparación con los estudios a nivel genético.

A pesar de existir información completa y detallada del genoma de diferentes especies vegetales, el estudio de los genes ofrece poca información acerca de la función que desempeñan las proteínas que codifican o de las variaciones de la expresión y síntesis de las diferentes proteínas como respuesta a diferentes estímulos.

Un inconveniente añadido es que el estudio de los genomas y los transcriptomas tampoco aporta respuestas acerca de cuáles son las posibles modificaciones que pueden sufrir las proteínas durante el proceso de maduración hasta que son capaces de desempeñar una función determinada.

Sin embargo, mediante el estudio de las proteínas en tejidos vegetales se pueden abordar preguntas prácticas en el campo de la biología, medicina o farmacia y permite identificar proteínas involucradas en el desarrollo de diversas fisiopatías.

Como se mencionó en la introducción, hace algunos años atrás se habían introducido dos bases de datos dedicadas a las proteínas vegetales, una de ellas el conjunto de datos de la *Arabidopsis Thaliana*, de la cual se presentó por vez primera el mapa genético, sugiriendo el potencial para la experimentación genética, entre otras razones por la brevedad de su ciclo vital.

A continuación, se detalla la descripción del conjunto de datos de dicha planta.

2.2 Descripción del conjunto de datos estudiado

El conjunto de datos de la *Arabidopsis thaliana* consta de 4314 pares de proteínas, 1438 son ejemplos de verdaderas interacciones y 2876 son ejemplos negativos (o al menos dudosos).

Los resultados reportados anteriormente demuestran que identificar simultáneamente ejemplos positivos y negativos resulta difícil, pues es raro encontrar reportes de pares de

proteínas que no interactúan, especialmente a gran escala y los casos negativos para el aprendizaje no son del todo seguros.

Con los datos originales solo se logra obtener un porciento de casos correctamente clasificados igual a 82.40 %, lo cual puede no estar expresando la realidad, debido al desbalance entre la cantidad de instancia de cada clase.

Rasgos del problema

Se seleccionaron en total 11 rasgos, los cuales se muestran a continuación, más la variable especial denominada clase, la cual identifica si hay o no una interacción de proteína:

1. "*GO similarity score biological process*: *average" (GO_sim_bp_avg)*
2. "*GO similarity score biological process*: sum" (GO_sim_bp_sum)
3. "*GO similarity score biological process*: maximum" (GO_sim_bp_max)
4. "*GO similarity score cellular component*: *average*" (GO_sim_cc_avg)
5. "*GO similarity score cellular component*: sum" (GO_sim_cc_sum)
6. "*GO similarity score cellular component*: maximum" (GO_sim_cc_max)
7. "*Pearson correlation coefficient for micro-array type*1" (PCC_1_devtissues)
8. "*Pearson correlation coefficient for micro-array type*2" (PCC_2_heterog)
9. "*Domain score* 1: *number of common domains*" (domain_match)
10. "*Domain score* 2: *number of common domains/ total number of different domains for the two proteins together*" (domain_score)
11. "*Orthologyinformation*" (orthology_score)
12. clase (valor cero identifica que no interactúan las proteínas, y el valor uno que hay una interacción de proteínas)

2.3 Resultados alcanzados previamente en el conjunto de datos en estudio.

Existen varias investigaciones sobre la interacción de proteínas en plantas, pero muy pocas tratan el desbalance en el conjunto de datos de la *Arabidopsis thaliana,* a continuación se

muestran las principales referencias encontradas sobre investigaciones anteriores en este tema:

1. (Lu et al., 2005) investiga la interacción de proteínas en hongos donde no se menciona nada sobre desbalance entre clases.
2. (Cui et al., 2008) se ofrecen características importantes del conjunto de datos de la *Arabidopsis thaliana*, pero no se trata el problema del desbalance.
3. (Mahdavi and Lin, 2007) reduce los falsos positivos auxiliándose de criterios biológicos.
4. (Chávez, 2008). Modelo de redes bayesianas en el estudio de secuencias genómicas y otros problemas biomédicos. *Tesis Doctoral*.

   Se aplica una técnica de filtrado para los falsos negativos que tienen un poco "más de confianza" de que sean negativos, auxiliándose de reglas probabilísticas, obteniéndose una mejoría en la clasificación con un 92 %, solo teniendo en cuenta algoritmos de redes bayesianas.

A continuación, se describen las herramientas usadas en los experimentos.

2.4 Herramientas que permiten aplicar técnicas para el desbalance y la clasificación.

En este epígrafe describiremos los elementos fundamentales de las herramientas que permiten aplicar las técnicas para el pre-procesamiento en datos desbalanceados y además los métodos de aprendizaje automatizado para la clasificación. Dichas herramientas son las usadas en los experimentos del siguiente capítulo, donde la primera que se describe es usada principalmente para aplicar los algoritmos de clasificación, mientras que la segunda es usada en la aplicación de las técnicas de pre-procesamiento en datos desbalanceados.

2.4.1 Weka (Waikato Environment for Knowledge Analysis)

Weka (Hall et al., 2009) 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 (poner significado).

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 3.7.11) 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: Graphic 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 (*Knowledge Flow*): 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 (*Arff Viewer*): 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 instancias 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 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 leerse 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). 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.
- Attribute Selection: 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 sub-paquetes 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 pre-procesamiento.
- 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. Además, Weka permite realizar manipulaciones sobre los datos aplicando filtros. Se pueden aplicar en dos niveles: atributos e instancias.

A continuación, se muestra en la figura 3.2 el principal panel usado en esta herramienta, el cual está constituido por el uso de los algoritmos de clasificación, los cuales son el *BayesNet* usando varios algoritmos de búsqueda como el Algoritmo *Bayes Chaid*, Algoritmo *ByNet* y el Algoritmo k2, además de *NaiveBayes*, pertenecientes a las Redes bayesianas. En las redes

neuronales usamos el MLP (*Multilayer Perceptron*,). En 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, tomando este los valores de 1, 5 y 10. En los árboles de decisión utilizamos el J48 (C4.5) que frecuentemente tiene mejores resultados que sus semejantes.

Figura 2.1 Interfaz principal del WEKA.

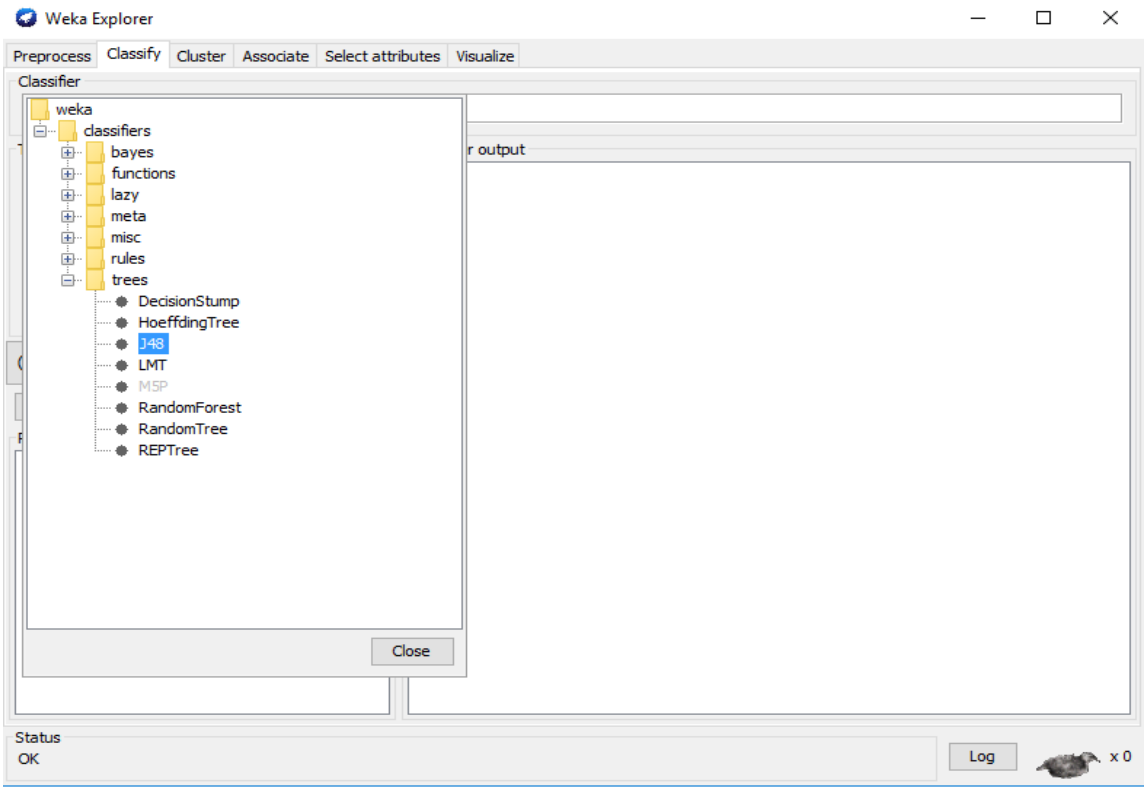


Figura 2.2 Panel principal del WEKA donde se encuentran los algoritmos de clasificación utilizados.

A continuación, se muestran las figuras con las principales configuraciones de los algoritmos, los demás tomamos los valores por defecto que trae el WEKA.

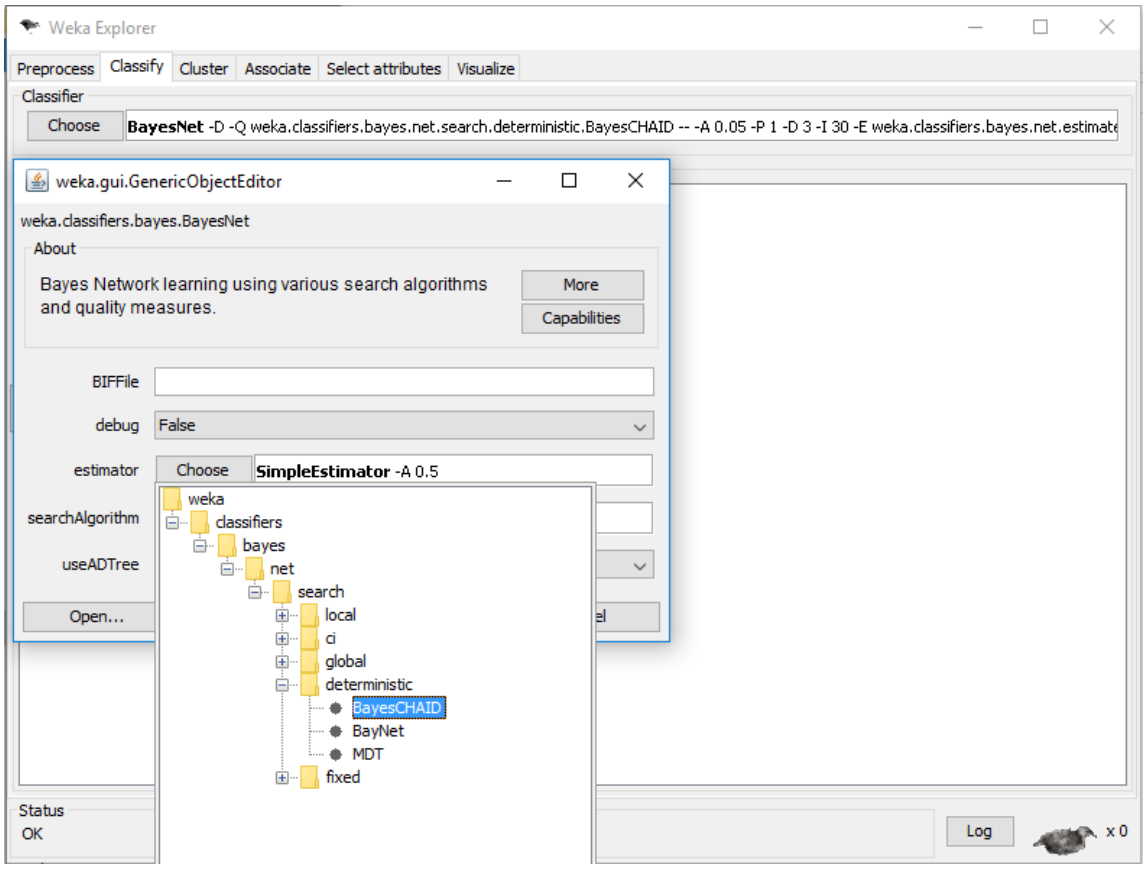


Figura 2.3 Panel del WEKA donde se selecciona el algoritmo BayesNet así como los algoritmos de búsqueda utilizados.

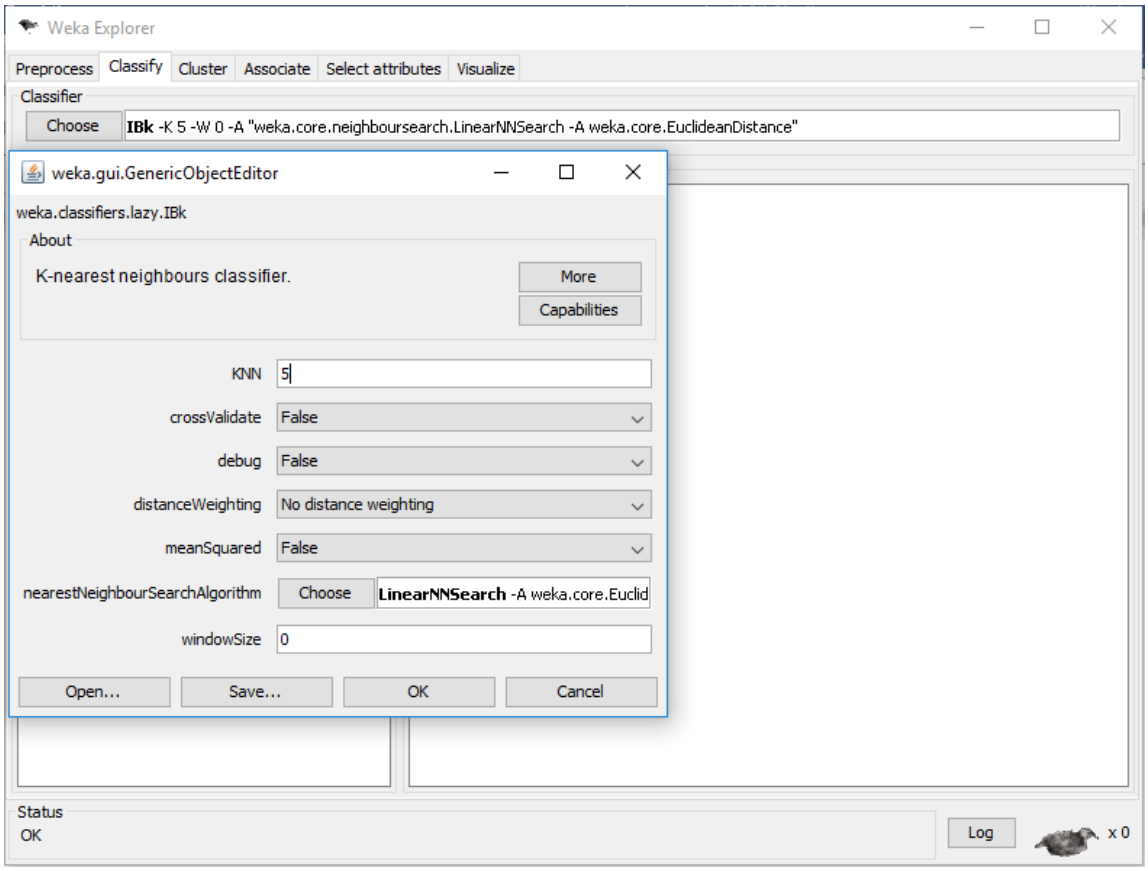


Figura 2.4 Panel del WEKA donde se selecciona el algoritmo IBK así como los distintos valores para el valor de K.

2.4.2 Keel (Knowledge Extraction based on Evolutionary Learning)

Keel (Alcalá et al., 2012) 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 pre-procesamiento (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. Esta herramienta permite el uso de diferentes formatos de entrada y salida como CSV, XML o ARFF.

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 brinda al usuario permiso de añadir sus algoritmos para la experimentación que se está diseñando. El único requisito es aceptar el formato del archivo de entrada del 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 ejecutará el algoritmo a partir del fichero global.
- Realizar un k-fold cross-validation (validación cruzada) (por defecto 10).

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 conjuntos de datos 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, como son:

- 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

Keel permite la aplicación de métodos de edición y de clasificación para conjuntos que tengan clases desbalanceadas (García et al., 2015).

A continuación, se muestra en la figura 2.3 el principal panel usado en esta herramienta, el cual está constituido por el uso de las técnicas de pre-procesamiento para el desbalance de los datos, las cuales son SMOTE, perteneciente a la categoría over-sampling. Además, de Random Under Samplig, Wilson Editing, One-Sided-Selection, pertenecientes a la categoría under-sampling, así como las técnicas híbridas las cuales son SMOTE- Tomek Hybrid, CNN + Tomek Links, Smote + ENN y el Smote RSB.

Figura 2.3 Interfaz principal del KEEL.

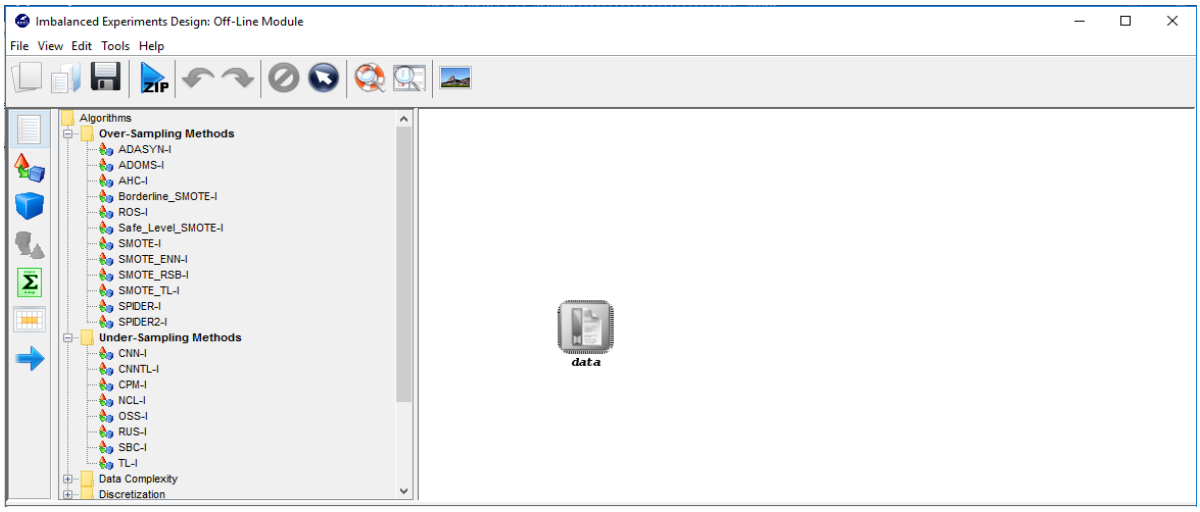


Figura 2.3 Panel principal del KEEL donde se encuentran las técnicas de pre-procesamiento utilizadas.

2.5 Consideraciones finales del capítulo

En este capítulo se describe la caracterización de la base de caso de la *Arabidopsis thaliana*, la cual constituye el objeto de estudio del presente trabajo, así como los principales resultados alcanzados en la misma en investigaciones anteriores. Además, se detallan las herramientas que incluyen los principales algoritmos que permiten realizar las técnicas de pre-procesamiento a los datos desbalanceados, así como la realización de la clasificación, dichas herramientas son usadas en la experimentación.

3 Aplicación de las técnicas para el desbalance y análisis de resultados en la clasificación.

CAPÍTULO 3. APLICACIÓN DE LAS TÉCNICAS PARA EL DESBALANCE Y ANALISIS DE RESULTADOS EN LA CLASIFICACIÓN.

En este capítulo presentamos el diseño de los experimentos realizados para aplicar coherentemente las técnicas de aprendizaje automatizado en la predicción de interacciones de proteínas en la *Arabidopsis thaliana*. Se muestran los resultados de las principales técnicas aplicadas en el pre-procesamiento, tanto a los datos originales como después de normalizar y discretizar, también se expresan los resultados de aplicar las técnicas de *over-sampling* y *under-sampling*. Además, se detallan los resultados mediante gráficos para visualizar mejor los mismos.

3.1 Diseño de los experimentos

Existe una gran cantidad de técnicas para el pre-procesamiento 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, como los que presenta nuestra base de estudio.

A continuación, describiremos cada uno de los experimentos realizados, aumentando el grado de complejidad delas técnicas de pre-procesamiento de datos, así como los métodos de clasificación a aplicar:

Los experimentos se realizaron utilizando las herramientas Weka y Keel como fue mencionado en el capítulo anterior. Para ellos se preparó el fichero con formato ". arff" con el conjunto de datos que estamos estudiando. El fichero creado se nombró "ppi_all_11features.arff". En el caso del Weka se utiliza validación cruzada con 10 subconjuntos en la estimación del error de la clasificación.

3.1.1 Experimento # 1

- Aplicar métodos clásicos de aprendizaje supervisado.
- Aplicar medidas de validación de la clasificación

Los resultados experimentales que se presentan en la Tabla 3.1 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* usando varios algoritmos de búsqueda como el Algoritmo *Bayes Chaid*, Algoritmo *ByNet* y el Algoritmo k2, además de *NaiveBayes*. En las redes neuronales usamos el MLP (*Multilayer Perceptron*,). En 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, tomando este los valores de 1, 5 y 10. En los árboles de decisión utilizamos el J48 (C4.5) que frecuentemente tiene mejores resultados que sus semejantes. Además, aplicamos las máquinas de soporte vectorial. También usamos multiclasificadores especificando el algoritmo base, el cual fue *Random Forest*.

Solo se muestran en la tabla los mejores resultados por cada clasificador, por ejemplo, en los basados en redes bayesianas solo se muestra el resultado de ROC 0.83 y de con F-measure de 0.87 con RNA.

Tabla 3.1. Resultados de las medidas de evaluación con los datos originales.

Algoritmos de Clasificación	**Área PRC**	**F-Measure**	**Área ROC**	**Exactitud**
Árboles de decisión	0,718	0.686	0.809	82.40
BayesNet (BayesChaid, K2)	0,798	0.674	0.831	81.68
Redes neuronales	0,790	0.664	0.826	82.12
IBK k= 10	0,775	0.682	0.813	81.45
Máquinas de soporte vectorial	0,612	0,666	0.749	81.52
Multiclasificador	0,792	0.67	0.809	78.55

Tanto en dicha tabla como en la Figura 3.1 se observa que el mejor resultado obtenido del Área bajo la curva *Precision_Recall* (PRC) fue de 0.798, aplicando el *BayesNet* usando como algoritmo de búsqueda Bayes Chaid y el algoritmo k2.

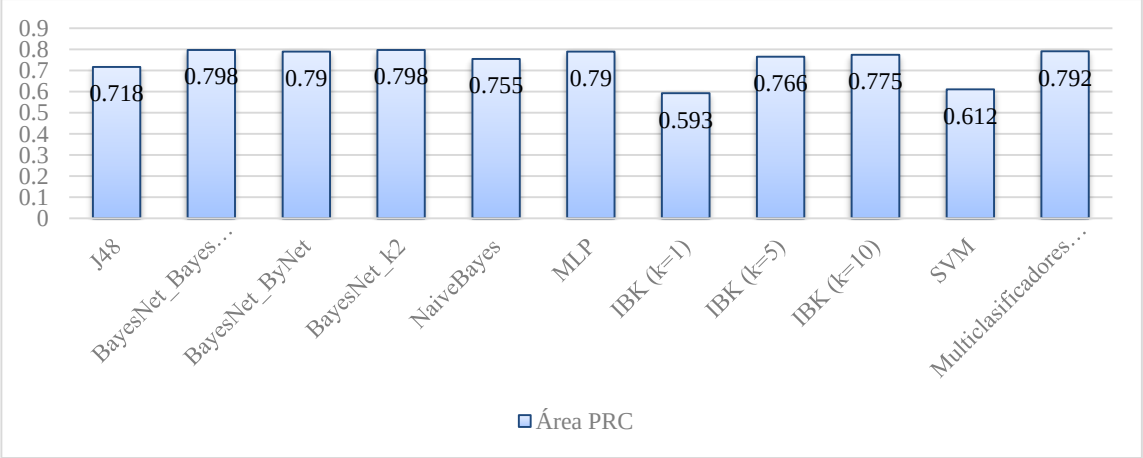


Figura 0.1 Resultados de algoritmos de clasificación aplicados al conjunto de datos sin pre-procesar.

En los anexos la tabla A.1 muestra los resultados de todos los clasificadores con todas las configuraciones incluidas para una mejor apreciación.

3.1.2 Experimento # 2

- Pre-procesar los datos (normalización de atributos, discretización de atributos)
- Guardar datos.

Ya analizamos que los resultados obtenidos en el Experimento # 1 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 se mencionó en el capítulo 1 las técnicas que aplicamos son la normalización y la discretización.

Para pre-procesar los datos, primeramente, se aplica el filtro no supervisado para normalizar los atributos y luego se dicretizan los atributos continuos, todo esto se aplica sobre los datos originales y luego este resultado se guarda en un fichero con el nombre de "ppi_all_11featuresND.arff", para proceder a la aplicación de los diferentes algoritmos para balancear los datos.

3.1.3 Experimento # 3

- Aplicar métodos de clasificación para conjuntos con clases desbalanceadas (SMOTE).

- Aplicar métodos clásicos de aprendizaje supervisado.
- Aplicar medidas de validación de la clasificación.

Para resolver el problema de desbalance se debe tratar de balancear las clases y para eso se utilizan algunos métodos de edición, en este caso usaremos el SMOTE el cual genera nuevas instancias de la clase minoritaria interpolando los valores de las instancias minoritarias más cercanas a una dada, los resultados se muestran en la siguiente tabla con solo las mejores configuraciones, ver más detalles en Tabla A.2 del Anexo 2.

Tabla 3.2. Resultados de las medidas de evaluación con los datos luego de aplicar filtro SMOTE.

Algoritmos de Clasificación	**Área PRC**	**F-Measure**	**Área ROC**	**Exactitud**
Árboles de decisión	0.813	0.721	0,815	76.25
BayesNet (BayesChaid, K2)	0,926	0.701	0.835	75.75
Redes neuronales	0,865	0.726	0.832	75.73
IBK k=1	0.872	0,762	0,851	77.43
Máquinas de soporte vectorial	0.728	0.719	0.728	76.13
Multiclasificador	0.884	0.736	0.813	73.67

En la Figura 3.2 se muestran los resultados obtenidos de los clasificadores anteriormente mencionados, aplicados tanto a los datos originales como a los datos resultantes de aplicar el filtro SMOTE perteneciente a la técnica de *over-sampling*, se tiene en cuenta la medida de clasificación del Área bajo la curva PRC. Se puede observar que los resultados mejoraron considerablemente con respecto al conjunto original, obteniendo un valor de 0.926 aplicando el *BayesNet* usando como algoritmo de búsqueda *BayesChaid* y el algoritmo k2.

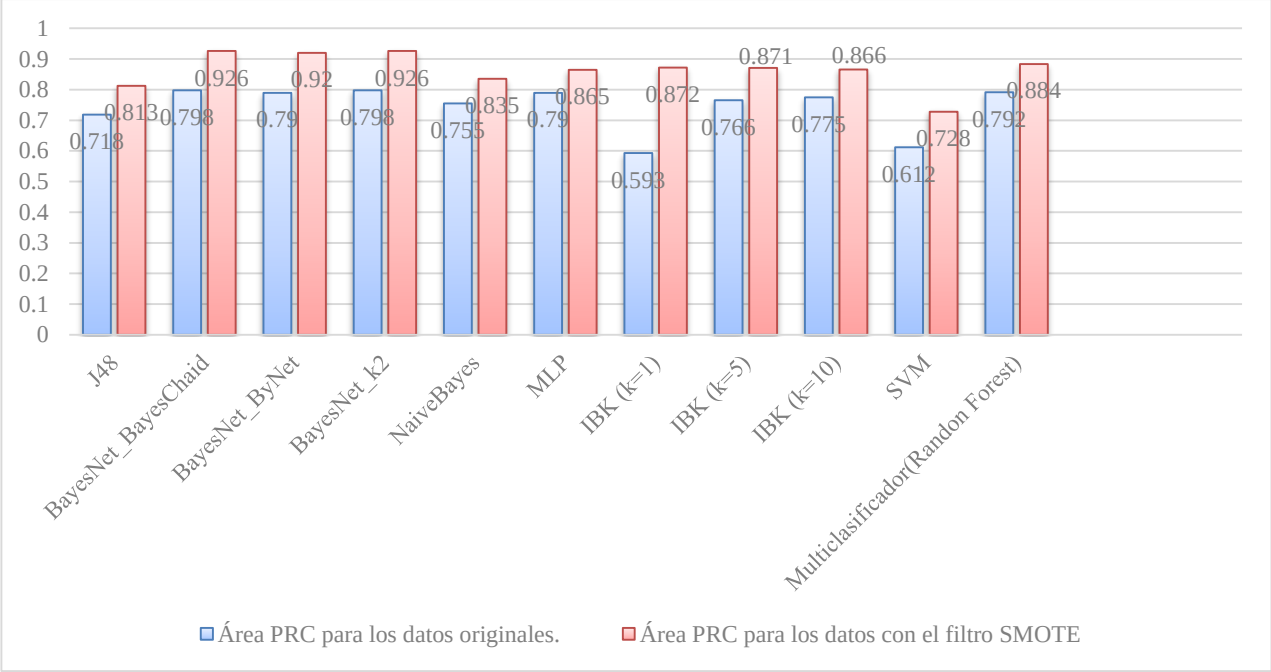


Figura 0.2 Comparación entre los resultados en datos originales y datos luego de aplicar filtro *SMOTE*.

3.1.4 Experimento # 4

- Aplicar métodos de clasificación para conjuntos con clases desbalanceadas (*Random under sampling*).
- Aplicar métodos clásicos de aprendizaje supervisado.
- Aplicar medidas de validación de la clasificación.

Otro de las técnicas que se usan para el balanceo de las clases es el *Random Under-Sampling* (RU) el cual selecciona de manera aleatoria instancias de la clase mayoritaria. En la siguiente tabla se muestran los resultados de las mejores configuraciones al aplicar esta técnica, ver más detalles en Tabla A.3 del Anexo 3.

Tabla 3.3. Resultados de las medidas de evaluación con los datos luego de aplicar el filtro RU.

Algoritmos de Clasificación	**Área PRC**	**F-Measure**	**Área ROC**	**Exactitud**
Árboles de decisión	0.824	0.696	0.785	75.20
Naives Bayes	0.864	0.692	0.828	75.17
Redes neuronales	0,848	0.725	0.805	74.58
IBK k=10	0.848	0.708	0.807	74.26

Máquinas de soporte vectorial	0.730	0.661	0.741	74.09
Multiclasificador	0.856	0.734	0.819	75.24

En la siguiente Figura 3.3 se muestran los valores obtenidos del Área bajo la curva de *Presicion_Recall* (PRC) por los clasificadores anteriormente mencionados aplicados tanto en los datos originales como en los resultantes de aplicar la técnica antes mencionada. En la Figura 3.3 se observa que los resultados mejoraron considerablemente con respecto al conjunto original obteniendo en este caso un 0.864 aplicando el *BayesNet* usando como algoritmo de búsqueda *BayesChaid* y el algoritmo k2 y el *Naives Bayes.*

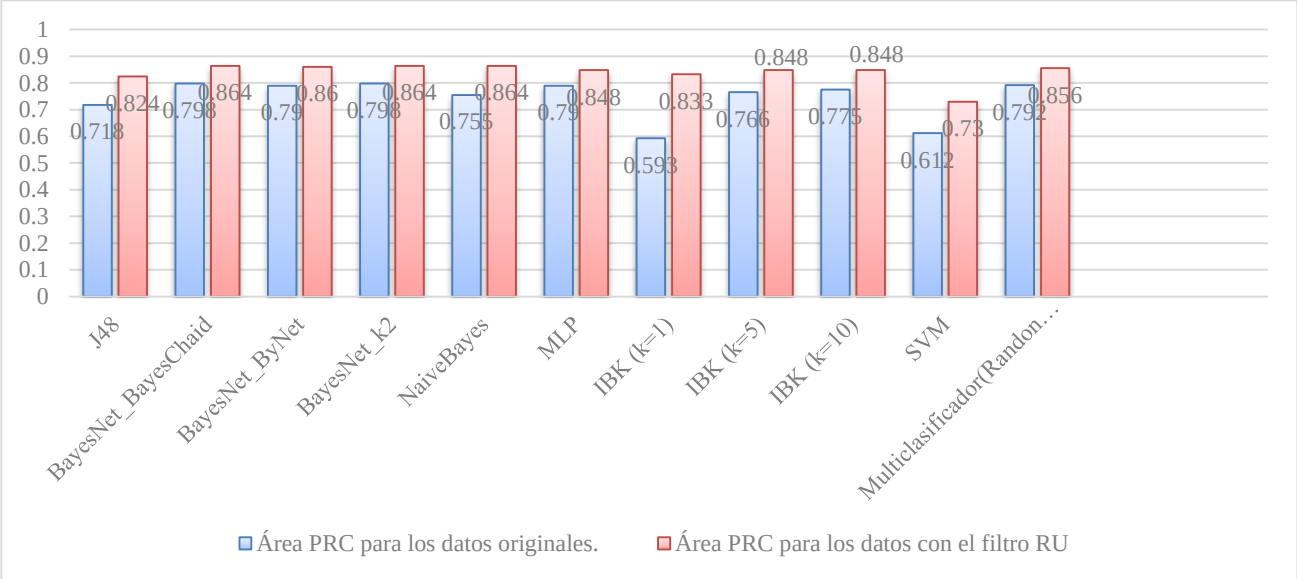


Figura 0.3. Comparación entre los resultados en datos originales y datos luego de aplicar la técnica *Random Under-Sampling*.

3.1.5 Experimento # 5

- Aplicar métodos de clasificación para conjuntos con clases desbalanceadas (*Neighborhood Cleaning Rule*).
- Aplicar métodos clásicos de aprendizaje supervisado.
- Aplicar medidas de validación de la clasificación.

El siguiente método es el *Neighborhood Cleaning Rule* (NCR) donde para cada elemento del conjunto de entrenamiento se buscan sus 3 vecinos más cercanos, perteneciente también a las técnicas de *under-sampling*. En la siguiente tabla se muestra los resultados con las mejores configuraciones, ver Tabla A.4 del Anexo 4para más detalles.

Tabla 3.4. Resultados de las medidas de evaluación con los datos luego de aplicar el filtro NCR.

Algoritmos de Clasificación	**Área PRC**	**F-Measure**	**Área ROC**	**Exactitud**
Árboles de decisión	0.895	0.828	0.873	82.46
BayesNet (k2)	0.904	0.745	0.872	77.07
Redes neuronales	0.909	0.788	0.880	78.71
IBK k=5	0.924	0.851	0.920	84.78
Máquinas de soporte vectorial	0.751	0.752	0.776	77.03
Multiclasificador	86.71	0.947	0.870	0.933

En la siguiente Figura 3.4 se muestran los los valores obtenidos del Área bajo la curva de *Presicion_Recall* (PRC) con los mismos clasificadores, aplicados tanto en los datos originales como en los resultantes luego de aplicar la técnica.

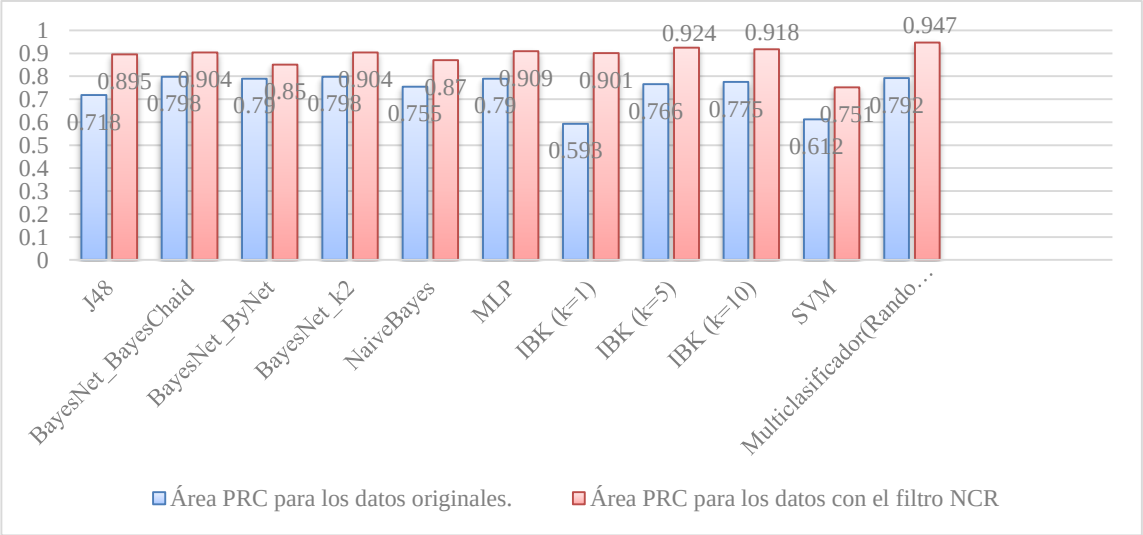


Figura 0.4. Comparación entre los resultados en datos originales y datos luego de aplicar la técnica *Neighborhood Cleaning Rule*.

En la figura 3.4 se observa que los resultados mejoraron considerablemente con respecto al conjunto original obteniendo en este caso un 0.947 aplicando el Multiclasificador *Random Forest.*

3.1.6 Experimento # 6

- Aplicar métodos de clasificación para conjuntos con clases desbalanceadas (*Wilson Editing*).
- Aplicar métodos clásicos de aprendizaje supervisado.

- Aplicar medidas de validación de la clasificación.

El siguiente método es el (*Wilson Editing ENN*), donde se elimina aquellas instancias donde la mayoría de sus vecinos pertenecen a otra clase, perteneciente a las técnicas de *under-sampling*. En la siguiente tabla se muestran los resultados de las mejores configuraciones, ver más detalles en la Tabla A.5 del Anexo 5.

Tabla 3.5. Resultados de las medidas de evaluación con los datos luego de aplicar el filtro ENN.

Algoritmos de Clasificación	Área PRC	F-Measure	Área ROC	Exactitud
Árboles de decisión	0.932	0.923	0.952	96.30
BayesNet (k2)	0.952	0.896	0.97	95.02
Redes neuronales	0.956	0.967	0.967	96.01
IBK k=1 y k=10	0.974	0.938	0.985	96.65
Máquinas de soporte vectorial	0.805	0.878	0.914	94.06
Multiclasificador	0.980	0.940	0.980	97.11

En la siguiente Figura 3.5 se muestran los porcientos de clasificación correcta para los mismos clasificadores, aplicados tanto a los datos originales como a los obtenidos luego de aplicar dicha técnica.

Como se puede apreciar nuevamente fueron mejorados los resultados.

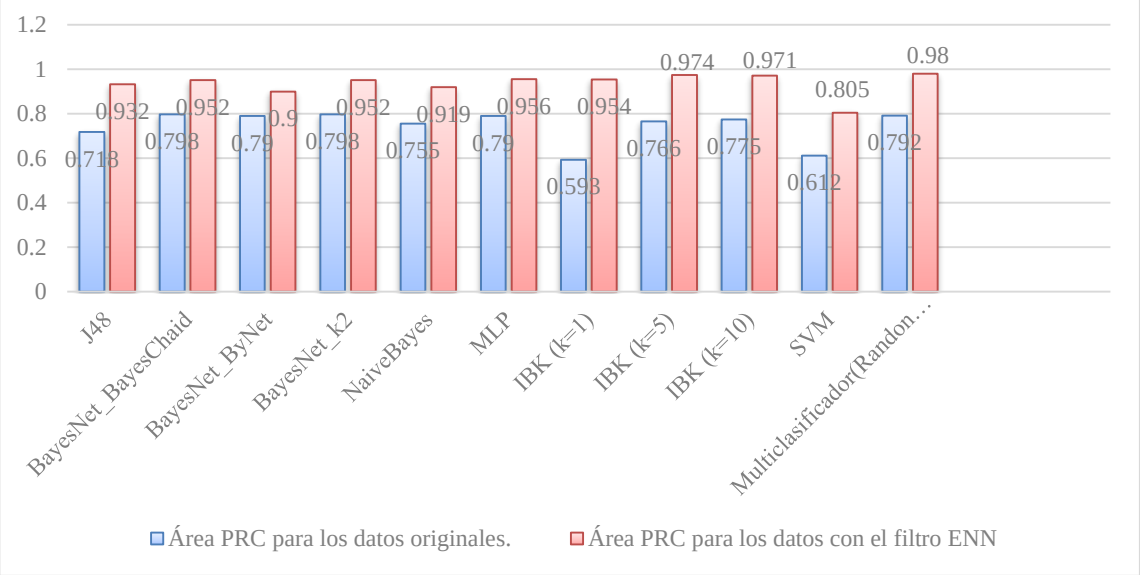


Figura 0.5. Comparación entre los resultados en datos originales y datos luego de aplicar la técnica *Wilson Editing ENN*.

En la figura 3.5 se observa que los resultados mejoraron considerablemente con respecto al conjunto original obteniendo en este caso un 0.98 aplicando el Multiclasificador *Random Forest.*

3.1.7 Experimento # 7

- Aplicar métodos de clasificación para conjuntos con clases desbalanceadas (*One-Sided-Selection*).
- Aplicar métodos clásicos de aprendizaje supervisado.
- Aplicar medidas de validación de la clasificación.

El siguiente método es el One-Sided Selection (OSS), resultante de la aplicación de Tomek links seguido de la aplicación de CNN, perteneciente a las técnicas de *under-sampling*. En la siguiente tabla se muestran los resultados con las mejores configuraciones de los clasificadores, ver más detalles en Tabla A.6 del Anexo 6.

Tabla 3.6. Resultados de las medidas de evaluación con los datos luego de aplicar el filtro OSS.

Algoritmos de Clasificación	**Área PRC**	**F-Measure**	**Área ROC**	**Exactitud**
Árboles de decisión	0.896	0.806	0.808	73.70
BayesNet (k2)	0.920	0.751	0.820	71.58
Redes neuronales	0.908	0.791	0.791	70.93
IBK k=1	0.848	0.697	0.808	74.40
Máquinas de soporte vectorial	0.730	0.661	0.741	74.09
Multiclasificadores	0.856	0.734	0.819	75.24

En la figura 3.6 se observa que los resultados mejoraron considerablemente con respecto al conjunto original obteniendo en este caso un 0.92 aplicando el *BayesNet* usando como algoritmo de búsqueda *BayesChaid* y el algoritmo k2.

Como se puede apreciar nuevamente fueron mejorados los resultados.

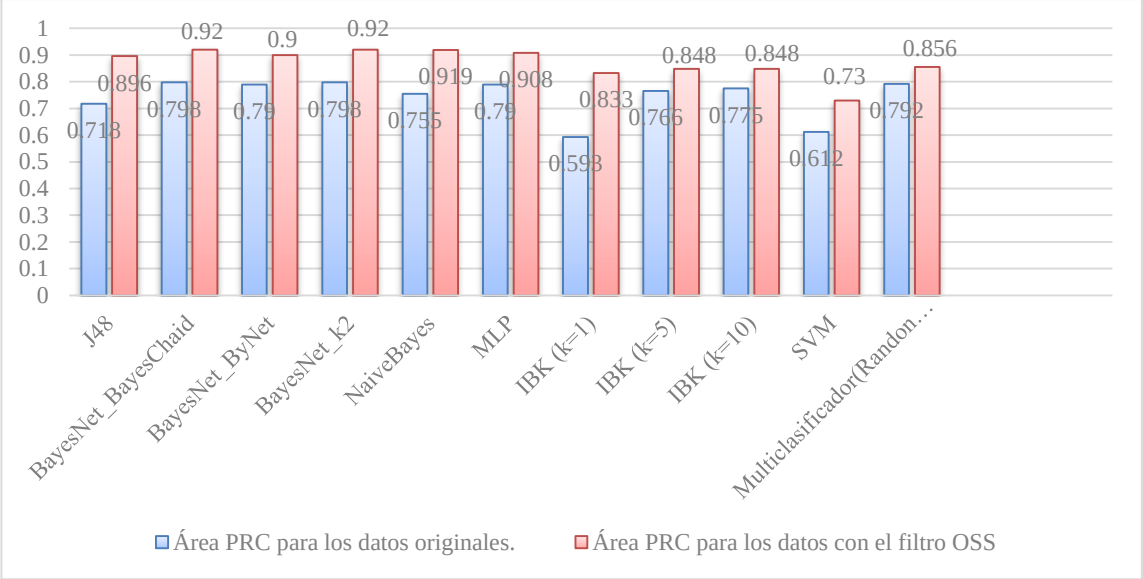


Figura 0.6. Comparación entre los resultados en datos originales y datos luego de aplicar la técnica *One-Sided Selection*.

3.1.8 Experimento # 8

- Aplicar métodos de clasificación para conjuntos con clases desbalanceadas (*SMOTE- TomekHybrid*).
- Aplicar métodos clásicos de aprendizaje supervisado.
- Aplicar medidas de validación de la clasificación.

El siguiente método es el (*SMOTE- Tomek Hybrid STL*), donde inicialmente se realiza el *over-sampling* con la clase minoritaria y luego se aplica el Tomek Link a ambas clases, constituyendo una técnica híbrida. En la siguiente tabla se muestra los resultados de los clasificadores con las mejores configuraciones, ver más detalle en Tabla A.7 del Anexo 7.

Tabla 3.7. Resultados de las medidas de evaluación con los datos luego de aplicar el filtro STL.

Algoritmos de Clasificación	**Área PRC**	**F-Measure**	**Área ROC**	**Exactitud**
Árboles de decisión	0.879	0.802	0.854	78.61
BayesNet (k2)	0.945	0.746	0.855	75.43
Redes neuronales	0.897	0.760	0.838	74.22
IBK k=5	0.917	0.848	0.905	82.91
Máquinas de soporte vectorial	0.769	0.741	0.763	74.52
Multiclasificador	0.957	0.880	0.939	86.42

En la siguiente Figura 3.7 se muestran los valores obtenidos del Área bajo la curva de *Presicion_Recall* (PRC) para los clasificadores, aplicados tanto en los datos originales como en los obtenidos luego de aplicar esta técnica.

Como se puede apreciar nuevamente fueron mejorados los resultados.

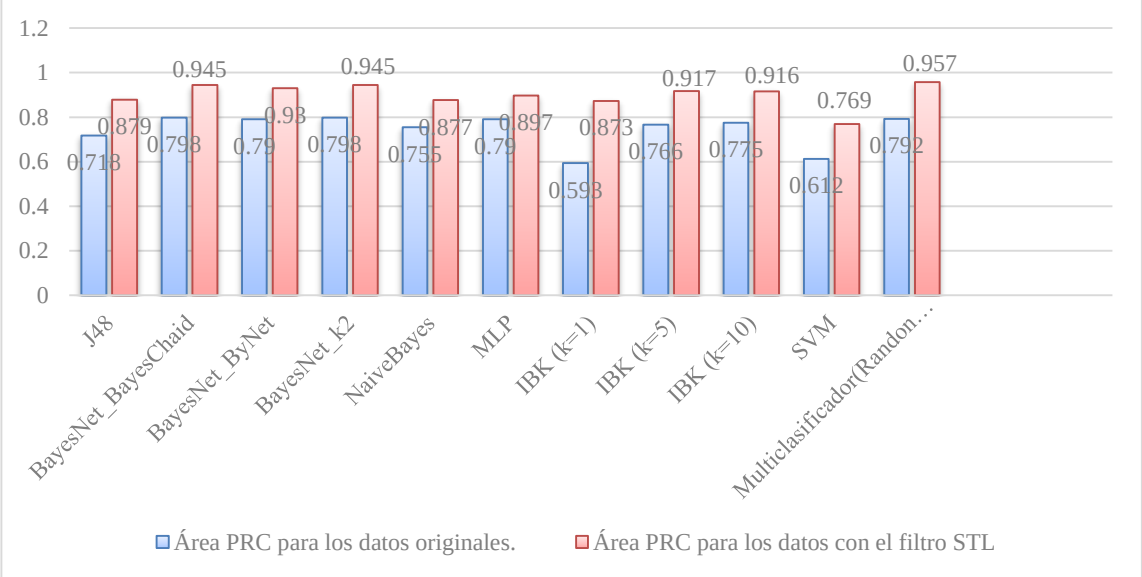


Figura 0.7 . Comparación entre los resultados en datos originales y datos luego de aplicar la técnica *SMOTE- Tomek Hybrid STL*.

En la figura 3.7 se observa que los resultados mejoraron considerablemente con respecto al conjunto original obteniendo en este caso un 0.957 aplicando el Multiclasificador *Random Forest.*

3.1.9 Experimento # 9

- Aplicar métodos de clasificación para conjuntos con clases desbalanceadas (*CNN + Tomek links*).
- Aplicar métodos clásicos de aprendizaje supervisado.
- Aplicar medidas de validación de la clasificación.

El siguiente método es el (CNN + *Tomek links* CNNTL), es similar a la selección de un solo lado (*one-sided selection*), pero el método para encontrar el subconjunto consistente se aplica antes del Tomek Links, constituyendo una técnica híbrida. En la siguiente tabla

se muestran los resultados de los clasificadores con las mejores configuraciones, ver más detalles en Tabla A.8 del Anexo 8.

Tabla 3.8. Resultados de las medidas de evaluación con los datos luego de aplicar el filtro CNNTL.

Algoritmos de Clasificación	**Área PRC**	**F-Measure**	**Área ROC**	**Exactitud**
Árboles de decisión	0.932	0.875	0.932	78.86
BayesNet (k2)	0.945	0.775	0.806	69.55
Redes neuronales	0.940	0.875	0.795	79.03
IBK k=5	0.914	0.849	0.750	75
Máquinas de soporte vectorial	0.768	0.876	0.515	78.15
Multiclasificador	0.932	0.875	0.788	78.86

En la siguiente Figura 3.8 se muestran los porcientos de clasificación correcta en los clasificadores, aplicados tanto en los datos originales como en los resultantes luego de aplicar la técnica.

Como se puede apreciar nuevamente fueron mejorados los resultados.

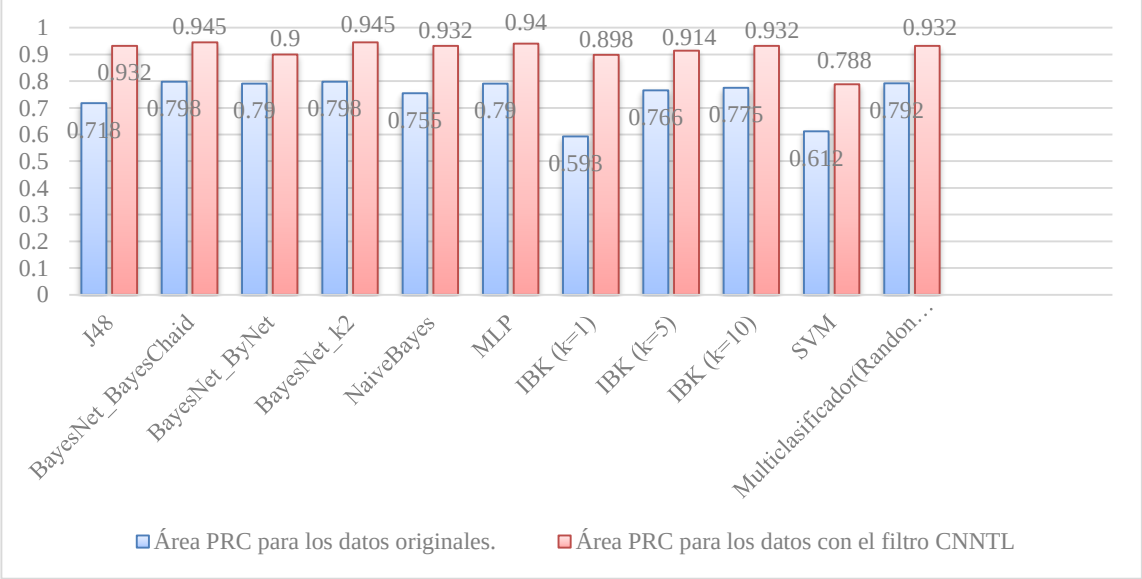


Figura 0.8. Comparación entre los resultados en datos originales y datos luego de aplicar la técnica *CNN* + Tomek links *CNNTL*.

En la figura 3.8 se observa que los resultados mejoraron considerablemente con respecto al conjunto original obteniendo en este caso un 0.945 aplicando el *BayesNet* usando como algoritmo de búsqueda *Bayes Chaid* y el algoritmo k2.

3.1.10 Experimento # 10

- Aplicar métodos de clasificación para conjuntos con clases desbalanceadas (*Smote + ENN*).
- Aplicar métodos clásicos de aprendizaje supervisado.
- Aplicar medidas de validación de la clasificación.

El siguiente método es el (*Smote + ENN SENN*), la motivación detrás de este método es similar al SMOTE + Tomek links. ENN tiende a eliminar más ejemplos que los que hace Tomek links, por lo que se espera que proporcionará una mayor depuración de los datos en profundidad, constituyendo una técnica híbrida. En la siguiente tabla se muestran los resultados de los clasificadores con sus mejores configuraciones, ver más detalles en Tabla A.9 del Anexo 9.

Tabla 3.9. Resultados de las medidas de evaluación con los datos luego de aplicar el filtro SENN.

Algoritmos de Clasificación	**Área PRC**	**F-Measure**	**Área ROC**	**Exactitud**
Árboles de decisión	0.818	0.933	0.856	87.06
BayesNet (k2)	0.910	0.792	0.888	82.86
Redes neuronales	0.908	0.806	0.885	84.09
IBK k=10	0.915	0.833	0.898	85.26
Máquinas de soporte vectorial	0.778	0.806	0.885	84.09
Multiclasificador	0.933	0.856	0.912	87.06

En la siguiente Figura 3.9 se muestran los porcientos de clasificación correcta en los clasificadores, aplicados tanto en los datos originales como en los obtenidos después de aplicar el método.

Como se puede apreciar nuevamente fueron mejorados los resultados.

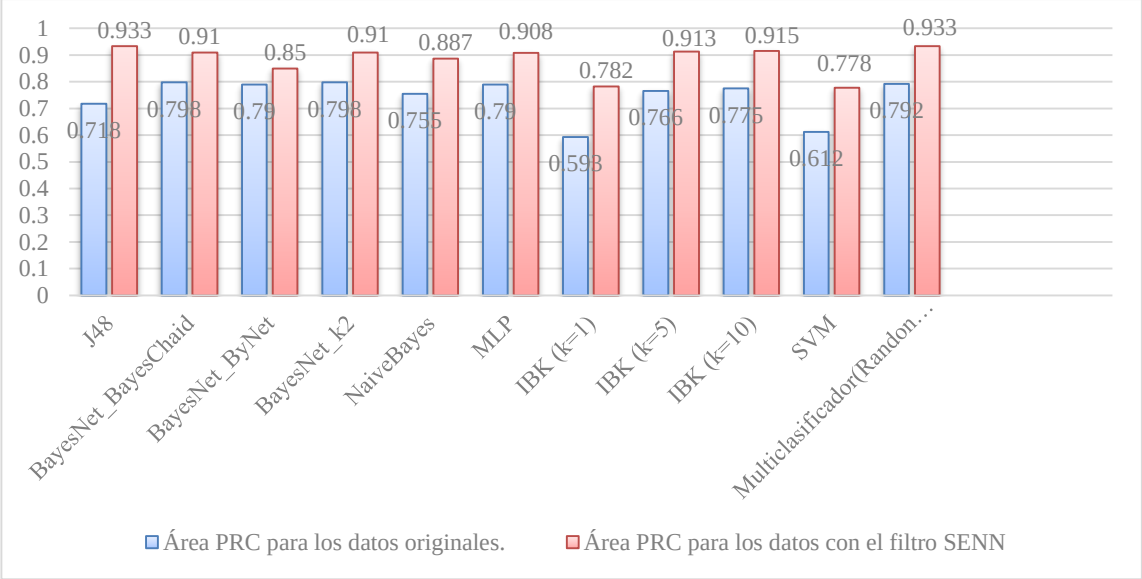


Figura 0.9. Comparación entre los resultados en datos originales y datos luego de aplicar la técnica *Smote + ENN SENN*.

En la figura 3.9 se observa que los resultados mejoraron considerablemente con respecto al conjunto original obteniendo en este caso un 0.933 aplicando el J48 y el Multiclasificador *Random Forest.*

3.1.11 Experimento # 11

- Aplicar métodos de clasificación para conjuntos con clases desbalanceadas (*SMOTE_RSB***)**
- Aplicar métodos clásicos de aprendizaje supervisado.
- Aplicar medidas de validación de la clasificación.

El siguiente método es el (Smote RSB, SRSB), es un nuevo método híbrido para procesamiento previo de conjuntos de datos desbalanceados a través de la construcción de nuevas muestras. ENN tiende a eliminar más ejemplos que los que hace *Tomek links*, por lo que se espera que proporcionará una mayor depuración de los datos en profundidad, constituyendo una técnica híbrida. En la siguiente tabla se muestran los resultados de los clasificadores con sus mejores configuraciones, ver más detalles en Tabla A.10 del Anexo 10.

Tabla 3.10. Resultados de las medidas de evaluación con los datos luego de aplicar el filtro SRSB.

Algoritmos de Clasificación	**Área PRC**	**F-Measure**	**Área ROC**	**Exactitud**
Árboles de decisión	0.842	0.735	0.826	77.22
BayesNet (k2)	0.868	0.699	0.832	75.05
Redes neuronales	0.861	0.701	0.824	74.94
IBK k=10	0.861	0.701	0.824	74.94
Máquinas de soporte vectorial	0.722	0.711	0.756	75.57
Multiclasificador	0.896	0.788	0.883	79.20

En la siguiente Figura 3.10 se muestran los porcientos de clasificación correcta en los clasificadores, aplicados tanto a los datos originales como a los resultantes después de aplicar este método.

Como se puede apreciar nuevamente fueron mejorados los resultados.

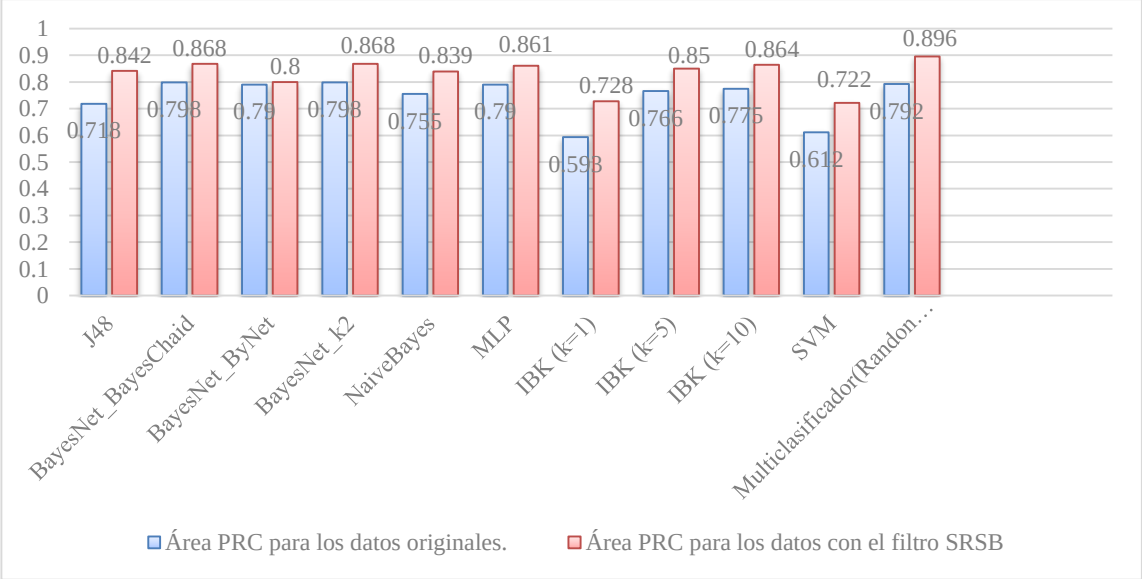


Figura 0.10 Comparación entre los resultados en datos originales y datos luego de aplicar la técnica *Smote RSB.*

En la figura 3.10 se observa que los resultados mejoraron considerablemente con respecto al conjunto original obteniendo en este caso un 0.89 aplicando el multiclasificador *Random Forest.*

3.2 Discusión de los resultados.

En la Figura 3.11 se muestran los mejores resultados por cada algoritmo de clasificación, después de aplicar las técnicas para problemas con clases desbalanceadas teniendo en

cuenta la métrica PRC, evidenciándose que el mejor resultado se obtiene al aplicar la técnica *Wilson Editing* (color rojo), usando el multiclasificador *Random Forest*, con un valor igual a 0.98.

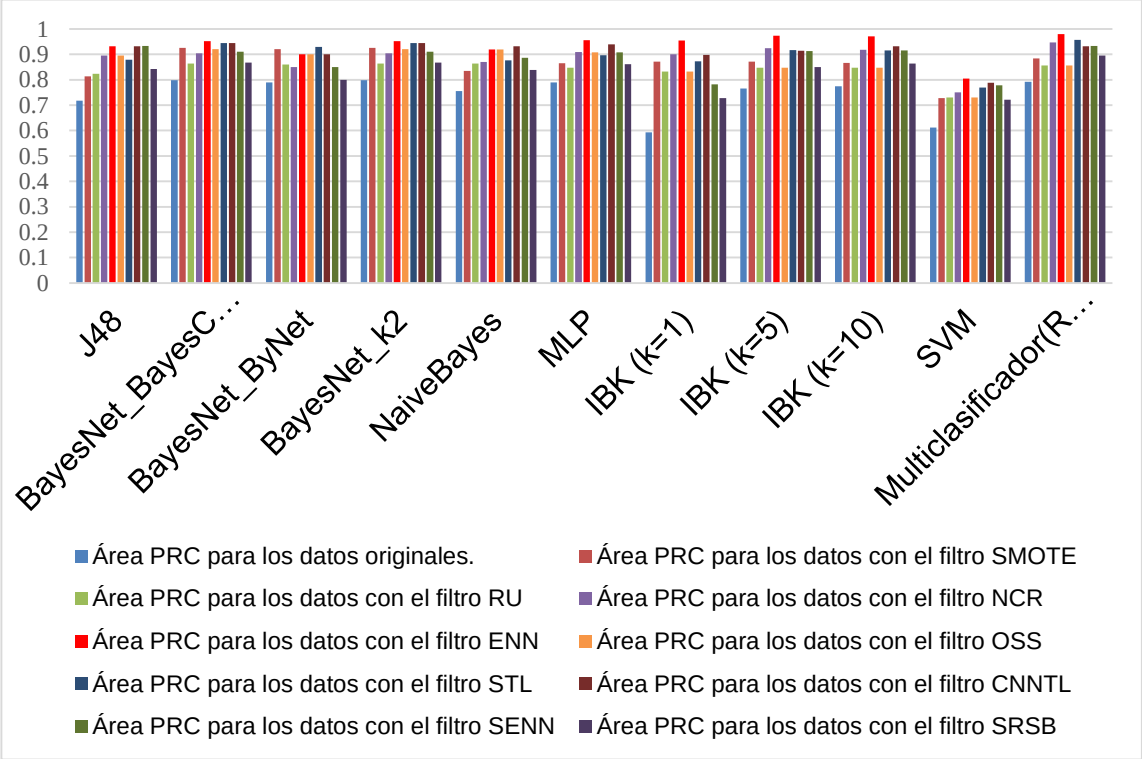


Figura 3.11. Resultados por cada clasificador con cada una de las técnicas aplicadas, usando la métrica PRC.

A continuación, se presenta un análisis de los resultados de las métricas para cada experimento.

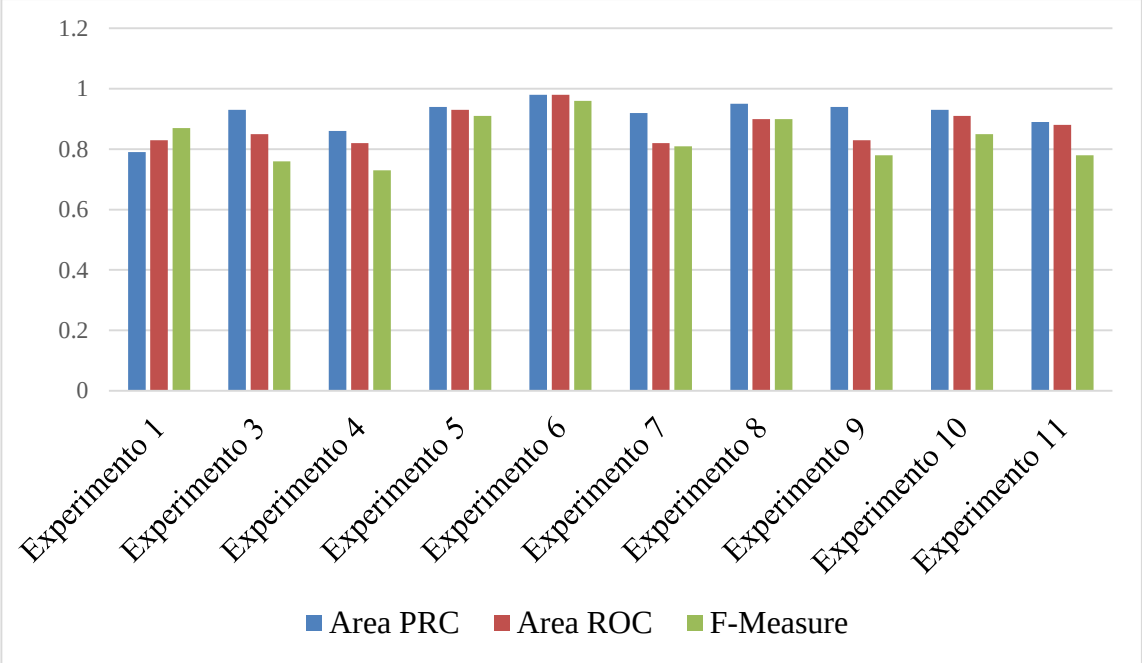


Figura 3.12. Mejores resultados de las métricas para cada experimento.

Se observa en la figura 3.12 que en la mayoría de los experimentos los resultados de las métricas mejoraron respecto a los obtenidos en los datos originales (experimento 1), las métricas PRC y ROC siempre superan sus valores, obteniéndose los mejores resultados con la técnica *Wilson Editing* (experimento 6).

Además, se puede observar que los peores resultados alcanzados para cada medida de evaluación fueron para el Área PRC con *Random Under-Sampling;* para el Área ROC los peores resultados se obtienen con *OSS* y para la medida *F-Measure* los peores resultados se obtienen con *Random Under-Sampling.*

A continuación, se observan otros resultados donde se muestra según las medidas de evaluación los clasificadores con mejores y peores resultados.

Figura 3.13. Mejores y peores resultados de la métrica PRC para cada clasificador.

Se observa en la figura 3.13 que el clasificador que mejor Área PRC obtiene es el multiclasificador *Random Forest.*

Además, se observa que el peor resultado se obtuvo con SVM.

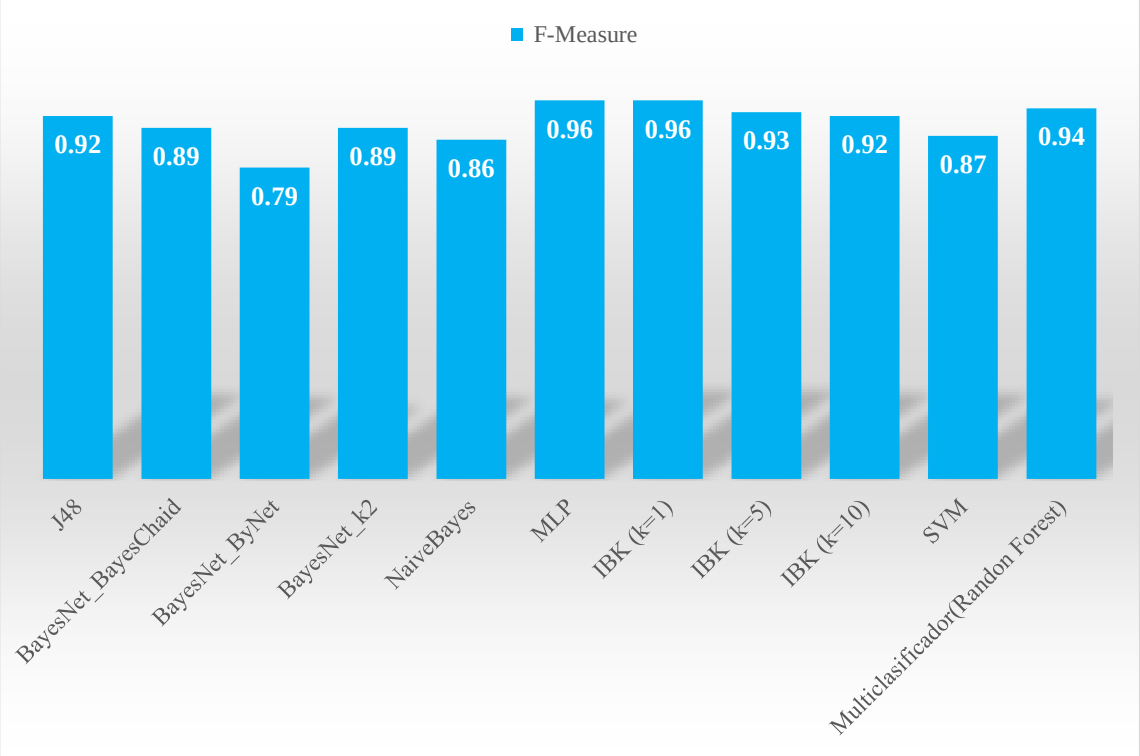


Figura 3.14. Mejores y peores resultados de la métrica *F-Measure* para cada clasificador.

Se observa en la figura 3.14 que varios clasificadores ofrecen los mejores resultados para la medida *F-Measure* como el MLP e IBK k=1.

Además, se observa que el peor resultado se obtuvo con *BayesNet* usando *Bynet*.

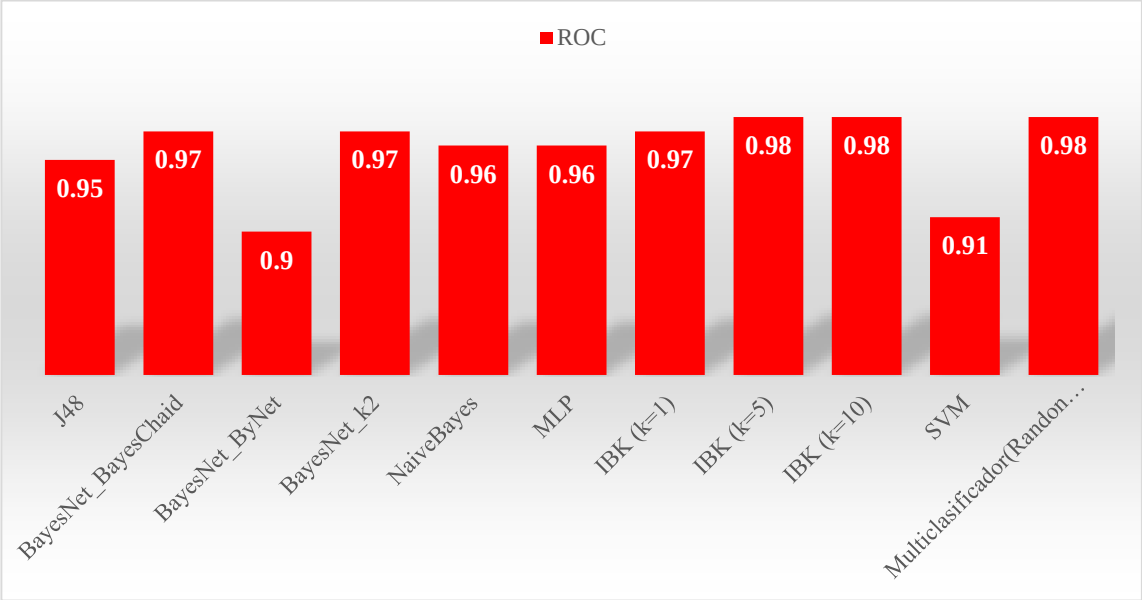


Figura 3.15. Mejores y peores resultados de la métrica ROC para cada clasificador.

Se observa en la figura 3.15 que varios clasificadores ofrecen los mejores resultados del Área ROC como el IBK para k=5, k= 10 y el multiclasificador *Random Forest.*

El peor resultado se obtuvo con *BayesNet* usando *Bynet* además del SVM.

3.3 Consideraciones finales del capítulo

Se diseñaron once experimentos para aplicar técnicas que mejoran el desbalance en el conjunto de datos de estudio. Se aplica la normalización y discretización en los datos para obtener un conjunto de mejor calidad. Luego se emplean técnicas de *over-sampling* y *under-sampling*, así como las híbridas entre ellas, usando los mismos clasificadores, donde los mejores resultados fueron obtenidos al realizar el experimento # 6 con la técnica *Wilson Editing*, lográndose una exactitud igual a 98.45 % con IBK (K=1), F-measure con valor igual a 0.96 de IBK (K=1) y ROC igual a 0. 98 (con una diferencia de 0.14 en el valor, respecto a otras investigaciones) con IBK (K=10).

También se obtienen buenos resultados con las técnicas *Neighborhood Cleaning Rule* y *SMOTE- Tomek Hybrid*.

Además, se demostró que las métricas PRC y ROC luego de aplicar cada técnica, siempre superan sus valores en los resultados.

CONCLUSIONES

Del análisis del marco teórico se concluye que resulta muy importante aplicar los métodos de pre-procesamiento de ejemplos dado el desbalance de la base de casos debiéndose evaluar su efecto. Los métodos recomendados son: over-sampling y under-sampling, donde los mejores resultados se muestran con Wilson Editing, Neighborhood Cleaning Rule y SMOTE-Tomek Hybrid.

Se identifican varios algoritmos de clasificación muy comunes, utilizando luego medidas de la matriz de confusión para la validación de los resultados, específicamente los valores del *Área Precision_Recall*, la medida *F-Measure* y las curvas ROC.

Se alcanza 98.4 % de clasificaciones correctas en la predicción de interacción de proteínas de la *Arabidopsis Thaliana*, mejorado en un 6%, respecto a los resultados anteriores. Los mejores resultados fueron obtenidos con la aplicación de la técnica Wilson Editing, usando el multiclasificador Random Forest e IBK.

Al evaluar el Área PRC y ROC comparando las diferentes técnicas de pre-procesamiento en todos los casos se aprecia resultados superiores con respecto a los datos crudos. Sin embargo, para el caso de la medida *F-Measure* se mejoran los resultados en siete de los 11 experimentos realizados.

RECOMENDACIONES

1. Utilizar algoritmos sensitivos al costo y evaluar resultados.
2. Aplicar el procedimiento empleado a otros conjuntos de datos desbalanceados.

REFERENCIAS BIBLIOGRAFICAS

ALBISUA, I., ARBELAITZ, O., GURRUTXAGA, I., LASARGUREN, A., MUGUERZA, J. & PÉREZ, J. M. 2013. The quest for the optimal class distribution: an approach for enhancing the effectiveness of learning via resampling methods for imbalanced data sets. *Progress in Artificial Intelligence,* 2**,** 45-63.

ALCALÁ, J., SÁNCHEZ, L. & GARCÍA, S. 2012. KEEL: A software tool to assess evolutionary algorithms to data mining problems *URL http://www. keel. es*.

ALFONSO, W. 2013. *Predicción de la evolución hacia la hipertensión arterial en la adultez desde la adolescencia utilizando técnicas de aprendizaje automatizado.* Universidad Central "Marta Abreu" de Las Villas.

ARCO, L. 2009. *Agrupamiento basado en la intermediación diferencial y su valoración utilizando la teoría de los conjuntos aproximados.* Tesis de doctorado en Ciencias Técnicas. Director: Rafael Esteban Bello Pérez. Tesis de Doctorado. Especialidad de Ingeniería Industrial. Facultad de Matemática, Física y Computación. Universidad Central "Marta Abreu" de las Villas.

BARANDELA, R., SÁNCHEZ, J. S., GARCıA, V. & RANGEL, E. 2003. Strategies for learning in class imbalance problems. *Pattern Recognition,* 36**,** 849-851.

BARKER, A. L. 1997. *Selection of distance metrics and feature subsets for K-nearest neighbor classifiers.* Citeseer.

BATISTA, G., PRATI, R. C. & MONARD, M. C. 2004a. A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter,* 6**,** 20-29.

BATISTA, G. E., PRATI, R. C. & MONARD, M. C. 2004b. A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter,* 6(1)**,** 20-29.

BAYES, M. & PRICE, M. 1763. An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfrs. *Philosophical Transactions (1683-1775)***,** 370-418.

BELLO, R., GARCÍA, Z., GARCÍA, M. & LOBATO, A. 2001. Aplicaciones de la IA. Santa Clara.

BONET, I. 2008. *Modelos para la clasificación de secuencias, en problemas de la Bioinformática, usando técnicas de la Inteligencia Artificial.* Doctorado, Universidad Cenral "Marta 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**,** 1145-1159.

BREIMAN, L. 2001. Random forests. *Machine learning,* 45**,** 5-32.

BURGES, C. J. 1998. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery,* 2**,** 121-167.

CABRERA, L., CASAS, G. M., BONET, I., FRANCO, P. E. & MORALES, A. 2013. *Métodos de combinación de clasificadores utilizando medidas de diversidad.* monografía Universidad Central "Marta Abreu" de Las Villas.

CHÁVEZ, M., CASAS, G., MOREIRA, J. E., BELLO, R. & GRAU, R. Perfeccionamiento de la matriz de confusión que resulta de un clasificador, en dependencia del dominio de aplicación Improvement the confusion matrix when the classifier is applied, in dependence of application domain.

CHÁVEZ, M. C. 2008. *Modelo de redes bayesianas en el estudio de secuencias genómicas y otros problemas biomédicos.* doctorado, Universidad Central "Marta Abreu" de las Villas.

CHÁVEZ, M. D. C., CASAS, G., MOREIRA, J., SILVEIRA, P., MOYA, I., BELLO, R. & GRAU, R. 2008. Predicción de mutaciones en secuencias de la proteína transcriptasa inversa del VIH usando nuevos métodos para Aprendizaje Estructural de Redes Bayesianas. . *Avances en Sistemas e Informática,,* 5(2).

CHAWLA, N. V. 2009. Data mining for imbalanced datasets: An overview. *Data mining and knowledge discovery handbook.* Springer.

CHAWLA, N. V., BOWYER, K. W., HALL, L. O. & KEGELMEYER, W. P. 2002a. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research,* 16**,** 321-357.

CHAWLA, N. V., BOWYER, K. W., HALL, L. O. & KEGELMEYER, W. P. 2002b. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research,* 16**,** 321-357.

CHAWLA, N. V., CIESLAK, D. A., HALL, L. O. & JOSHI, A. 2008. Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery,* 17**,** 225-252.

CHEN, M.-C., CHEN, L.-S., HSU, C.-C. & ZENG, W.-R. 2008. An information granulation based data mining approach for classifying imbalanced data. *Information Sciences,* 178**,** 3214-3227.

COHEN, G., HILARIO, M., SAX, H., HUGONNET, S. & GEISSBUHLER, A. 2006. Learning from imbalanced data in surveillance of nosocomial infection. *Artificial Intelligence in Medicine,,* 37(1)**,** 7-18.

COOPER, G. F. & HERSKOVITS, E. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine learning,* 9**,** 309-347.

CUI, J., LI, P., LI, G., XU, F., ZHAO, C., LI, Y., YANG, Z., WANG, G., YU, Q. & LI, Y. 2008. AtPID: Arabidopsis thaliana protein interactome database—an integrative platform for plant systems biology. *Nucleic acids research,* 36**,** D999-D1008.

DAVIS, J. & GOADRICH, M. The relationship between Precision-Recall and ROC curves. Proceedings of the 23rd international conference on Machine learning, 2006. ACM, 233-240.

DÍAZS, H., ALEMÁN, Y., CABRERA, L., MORALES, A., CHÁVEZ, M. D. C. & CASAS, G. M. 2015. Algoritmos de aprendizaje automático para clasificación de Splice Sites en secuencias genómicas. *Revista Cubana de Ciencias Informáticas,* 9**,** 155-170.

DOUGHERTY, J., KOHAVI, R. & SAHAMI, M. Supervised and unsupervised discretization of continuous features. In Machine learning: proceedings of the twelfth international conference 1995. 194-202.

DUDA, R. O. & HART, P. E. 1973. *Pattern classification and scene analysis*, Wiley New York.

FELGAER, P. 2004. Optimización de Redes Bayesianas basado en técnicas de aprendizaje por inducción. . *Reportes Técnicos en Ingeniería del Software,* 6(2)**,** 64-69.

FIX, E. & HODGES, J. L. 1951. Discriminatory analysis-nonparametric discrimination: consistency properties. DTIC Document.

FREITAS, A. 2011. Building cost-sensitive decision trees for medical applications. *AI Communications,* 24**,** 285-287.

GALPERT, D. 2016. *Contribuciones al enfoque de comparación para par en la detección de genes ortólogos.* Universidad Central "Marta Abreu" de Las Villas.

GARCÍA, S., LUENGO, J. & HERRERA, F. 2015. *Data preprocessing in data mining. New York: Springer.*, New York: Springer.

GARCIA, S., LUENGO, J., SÁEZ, J. A., LOPEZ, V. & HERRERA, F. 2013. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering,* 25(4)**,** 734-750.

GARCÍA, V., SÁNCHEZ, J. S. & MOLLINEDA, R. A. 2012. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowledge-Based Systems,* 25**,** 13-21.

GONZÁLEZ, O. L., TRINIDAD, J. F. M., BORROTO, M. G. & ERRO, L. E. 2014. Clasificadores Supervisados basados en Patrones Emergentes para Bases de Datos con Clases Desbalanceadas.

GOYA , A. A., GALPERT CAÑIZARES, D. & ENRÍQUEZ RODRÍGUEZ, F. A. 2015. Inclusión en Weka de filtros basados en conjuntos aproximados para bases desbalanceadas (Inclusion of filters in Weka based in rough sets for imbalanced bases). *GECONTEC: Revista Internacional de Gestión del Conocimiento y la Tecnología,* 3.

GUILLÉN, E. O., GARCÍA, G. I., VÁZQUEZ, S., ATENCIO, J. A. D., RAMOS, J. C. & DELGADO, F. G. 2010. Métodos de clasificación para identificar lesiones en piel a partir de espectros de reflexión difusa. *Revista Ingeniería Biomédica,* 4**,** 34-39.

HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P. & WITTEN, I. H. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter,* 11**,** 10-18.

HAN, H., WANG, W. Y. & MAO, B. H. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. . In International Conference on Intelligent Computing, 2005. Springer Berlin Heidelberg, pp. 878-887.

HAN, J., PEI, J. & KAMBER, M. 2011. *Data mining: concepts and techniques. Elsevier.*, Elsevier.

HANLEY, J. A. & MCNEIL, B. J. 1983. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology,* 148**,** 839-843.

HART, P. E. 1968. the condensed nearesty neighbor rule. *I EEE Transactions of Information theory,* IT-14**,** 515-516.

HASSINGER RODRIGUEZ, M. M. 2016. Aplicación de técnicas de minería de datos En accidentes de tráfico.

HAYKIN, S. 2004. A comprehensive foundation. *Neural Networks,* 2.

HERNÁNDEZ, C. L. & RODRÍGUEZ, J. 2013. Preprocesamiento de datos estructurados. *Vinculos,* 4 (2)**,** 27-48.

HUANG, J. & LING, C. X. 2005. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering,* 17**,** 299-310.

JACKOWSKI, K., KRAWCZYK, B. & WOŹNIAK, M. Cost-sensitive splitting and selection method for medical decision support system. International Conference on Intelligent Data Engineering and Automated Learning, 2012. Springer, 850-857.

JOHN, G. H. & LANGLEY, P. Estimating continuous distributions in Bayesian classifiers. Proceedings of the Eleventh conference on Uncertainty in artificial intelligence, 1995. Morgan Kaufmann Publishers Inc., 338-345.

KRAWCZYK, B., WOŹNIAK, M. & SCHAEFER, G. 2014. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing,* 14**,** 554-562.

KUBAT, M. & MATWIN, S. 1997. Addressing the curse of imbalanced training sets: one-sided selection. . *In ICML,* Vol. 97**,** 179-186.

LANGLEY, P., IBA, W. & THOMPSON, K. An analysis of Bayesian classifiers. Aaai, 1992. 223-228.

LAURIKKALA, J. Improving identification of difficult small classes by balancing class distribution. In Conference on Artificial Intelligence in Medicine in Europe., 2001. Springer Berlin Heidelberg., (pp. 63-66).

LAZKANO, E. & SIERRA, B. Bayes-nearest: a new hybrid classifier combining bayesian network and distance based algorithms. Portuguese Conference on Artificial Intelligence, 2003. Springer, 171-183.

LENCA, P., LALLICH, S., DO, T.-N. & PHAM, N.-K. A comparison of different off-centered entropies to deal with class imbalance for decision trees. Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2008. Springer, 634-643.

LI, D.-C., LIU, C.-W. & HU, S. C. 2010. A learning method for the class imbalance problem with medical data sets. *Computers in biology and medicine,* 40**,** 509-518.

LIN, N., WU, B., JANSEN, R., GERSTEIN, M. & ZHAO, H. 2004. Information assessment on predicting protein-protein interactions. *BMC bioinformatics,* 5(1)**,** 1.

LIU, W., CHAWLA, S., CIESLAK, D. A. & CHAWLA, N. V. A Robust Decision Tree Algorithm for Imbalanced Data Sets. SDM, 2010. SIAM, 766-777.

LOMAX, S. & VADERA, S. 2013. A survey of cost-sensitive decision tree induction algorithms. *ACM Computing Surveys (CSUR),* 45**,** 16.

LÓPEZ, M. 2007. *Mejoras en la usabilidad de la Web a través de una estructura complementaria* (Doctoral dissertation, Facultad de Informática).

LÓPEZ, V., FERNÁNDEZ, A., GARCÍA, S., PALADE, V. & HERRERA, F. 2013. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences,* 250**,** 113-141.

LÓPEZ, V., FERNÁNDEZ, A. & HERRERA, F. 2014a. On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed. *Information Sciences,* 257**,** 1-13.

LÓPEZ, V., TRIGUERO, I., CARMONA, C. J., GARCÍA, S. & HERRERA, F. 2014b. Addressing imbalanced classification with instance generation techniques: IPADE-ID. *Neurocomputing,* 126**,** 15-28.

LOWD, D. & DOMINGOS, P. Naive Bayes models for probability estimation. Proceedings of the 22nd international conference on Machine learning, 2005. ACM, 529-536.

LU, L. J., XIA, Y., PACCANARO, A., YU, H. & GERSTEIN, M. 2005. Assessing the limits of genomic data integration for predicting protein networks. *Genome research,* 15**,** 945-953.

LUENGO, J., FERNÁNDEZ, A., GARCÍA, S. & HERRERA, F. 2011. Addressing data complexity for imbalanced data sets: analysis of SMOTE-based oversampling and evolutionary undersampling. *Soft Computing,* 15**,** 1909-1936.

LUENGO, J., GARCÍA, S. & HERRERA, F. 2012. On the choice of the best imputation methods for missing values considering three groups of classification methods. , . *Knowledge and information systems,* 32(1)**,** 77-108.

MAHDAVI, M. A. & LIN, Y.-H. 2007. False positive reduction in protein-protein interaction predictions using gene ontology annotations. *BMC bioinformatics,* 8**,** 1.

MÀRQUEZ, L. 2002. Aprendizaje automático y procesamiento del lenguaje natural. *Tratamiento del lenguaje natural, Edicions de la Universitat de Barcelona*.

MATHER, P. & TSO, B. 2016. *Classification methods for remotely sensed data*, CRC press.

MENARDI, G. & TORELLI, N. 2014. Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery,* 28**,** 92-122.

MIN, F. & ZHU, W. A competition strategy to cost-sensitive decision trees. International Conference on Rough Sets and Knowledge Technology, 2012. Springer, 359-368.

MITCHELL, T. M. 1997. *Machine learning.* , New York.

MORENO, J., RODRÍGUEZ, D., SICILIA, M. A., RIQUELME, J. C. & RUIZ, R. 2009. SMOTE-I: mejora del algoritmo SMOTE para balanceo de clases minoritarias. . *Actas de los Talleres de las*

Jornadas de Ingeniería del Software y Bases de Datos, . Madrid-España: Universidda de Alcala.

NOOREN, I. M. & THORNTON, J. M. 2003. Diversity of protein–protein interactions. *The EMBO journal,* 22**,** 3486-3492.

ORALLO, H., RAMIREZ, J., QUINTANA, C. R., ORALLO, M. J. H., QUINTANA, M. J. R., RAMÍREZ, C. F., BOYD, H. W., STASCH, S. F. W., GOULD, F. & GD SCHMIDT, C. 2004. *Introducción a la Minería de Datos*, Pearson Prentice Hall.

ORRIOLS, A. & BERNADÓ, E. 2009. Evolutionary rule-based systems for imbalanced data sets. *Soft Computing,* 13**,** 213-225.

PADMAJA, T. M., KRISHNA, P. R. & BAPI, R. S. Majority filter-based minority prediction (MFMP): An approach for unbalanced datasets. TENCON 2008-2008 IEEE Region 10 Conference, 2008. IEEE, 1-6.

PEARL, J. 1998. Probabilistic reasoning in intelligent systems. *Morgan-Kaufmann(San Mateo)*.

PEARL, J. 2014. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* , Morgan Kaufmann.

PÉREZ, D. 2013. *Algoritmos supervisados para la detección de ortólogos con manejo del desbalance.* Universidad Central "Marta Abreu" de Las Villas.

PHIZICKY, E. M. & FIELDS, S. 1995. Protein-protein interactions: methods for detection and analysis. *Microbiological reviews,* 59**,** 94-123.

POLIKAR, R. 2006. Ensemble based systems in decision making. *IEEE Circuits and systems magazine,* 6**,** 21-45.

QUERO , J. M. Q. & MORENO, N. M. 2001. Las redes neuronales y la industria eléctrica: desarrollo de aplicaciones prácticas. *Automática e instrumentación***,** 84-88.

QUINLAN, J. R. 1993. *C4. 5: programs for machine learning.* , Morgan Kauffmann.

QUINLAN, J. R. 1996. Improved use of continuous attributes in C4. 5. *Journal of artificial intelligence research,* 4**,** 77-90.

RAMENTOL, E., CABALLERO, Y., BELLO, R. & HERRERA, F. 2012. . SMOTE-RSB*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory. *Knowledge and information systems,* 33(2)**,** 245-265.

RISH, I. An empirical study of the naive Bayes classifier. IJCAI 2001 workshop on empirical methods in artificial intelligence, 2001. IBM New York, 41-46.

RODDA, S. & MOGALLA, S. 2011. A normalized measure for estimating classification rules for multi-class imbalanced datasets. *International Journal of Engineering Science and Technology,* 3**,** 3216-3220.

RODRÍGUEZ, S. M. R. 2011. Modelo de un sistema de razonamiento basado en casos para el análisis en la gestión de riesgos. *Serie Científica-Universidad de las Ciencias Informáticas,* 4.

RUMELHART, D. E., HINTON, G. E. & WILLIAMS, R. J. 1985. Learning internal representations by error propagation. DTIC Document.

SÁEZ, J. A., LUENGO, J. & HERRERA, F. 2013. Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. *Pattern Recognition,* 46(1)**,** 355-364.

SALAZAR, S. 2005. *Una Herramienta Software para Redes Neuronales Recurrentes.. NEngine v1. 0.*

SINGH, M. & PROVAN, G. M. 1995. Efficient learning of selective Bayesian network classifiers.

SODA, P. 2011. A multi-objective optimisation approach for class imbalance learning. *Pattern Recognition,* 44**,** 1801-1810.

SUN, Y., KAMEL, M. S., WONG, A. K. & WANG, Y. 2007. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition,* 40**,** 3358-3378.

TOMEK, I. 1976. Two modifications of CNN. . *IEEE Systems, Man and Cybernetics,* 6**,** 769-772.

URCELAY, G. & BENTOS, M. 2014. *Reconocimiento de Patrones.* Instituto de Ingeniería Eléctrica Facultad de Ingeniería, Universidad de la República.

VALDOVINOS, R. M. 2006. *Técnicas de Submuestreo , Toma de Decisiones y Análisis de Diversidad en Aprendizaje Supervisado con Sistemas Múltiples de Clasificación.* Doctoral, Universidad de Jaume.

VAPNIK, V. 1995. *The Nature of Statistical Learning Theory,* New York, Springer-Verlag.

VAPNIK , V. 1995. The nature of statistical learning theory. Springer Heidelberg.

WEI, W., LI, J., CAO, L., OU, Y. & CHEN, J. 2013. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web,* 16**,** 449-475.

WEISS, G. M., MCCARTHY, K. & ZABAR, B. 2007. Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs? *DMIN,* 7**,** 35-41.

WITTEN, I. H., & FRANK, E. 2005. *Data Mining: Practical machine learning tools and techniques.* , Morgan Kaufmann.

WITTEN, I. H. & FRANK, E. 2005. *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann.

YOON, K. & KWEK, S. An unsupervised learning approach to resolving the data imbalanced issue in supervised learning problems in functional genomics. Fifth International Conference on Hybrid Intelligent Systems (HIS'05), 2005. IEEE, 6 pp.

ZHANG, H. 2004. The optimality of naive Bayes. *AA,* 1**,** 3.

ZHANG, S., ZHANG, Q. C. & YAN, Y. 2003. Data preparation for data mining. *Applied artificial inteligent***,** 375-381.

ANEXOS

Anexo 1: Resultados del Experimento # 1.

Tabla A.1 Resultados de algoritmos de clasificación aplicados al conjunto de datos sin pre-procesar.

Algoritmos de Clasificación		**RVP**	**RVN**	**Exactitud**	**Recall**	**Área PRC**	**F-Measure**	**Área ROC**
Árboles de decisión	J48	0.577	0.947	82.40	0.577	0,718	0.686	0.809
Redes bayesianas	BayesNet BayesChaid	0.567	0.942	81.68	0.567	0,798	0.674	0.831
	BayesNet ByNet	0.458	0.989	81.20	0.458	0.790	0.619	0.77
	BayesNet K2	0.567	0.942	81.68	0.567	0,798	0.674	0.831
	NaiveBayes	0.538	0.943	80.78	0.538	0,755	0.651	0.799
Redes neuronales	MLP	0.531	0.967	82.12	0.531	0,790	0.664	0.826
Métodos basados en instancias	IBK k=1	0.663	0.813	76.33	0.663	0,593	0.651	0.755
	IBK k=5	0.61	0.921	81.75	0.61	0,766	0.69	0.81
	IBK k=10	0.596	0.924	81.45	0.596	0,775	0.682	0.813
Máquinas de soporte vectorial	SVM	0.551	0.947	81.52	0.551	0,612	0,666	0.749
Multiclasificador	(Randon	0.652	0.852	78.55	0.652	0,792	0.67	0.809

	Forest)							

Anexo 2. Resultados del Experimento # 3.

Tabla A.2 Resultados de algoritmos de clasificación aplicados al conjunto de datos obtenido después de realizar el Experimento # 2 y aplicar el filtro SMOTE.

Algoritmos de Clasificación		**RVP**	**RVN**	**Exactitud**	**Recall**	**Área PRC**	**F-Measure**	**Área ROC**
Árboles de decisión	J48	0.613	0.912	76.25	0,613	0.813	0.721	0,815
Redes bayesianas	BayesNet Bayes Chaid	0.568	0.947	75.75	0.568	0,926	0.701	0.835
	BayesNet ByNet	0.560	0.950	70.75	0.560	0.920	0.7	0.830
	BayesNet K2	0.568	0.947	75.75	0.568	0,926	0.701	0.835
	NaiveBayes	0,568	0,948	75.78	0,568	0.835	0,701	0,835
Redes neuronales	MLP	0.645	0.87	75.73	0.645	0,865	0.726	0.832
Métodos basados en instancias	IBK k=1	0,724	0,825	77.43	0,724	0.872	0,762	0,851
	IBK k=5	0,696	0,836	76.63	0,696	0.871	0,749	0,844
	IBK k=10	0,671	0,845	75.81	0,671	0.866	0,735	0,836
Máquinas de soporte vectorial	SVM	0.611	0.912	76.13	0.611	0.728	0.719	0.728
Multiclasificador	(Randon Forest)	0.734	0.739	73.67	0.734	0.884	0.736	0.813

Anexo 3. Resultados del Experimento # 4.

Tabla A.3 Resultados de algoritmos de clasificación aplicados al conjunto de datos obtenido después de realizar el Experimento # 2 y aplicar el filtro RU.

Algoritmos de Clasificación		**RVP**	**RVN**	**Instancias correctas**	**Recall**	**Área PRC**	**F-Measure**	**Área ROC**
Árboles de decisión	J48	0.569	0.935	75.20	0.569	0.824	0.696	0.785
Redes Bayesianas	BayesNet BayesChaid	0.561	0.946	75.34	0.561	0.864	0.695	0.828
	BayesNet ByNet	0.471	0.971	72.11	0.471	0.86	0.628	0.745
	BayesNet K2	0.561	0.946	75.34	0.561	0.864	0.695	0.828
	NaiveBayes	0.558	0.945	75.17	0.558	0.864	0.692	0.828
Redes neuronales	MLP	0.671	0.821	74.58	0.671	0,848	0.725	0.805
Métodos basados en instancias	IBK k=1	0.679	0.807	74.30	0.679	0.833	0.726	0.799
	IBK k=5	0.625	0.86	74.26	0.625	0.848	0.708	0.807
	IBK k=10	0.603	0.873	73.78	0.603	0.848	0.697	0.808
Máquinas de soporte vectorial	SVM	0.505	0.977	74.09	0.505	0.730	0.661	0.741
Multiclasificador	(Randon Forest)	0.684	0.821	75.24	0.684	0.856	0.734	0.819

Anexo 4. Resultados del Experimento # 5.

Tabla A.4 Resultados de algoritmos de clasificación aplicados al conjunto de datos obtenido después de realizar el Experimento # 2 y aplicar el filtro *Neighborhood Cleaning Rule.*

Algoritmos de Clasificación		**RVP**	**RVN**	**Exactitud**	**Recall**	**Área PRC**	**F-Measure**	**Área ROC**
Árboles de decisión	J48	0.806	0.845	82.46	0.806	0.895	0.828	0.873
Redes Bayesianas	BayesNet Bayes Chaid	0.639	0.915	77.07	0.639	0.904	0.745	0.872
	BayesNet ByNet	0.538	0.936	72.74	0.538	0.85	0.674	0.76
	BayesNet K2	0.639	0.915	77.07	0.639	0.904	0.745	0.872
	NaiveBayes	0.551	0.942	73.76	0.551	0.870	0.687	0.838
Redes neuronales	MLP	0.757	0.821	78.71	0.757	0.909	0.788	0.880
Métodos basados en instancias	IBK k=1	0.909	0.907	90.79	0.909	0.901	0.912	0.916
	IBK k=5	0.827	0.871	84.78	0.827	0.924	0.851	0.920
	IBK k=10	0.839	0.805	82.27	0.839	0.918	0.832	0.903
Máquinas de soporte vectorial	SVM	0.665	0.886	77.03	0.665	0.751	0.752	0.776
Multiclasificador	(Randon Forest)	0.850	0.885	86.71	0.850	0.947	0.870	0.933

Anexo 5. Resultados del Experimento # 6.

Tabla A.5 Resultados de algoritmos de clasificación aplicados al conjunto de datos obtenido después de realizar el Experimento # 2 y aplicar el filtro *Wilson Editing.*

Algoritmos de Clasificación		**RVP**	**RVN**	**Exactitud**	**Recall**	**Área PRC**	**F-Measure**	**Área ROC**
Árboles de decisión	J48	0.893	0.986	96.30	0.893	0.932	0.923	0.952
Redes Bayesianas	BayesNet Bayes Chaid	0.87	0.976	95.02	0.87	0.952	0.896	0.97
	BayesNet ByNet	0.766	0.995	93.83	0.766	0.9	0.86	0.908
	BayesNet K2	0.87	0.976	95.02	0.87	0.952	0.896	0.97
	NaiveBayes	0.867	0.956	93.36	0.867	0.919	0.866	0.963
Redes neuronales	MLP	0.880	0.986	96.01	0.880	0.956	0.967	0.967
Métodos basados en instancias	IBK k=1	0.961	0.992	98.45	0.961	0.954	0.969	0.977
	IBK k=5	0.902	0.993	97.03	0.902	0.974	0.938	0.985
	IBK k=10	0.883	0.994	96.65	0.883	0.971	0.929	0.985
Máquinas de soporte vectorial	SVM	0.861	0.967	94.06	0.861	0.805	0.878	0.914
Multiclasificador	(Randon Forest)	0.908	0.992	97.11	0.908	0.980	0.940	0.980

Anexo 6. Resultados del Experimento # 7.

Tabla A.6 Resultados de algoritmos de clasificación aplicados al conjunto de datos obtenido después de realizar el Experimento # 2 y aplicar el filtro *One-Sided-Selection.*

Algoritmos de Clasificación		**RVP**	**RVN**	**Exactitud**	**Recall**	**Área PRC**	**F-Measure**	**Área ROC**
Árboles de decisión	J48	0.809	0.586	73.70	0.809	0.896	0.806	0.808
Redes Bayesianas	BayesNet Bayes Chaid	0.634	0.887	71.58	0.634	0.920	0.751	0.820
	BayesNet ByNet	0.836	0.419	70.08	0.836	0.9	0.791	0.772
	BayesNet K2	0.634	0.887	71.58	0.634	0.920	0.751	0.820
	NaiveBayes	0.545	0.929	66.88	0.545	0.903	0.690	0.789
Redes neuronales	MLP	0.811	0.497	70.93	0.811	0.908	0.791	0.791
Métodos basados en instancias	IBK k=1	0.679	0.807	74.40	0.679	0.833	0.726	0.799
	IBK k=5	0.625	0.860	74.26	0.625	0.848	0.708	0.807
	IBK k=10	0.603	0.873	73.78	0.603	0.848	0.697	0.808
Máquinas de soporte vectorial	SVM	0.505	0.977	74.09	0.505	0.730	0.661	0.741
Multiclasificador	(Randon Forest)	0.684	0.821	75.24	0.684	0.856	0.734	0.819

Anexo 7. Resultados del Experimento # 8.

Tabla A.7 Resultados de algoritmos de clasificación aplicados al conjunto de datos obtenido después de realizar el Experimento # 2 y aplicar el filtro *SMOTE- TomekHybrid*.

Algoritmos de Clasificación		**RVP**	**RVN**	**Exactitud**	**Recall**	**Área PRC**	**F-Measure**	**Área ROC**
Árboles de decisión	J48	0.761	0.82	78.61	0.761	0.879	0.802	0.854
Redes Bayesianas	BayesNet Bayes Chaid	0.632	0.917	75.43	0.632	0.945	0.746	0.855
	BayesNet ByNet	0.518	0.966	71.05	0.518	0.930	0.671	0.757
	BayesNet K2	0.632	0.917	75.43	0.632	0.945	0.746	0.855
	NaiveBayes	0.545	0.955	72.10	0.545	0.815	0.690	0.877
Redes neuronales	MLP	0.716	0.777	74.22	0.716	0.838	0.760	0.897
Métodos basados en instancias	IBK k=1	0.918	0.854	89.10	0.918	0.873	0.905	0.890
	IBK k=5	0.835	0.822	82.90	0.835	0.917	0.848	0.905
	IBK k=10	0.828	0.760	79.86	0.828	0.916	0.824	0.887
Máquinas de soporte vectorial	SVM	0.640	0.885	74.52	0.640	0.769	0.741	0.763
Multiclasificador	(Randon Forest)	0.872	0.854	86.42	0.872	0.957	0.880	0.939

Anexo 8. Resultados del Experimento # 9.

Tabla A.8 Resultados de algoritmos de clasificación aplicados al conjunto de datos obtenido después de realizar el Experimento # 2 y aplicar el filtro *CNN + Tomek links*.

Algoritmos de Clasificación		**RVP**	**RVN**	**Exactitud**	**Recall**	**Área PRC**	**F-Measure**	**Área ROC**
Árboles de decisión	J48	0.944	0.229	78.86	0.944	0.932	0.875	0.932

Redes Bayesianas	BayesNet Bayes Chaid	0.669	0.791	69.55	0.669	0.945	0.775	0.806
	BayesNet ByNet	1	0	78.32	1	0.9	0.878	0.749
	BayesNet K2	0.669	0.791	69.55	0.669	0.945	0.775	0.806
	NaiveBayes	0.537	0.932	62.25	0.537	0.932	0.690	0.773
Redes neuronales	MLP	0.938	0.256	79.03	0.938	0.940	0.875	0.795
Métodos basados en instancias	IBK k=1	0.885	0.593	82.13	0.885	0.898	0.886	0.767
	IBK k=5	0.898	0.214	75	0.898	0.914	0.849	0.750
	IBK k=10	0.945	0.106	76.30	0.945	0.932	0.862	0.778
Máquinas de soporte vectorial	SVM	0.985	0.045	78.15	0.985	0.768	0.876	0.515
Multiclasificador	(Randon Forest)	0.944	0.229	78.86	0.944	0.932	0.875	0.788

Anexo 9. Resultados del Experimento # 10.

Tabla A.9 Resultados de algoritmos de clasificación aplicados al conjunto de datos obtenido después de realizar el Experimento # 2 y aplicar el filtro *Smote + ENN*.

Algoritmos de Clasificación		**RVP**	**RVN**	**Exactitud**	**Recall**	**Área PRC**	**F-Measure**	**Área ROC**
Árboles de decisión	J48	0.818	0.917	87.06	0.818	0.933	0.856	0.912
Redes bayesianas	BayesNet BayesChaid	0.694	0.948	82.86	0.694	0.910	0.792	0.888
	BayesNet ByNet	0.604	0.991	80.99	0.604	0.850	0.749	0.838
	BayesNet K2	0.694	0.948	82.86	0.694	0.910	0.792	0.888
	NaiveBayes	0.663	0.963	82.24	0.663	0.887	0.868	0.868
Redes neuronales	MLP	0.702	0.963	84.09	0.702	0.908	0.806	0.885
Métodos basados en instancias	IBK k=1	0.832	0.840	83.64	0.832	0.782	0.827	0.842
	IBK k=5	0.800	0.939	87.39	0.800	0.913	0.856	0.900
	IBK k=10	0.785	0.912	85.26	0.785	0.915	0.833	0.898
Máquinas de soporte vectorial	SVM	0.702	0.963	84.09	0.702	0.778	0.806	0.885
Multiclasificador	(Randon Forest)	0.818	0.917	87.06	0.818	0.933	0.856	0.912

Anexo 10. Resultados del Experimento # 11.

Tabla A.10 Resultados de algoritmos de clasificación aplicados al conjunto de datos obtenido después de realizar el Experimento # 2 y aplicar el filtro *SMOTE_RSB*.

Algoritmos de Clasificación		**RVP**	**RVN**	**Exactitud**	**Recall**	**Área PRC**	**F-Measure**	**Área ROC**
Árboles de decisión	J48	0.632	0.912	77.22	0.632	0.842	0.735	0.826
	BayesNet Bayes Chaid	0.580	0.921	75.05	0.580	0.868	0.699	0.832
	BayesNet Bayes Chaid	0.535	0.954	74.44	0.535	0.800	0.677	0.76
	BayesNet K2	0.580	0.921	75.05	0.580	0.868	0.699	0.832
	NaiveBayes	0.539	0.944	74.18	0.539	0.839	0.676	0.801
Redes neuronales	MLP	0.588	0.911	74.94	0.588	0.861	0.701	0.824

Métodos basados en instancias	IBK k=1	0.820	0.735	77.76	0.820	0.728	0.787	0.783
	IBK k=5	0.760	0.789	77.46	0.760	0.850	0.771	0.850
	IBK k=10	0.752	0.766	75.90	0.752	0.864	0.757	0.844
Máquinas de soporte vectorial	SVM	0.602	0.909	75.57	0.602	0.722	0.711	0.756
Multiclasificador	(Randon Forest)	0.775	0.809	79.20	0.775	0.896	0.788	0.883

Anexo 11. Figuras de las plamta *Arabidopsis Thaliana*.

Figura A.11.1 Lámina de *Arabidopsis thaliana* con ilustraciones de las diferentes estructuras de la planta.

Figura A.11.2 Arabidopsis thaliana en el hábitat natural

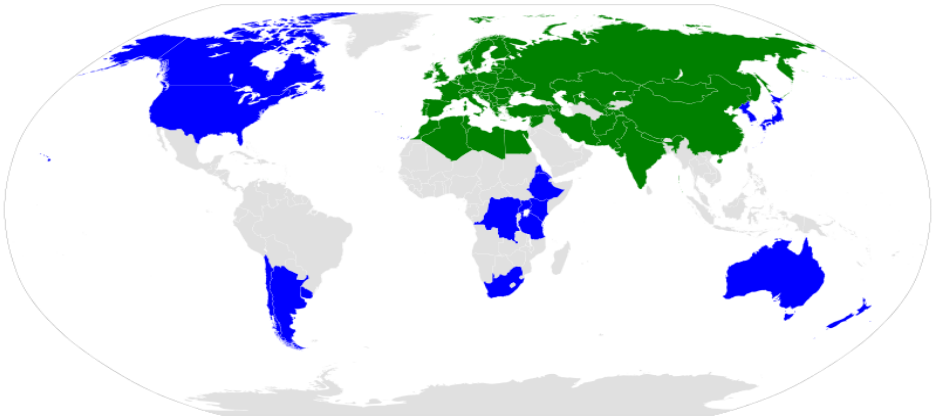
Figura A.11.3 Inflorescencias

Figura A.11.5 Imagen tomada con un microscopio electrónico de barrido de un tricoma de *Arabidopsis thaliana*.

Figura A.11.7 Hojas del tallo diferentes a las basales.

Figura A.11.4 Detalle de las hojas basales más grandes.

Figura A.11.6 Fruto.



Países donde la *Arabidopsis Thaliana* es nativa.

Países donde la *Arabidopsis Thaliana* es naturalizada.

Países donde la *Arabidopsis Thaliana* is no se encuentra.

Figura A.11.7 Figura donde se muestra la Ubicación geográfica de la planta