UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 6

TÍTULO: **Propuesta de algoritmos para reestructurar Muestras Desbalanceadas.**

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

AUTORES: Adrián Valdés Cabrera.

Joan Maykel Rojas Corvea.

TUTORES: Dr. Ramón Carrasco Velar.

Msc. Yaikiel Hernández Díaz.

Ciudad de La Habana, Cuba

Junio, 2010

" En dos palabras se puede resumir el éxito de la vida: sigue adelante."

-Robert Frost-

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Adrián Valdés Cabrera.

_____

Firma del Autor

Joan Maykel Rojas Corvea

_____

Firma del Autor

Dr. Ramón Carrasco Velar

_____

Firma del Tutor

Msc. Yaikiel Hernández Díaz

_____

Firma del Tutor

DATOS DE CONTACTO

Autores:

Adrián Valdés Cabrera email: avcabrera@estudiantes.uci.cu

Joan Maykel Rojas Corvea email: jmrojas@estudiantes.uci.cu

Tutores:

Ramón Carrasco Velar email: rcarrasco@uci.cu

Yaikiel Hernández Díaz email: yhernandezd@uci.cu

AGRADECIMIENTOS

De Adrián:

- *A mis dos tutores, Carrasco y Yaikiel, por toda su ayuda brindada.*
- *A una persona que oficialmente no aparece como mi tutor, pero siempre lo considere un tutor más, gracias Julio Omar.*
- *A Rachid y a Acro.*
- *A todo el que de cierta forma me ayudó, gracias.*

DEDICATORIA

De Adrián:

- *A todos mis amigos, en especial a Sierrita, a la Yula, a la Imis, al pekeñin, a Edel, a todos ustedes.*

- *A mis abuelos, Nidia y Dago, por su constante preocupación y por haber confiado siempre en mí.*
- *A mis tíos, Ileana y Panchi.*
- *A mis hermanas, Diana y Thais, y a mi cuñado el bocado.*
- *A mis padres, que hicieron de mí la persona que soy.*

  - *Mi papá, que aunque no se lo diga tiene que darse cuenta que "esto" tengo que haberlo heredado de alguien, y que me siento orgulloso de él.*
  - *Mi mamá, que es lo más grande que tengo en este mundo y esta tesis en especial se la dedico a ella.*

- *A toda mi familia en general, los quiero.*

RESUMEN

La aplicación de técnicas de inteligencia artificial a problemas reales ha traído consigo una serie de retos que previamente no habían sido considerados como relevantes. Uno de ellos es el problema de las clases desbalanceadas, presente cuando se tienen muchos ejemplos de una clase, pero muy pocos de otra. Este tema ha cobrado recientemente gran interés en la comunidad científica, debido a que el problema de las clases desbalanceadas es relativamente común en una gran cantidad de aplicaciones reales, y que los algoritmos de aprendizaje actuales tienen pobres desempeños. La presente investigación propone la implementación de algoritmos de rebalanceo para ser aplicados a las muestras desbalanceadas de la Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos (alasGRATO) y así obtener predicciones satisfactorias de los modelos generados. Se seleccionaron tres algoritmos a ser implementados: SMOTE, SMOTE + Tomek Links y SMOTE + ENN; resultando un sistema que fue probado con dos ensayos reales (fragmentos y mespectrales), mediante el cual se comprobó la efectividad de los algoritmos. Se pudo observar que los algoritmos híbridos ofrecen mejores resultados; los cuales para la muestra de fragmentos se encuentran entre el 62% y 63% y para la muestra de mespectrales entre 76% y 77%, con el algoritmo SMOTE + Tomek – Links; mientras que con el algoritmo SMOTE + ENN, para la muestra fragmentos se encuentra entre 62% y 64% y para la muestra de mespectrales entre 76% y 77%. Se presenta el diseño de clases de la aplicación desarrollada, así como el uso de varios patrones de diseño, como contribución a su posterior inclusión en la plataforma.

PALABRAS CLAVE: inteligencia artificial, clases desbalanceadas, algoritmos de aprendizaje, predicciones, muestras desbalanceadas.

ÍNDICE

INTRODUCCIÓN

Desde la segunda mitad del siglo pasado la farmacología ha encaminado sus avances para dar respuesta en el marco de la salud a problemas y enfermedades que han surgido. Esta disciplina científica tiene un gran número de funciones, pero la más importante de todas es la fabricación de medicamentos de alta calidad para la preservación de la salud de la especie humana y de otros animales.

En cualquier país, el gasto farmacéutico constituye un gran por ciento del gasto total de la salud y se necesitan medicamentos de buena calidad para mejorar muchos programas de salud y para la cura de diversas enfermedades. Aunque también existen otras enfermedades, como son el cáncer y el Virus de Inmunodeficiencia Humana (VIH), que no pueden curarse pero para las cuales se han creado medicamentos que alivian las dolencias de las mismas. En ambos casos muchos países no cuentan con el fondo monetario necesario para la obtención de dichos medicamentos. Por ejemplo: naciones de Norteamérica, Europa y Japón consumen el 80 % de los medicamentos, con solo el 14,9 % de la población mundial; mientras que el resto del planeta con el 85,1 % de la población tiene acceso solo al 20 % de los fármacos (1). Como se puede ver, la industria farmacéutica capitalista tiene como principal objetivo maximizar las ganancias y es poco probable que las organizaciones orientadas a la rentabilidad, dediquen muchos esfuerzos para solucionar los problemas de salud de los países en desarrollo, donde viven las dos terceras partes de la población mundial y donde los gobiernos no pueden asumir el gasto en medicamentos que requieren sus pueblos.

La Universidad de las Ciencias Informáticas (UCI), en específico la Facultad 6, de bioinformática, desarrolla diversos proyectos en conjunto con centros científicos de nuestro país. Uno de ellos, es el proyecto CITMA (Ministerio de Ciencia Tecnología y Medio Ambiente) de investigación-desarrollo-formación denominado: Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos (alasGRATO). La plataforma alasGRATO cuenta con una base de datos de mediano tamaño compuesta por moléculas y sus descriptores asociados, estos son utilizados por los métodos de inteligencia artificial implementados en la plataforma para la predicción de actividad biológica asociando esta a la estructura química. Sin embargo, los compuestos evaluados para cada bioensayo contenido en dicha base de datos, presentan como norma un alto grado de desbalance entre la cantidad de sustancias

reportadas como activas e inactivas. Esto constituye una dificultad para la predicción de actividad biológica por parte de la plataforma.

Teniendo en cuenta lo analizado, se plantea como **Problema científico:** ¿Cómo reestructurar las muestras desbalanceadas para la plataforma alasGRATO?

Por tanto, se define como **Objeto de estudio:** La reestructuración de muestras desbalanceadas. Que específicamente aborda como **Campo de acción:** Los algoritmos de muestreo aplicados a la reestructuración de muestras desbalanceadas.

Por lo que se adopta como **Objetivo general:** Proponer algoritmos para el trabajo con muestras desbalanceadas, que permitan generar modelos para la base de conocimientos del proyecto alasGRATO. Para dar cumplimiento al objetivo general se definen como **objetivos específicos:**

- Identificar algoritmos y metodologías aplicables.
- Implementar los algoritmos seleccionados.
- Validar los algoritmos implementados.

Para lograr los objetivos específicos se trazaron las siguientes **tareas:**

- ✓ Revisión del estado del arte acerca de los algoritmos y metodologías aplicables para la reestructuración de muestras desbalanceadas.
- ✓ Identificación de algoritmos de muestreo aplicables a la reestructuración de muestras desbalanceadas.
- ✓ Análisis de los algoritmos identificados.
- ✓ Implementación de los algoritmos seleccionados.
- ✓ Validación de los algoritmos implementados a través de pruebas con datos reales.
- ✓ Realización de pruebas no paramétricas para la comparación de los algoritmos.
- ✓ Construcción de un *plug-in* con los algoritmos seleccionados para la versión 1.0 de la plataforma alasGRATO.

Como aporte práctico se espera brindar un plug-in que reestructure las muestras desbalanceadas.

El trabajo de diploma se encuentra estructurado de la siguiente manera: resumen, introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos.

Capítulo 1: Fundamentación Teórica, se explica y debate el problema de las muestras desbalanceadas. Se plantean algoritmos con sus características, que se usan en el mundo para la reestructuración de los conjuntos imbalanceados.

Capítulo 2: Métodos y Materiales, se presentan los algoritmos seleccionados para la reestructuración de las muestras desbalanceadas, de estos se explican características referentes a su implementación. Además, se mencionan y explican las herramientas y metodologías que se emplearán para el desarrollo de la investigación.

Capítulo 3: Resultados y Discusión, se presenta un modelo conceptual del sistema para lograr una mayor claridad del sistema implementado. Se muestra el uso de los patrones de diseño utilizados y se realiza la validación de los algoritmos a través de pruebas con datos reales.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo, primeramente se mencionan algunos conceptos básicos para comprender mejor el problema del desbalance. Seguidamente se explica dicho problema y el tratamiento del mismo. En la sección *1.4 Algoritmos de Rebalanceo*, se hace una breve presentación de los métodos de rebalanceo identificados durante la investigación. Para la clasificación supervisada de estos métodos se usan las Máquinas de Soporte Vectorial (MSV), fundamentando su uso en el epígrafe 1.5. Posteriormente se realiza una revisión a la literatura existente sobre el tema, donde se mencionan algunas publicaciones realizadas sobre el tratamiento del desbalance en el mundo, por la comunidad científica internacional. Para finalizar se presentan tres software vinculados al tema de los conjuntos imbalanceados a nivel mundial.

1.1 Conceptos Básicos

1.1.1 Minería de Datos

La minería de datos consiste en la extracción no trivial de información que reside de manera implícita en los datos. Esta prepara, sondea y explora los mismos para sacar la información oculta en ellos.

La Minería de Datos se usa en diversas aplicaciones, por ejemplo: en el *marketing* para la publicidad dirigida y los estudios de competencia; en la meteorología para mejorar las predicciones; en la física para los datos de colisiones de partículas (búsqueda de patrones) y en la bioinformática para proyectos científicos como genoma humano, datos geofísicos y altas energías.

Mediante los modelos extraídos utilizando técnicas de minería de datos se aborda la solución a problemas de clasificación, segmentación y predicción, este último será el tratado en este trabajo. Las bases de la minería de datos se encuentran en el análisis estadístico y en la inteligencia artificial (2).

1.1.2 Inteligencia Artificial

Se denomina Inteligencia Artificial (IA) a la ciencia que desarrolla procesos que imitan a la inteligencia de los seres vivos.

En la bioinformática tiene múltiples aplicaciones, como son: en la predicción de la estructura y función de proteínas, en el modelado y simulación de redes de señalización biológica, en la identificación de blancos terapéuticos, en el análisis de secuencias biológicas, en el modelado de redes genéticas, en la minería de textos y extracción de información, entre otras.

Esta rama implica la creación de máquinas que puedan simular la inteligencia humana. También es capaz de incorporar a un sistema informático conocimiento o características propias del ser humano, mediante diferentes técnicas y algoritmos de IA (3).

1.1.3 Algoritmo de Aprendizaje

Se define como algoritmo de aprendizaje al conjunto de reglas bien definidas para la solución de un problema de aprendizaje, este aprendizaje del algoritmo será a través de ejemplos.

La idea clave del algoritmo de aprendizaje es encontrar el atributo más importante. El significado de *más importante* es que sea el más significativo con respecto a la clasificación de los ejemplos.

Los algoritmos de aprendizaje tienen diferentes aplicaciones, por ejemplo: en la robótica se usan para el reconocimiento táctil de objetos sólidos, en las redes neuronales para acelerar su aprendizaje y reducir el tiempo que le toma a la misma alcanzar el estado interno deseado y en este trabajo serán utilizados para balancear muestras de la plataforma alasGRATO (4).

1.2 El problema del desbalance

Los conjuntos de datos desbalanceados y concretamente los desbalanceados por clases, son conjuntos de datos cuya característica principal es que existe gran diferencia entre el número de casos de cada categoría. Cada uno de los datos usados para trabajar pertenecen a una categoría y el hecho de que

existan muchos más casos de una categoría que de otra puede dar lugar a problemas al intentar hallar Métodos de Aprendizaje. A cada categoría se le denomina clase y este tipo de conjunto de datos se puede presentar en problemas como la detección de intrusos, fraude bancario y como en el caso del presente trabajo, en la predicción de actividad biológica de compuestos orgánicos (5).

El desbalance, es uno de los problemas planteados en los temas de *Machine Learning* (Máquinas de Entrenamiento) y *Data Mining* (Minería de Datos). Los métodos de aprendizaje que se conocen necesitan aprender con tantos ejemplos como sea posible de cada clase, para luego poder generalizar de la mejor manera. Sin embargo, el problema en el aprendizaje viene dado cuando existe una clase o más, con un número superior de ejemplos con respecto al resto de las clases. Este es el caso del problema del desbalance que se va a focalizar en un problema de dos clases: la clase mayor que es la que contiene un número superior de ejemplos, que por lo general corresponde a los patrones negativos**,** y la clase menor, con un número muy inferior, generalmente de patrones positivos (5, p. 36).

1.3 Tratamiento del desbalance

El problema del desbalance se ha intentado tratar desde dos direcciones principales. La primera corresponde a continuar usando los métodos de aprendizaje tradicionales y rebalancear el conjunto de datos y/o realizar el aprendizaje realizando pequeñas modificaciones en los algoritmos clásicos, como proporcionar diferentes costes de aprendizaje dependiendo de la clase. La segunda dirección correspondería a encontrar nuevos métodos de aprendizaje que solucionasen un problema de desbalance imposible de trabajar con algoritmos clásicos (5, p. 36).

Así pues, los métodos utilizados para trabajar con los conjuntos desbalanceados, han sido divididos en grupos, según su estrategia. Primero, la Estrategia de Costes que trabaja con análisis de costos, esta técnica fue una de las más utilizadas en un primer momento y su gran problema es que los costos son generalmente desconocidos y difíciles de encontrar porque dependen del dominio en cuestión. Segundo, la Estrategia de Rebalanceo enfocada en modificar la distribución de los datos abordada a través del submuestreo y el sobremuestreo de las clases mayoritarias y minoritarias respectivamente, siendo esta última la estrategia a seguir en el trabajo para el manejo de las muestras desbalanceadas. También

existen nuevos métodos específicos de conjuntos imbalanceados, incluso otros para tratar problemas particulares, que ninguno de estos se incluyen en las dos estrategias anteriores (5, p. 49).

Volviendo a la estrategia de rebalanceo, lo mejor es empezar explicando las dos variantes principales: *oversampling* (sobremuestreo) y *undersampling* (submuestreo). El *oversampling* consiste en incrementar la cantidad de patrones de la clase-menor y el *undersampling* decrementar el número de patrones de la clase-mayor. El rebalanceo es la opción más usada para tratar con conjuntos de datos no balanceados, ya que se basa simplemente en intentar balancear de nuevo el conjunto de patrones para poder aplicar sobre él alguno de los métodos clásicos de aprendizaje. Sin embargo, estas variantes también tienen sus desventajas, por ejemplo el submuestreo aleatorio puede eliminar ejemplos de la clase mayoritaria potencialmente útiles y por otra parte, el sobremuestreo aleatorio puede llevar a introducir ruidos en la muestra y provocar además el sobreaprendizaje en el entrenamiento (5, p. 49).

1.4 Algoritmos de Rebalanceo

Existen diversos algoritmos de aprendizaje para tratar el problema de los conjuntos no balanceados. Este apartado se centrará en los algoritmos más importantes que usan técnicas de *oversampling*, *undersampling* o híbridos (sobremuestreo + submuestreo), que como se puede apreciar es una estrategia que combina a las dos primeras (6).

1.4.1 Condensed Nearest Neighbor Rule (CNN)

Regla Condensada del Vecino más Cercano (CNN), es usado para buscar subconjuntos consistentes en conjuntos de datos. Un subconjunto (ˆE) que pertenece a (E), es consistente con (E), si usando *1-Nearest Neighbor* (1-NN), (ˆE) clasifica correctamente los ejemplos en (E). El algoritmo para crear el subconjunto (ˆE) de (E) es un método de submuestreo. Este método toma ejemplos aleatorios de la clase mayoritaria y todos los ejemplos de la clase minoritaria y los pone en (ˆE). Luego de usar un 1-NN sobre los ejemplos de (ˆE) para clasificarlos en (E), cada ejemplo clasificado erróneamente en (E) es pasado a (ˆE).

La idea detrás de la creación de este subconjunto consistente, es eliminar los ejemplos de la clase mayoritaria que se encuentran lejos de la frontera de decisión, ya que este tipo de ejemplos podrían ser considerados menos relevantes para el aprendizaje.

Tiende a quedarse con los ejemplos cercanos a la frontera de decisión de cada clase, ya que estos son los más propensos a ser mal clasificados, aunque dependiendo del orden, deja muchas veces muchos más de los necesitados para lograr la consistencia(6, p. 23).

1.4.2 Edited Nearest Neighbors (ENN)

Regla Editada del Vecino más Cercano (ENN), este método de submuestreo fue creado por Wilson en 1972. Realiza el *undersampling* en forma contraria a CNN y tiende a eliminar más ejemplos. ENN trabaja por lotes, por lo que los objetos son marcados primero, y después se eliminan simultáneamente todos los marcados.

Tiene como ventaja que no depende del orden ni del azar, ya que utiliza los resultados de la aplicación de un clasificador. Trabaja bien con bases de datos pequeñas y con clases mezcladas.

Este método tiende a eliminar los objetos de la frontera y los aislados, debido a que éstos son generalmente mal clasificados. Como trabaja con información local, puede dejar conjuntos de objetos totalmente aislados de su clase (7).

1.4.3 Neighborhood Cleaning Rule (NCL)

Regla de Limpieza de Zona (NCL), usa el método de Wilson, Regla Editada del Vecino más Cercano (*Edited Nearest Neighbor Rule* (ENN)), para eliminar ejemplos de la clase mayoritaria. ENN elimina cualquier ejemplo cuya etiqueta de clase difiera de la clase de al menos dos de sus tres vecinos más cercanos. Para un problema de dos clases, el algoritmo NCL puede describirse de la siguiente forma: para cada ejemplo (Ei) en el conjunto de entrenamiento, se buscan sus tres vecinos más cercanos. Si (Ei) pertenece a la clase mayoritaria y la clasificación dada por sus tres vecinos más cercanos contradice la

clase original de (Ei), entonces (Ei) es eliminado. Si (Ei) pertenece a la clase minoritaria y sus tres vecinos más cercanos son de la clase mayoritaria, entonces dichos vecinos se eliminan (6, p. 23).

1.4.4 Tomek Links

Tomek Links es usado como método de submuestreo para la limpieza de conjuntos de datos no balanceados, eliminando ejemplos de ruido en la clase mayoritaria y otros que se encuentran en los bordes de ambas clases.

Este trabaja definiendo dos ejemplos, Ei y Ej, pertenecientes a clases diferentes; además D (Ei, Ej) que viene siendo la distancia existente entre Ei y Ej; y A (Ei, Ej) que va a ser llamado el par *Tomek Link* si no existiera un ejemplo El tal que D (Ei, El) < D (Ei, Ej) o D (Ej, El) < D (Ej, El). Si estos dos ejemplos formaran un *Tomek Link*, entonces uno de ellos será ruidoso o ambos estarán en el borde de sus clases (6, p. 23).

1.4.5 One-Sided Selection (OSS)

Selección de un Solo Lado (OSS), es un método de selección de instancias resultado de la aplicación del *Tomek Links*, seguido por la aplicación del CNN. Como ya se ha explicado, el *Tomek Links* se usa como un método de selección de instancias que elimina ejemplos ruidosos y fronterizos de la clase mayoritaria, y CNN pretende eliminar los ejemplos de la clase mayoritaria que están distantes de la frontera de decisión. La unión de ambos algoritmos da como resultado el OSS (6, p. 23).

1.4.6 CNN + Tomek Links

CNN + *Tomek Links*, es un método muy similar al OSS, pero el método para encontrar el subconjunto consistente es aplicado antes del Tomek Links, porque como Tomek Links es tan exigente computacionalmente se le aplica solo al subconjunto consistente que resulta del CNN y así la demanda computacional es menor (6, p. 23).

1.4.7 Synthetic Minority Oversampling Technique (SMOTE)

La Técnica de Sobremuestreo Sintético de la Clase Minoritaria (SMOTE), fue publicado por Chawla y es una técnica de *oversampling* que consiste en crear patrones artificiales en el espacio que hay entre un patrón dado de la clase-menor y otro de su misma clase. Para cada uno de los patrones de la clase-menor, se elige un segundo patrón de entre sus k vecinos más cercanos (aleatoriamente) y se crea un patrón artificial de la misma clase entre éstos dos. Dependiendo de la cantidad que se quiera hacer el *oversampling*, se crearán más o menos patrones (5, p. 41).

SMOTE (8) además presenta las siguientes características:

- Crea ejemplos sintéticos en lugar de hacer un sobremuestreo con reemplazo.
- Opera en el espacio de atributos (*feature space*), en lugar del espacio de datos (*data space*).
- Crea un ejemplo sintético a lo largo de los segmentos de línea que unen alguno o todos los k vecinos más cercanos de la clase minoritaria.
- Se eligen algunos de los k vecinos más cercanos de manera aleatoria (no se utilizan todos).
- SMOTE utiliza k = 5.

Es la técnica más utilizada para realizar el sobremuestreo, pero presenta los siguientes inconvenientes:

- Puede generar muchos ejemplos artificiales cuyas semillas son ejemplos con ruido.
- Al generar un nuevo ejemplo, interpola entre dos ejemplos de la clase minoritaria, sin embargo, pueden existir muchos ejemplos cercanos o inclusive entre ellos de la clase mayoritaria, generando modelos incorrectos.
- Solo funciona con variables continuas.
- No tiene una forma clara de decidir cuántos ejemplos generar.

1.4.8 SMOTE + Tomek Links

Aunque el sobremuestreo a los ejemplos de la clase minoritaria puede balancear la muestra, algunos problemas se han presentado en los datos o muestras analizadas que no se han podido resolver.

Frecuentemente grupos de datos no bien definidos, algunos de la clase mayoritaria, invaden el reducido espacio de la clase minoritaria. Para tratar este problema se propone aplicar después del sobremuestreo, el método *Tomek Links* como método de limpieza de datos, como se muestra en la Figura 1.2.

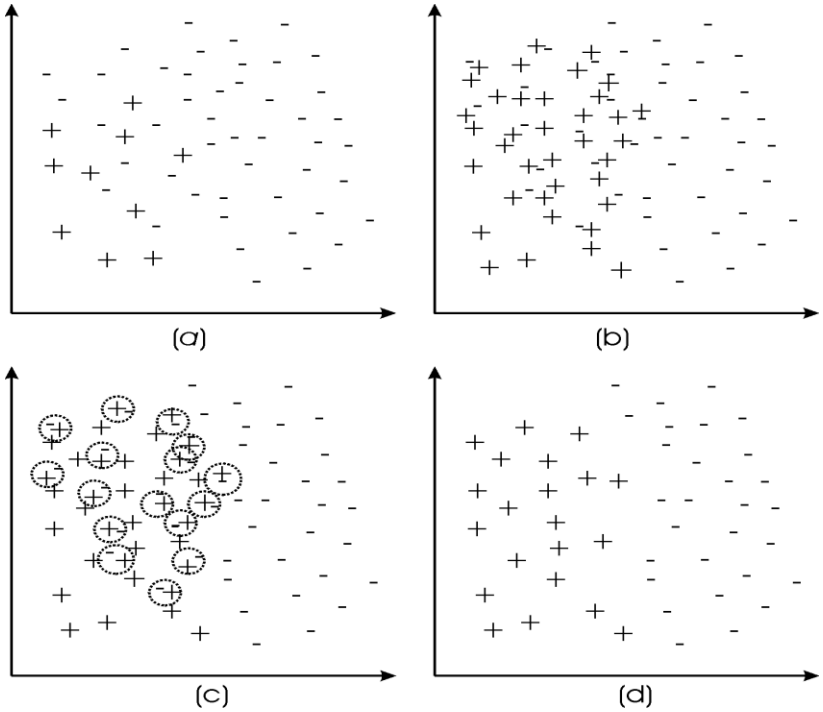


Figura 1.1: *Método SMOTE + Tomek Links: conjunto imbalanceado (a); aplicando sobremuestreo (b); identificando Tomek Links (c); ejemplos de ruido y de la frontera de decisión eliminados (d).*

Este método, SMOTE + *Tomek Links*, fue usado por primera vez para mejorar la clasificación de ejemplos en los problemas de anotación de proteínas para Bioinformática (6, p. 24).

1.4.9 SMOTE + ENN

La motivación detrás de SMOTE + ENN es similar a SMOTE + *Tomek Links*. ENN tiende a remover o eliminar más ejemplos que *Tomek Links*, por lo que se espera que ENN provea mucha más limpieza en los datos. Diferentemente de NCL, el cual es un método de submuestreo también, *Edited Nearest Neighbor Rule* (ENN) es usado para eliminar ejemplos de ambas clases. Es decir, cualquier ejemplo que es descartado o no es clasificado por sus 3 vecinos más cercanos es removido o eliminado (6, p. 24).

1.5 Clasificación supervisada: Máquinas de Soporte Vectorial (MSV)

A la hora de resolver los problemas de clasificación supervisada de los métodos de entrenamiento para el tratamiento del desbalance, los campos de la estadística y el aprendizaje automático han desarrollado diferentes técnicas, como por ejemplo: el Análisis Discriminante Lineal, las Redes Neuronales Artificiales, los Clasificadores K-NN, los Árboles de Clasificación, la Regresión Logística, la Inducción de Reglas y las Máquinas de Soporte Vectorial (9).

1.5.1 Máquinas de Soporte Vectorial (MSV)

El algoritmo Vector de Soporte (VS) es una generalización no-lineal del algoritmo Semblanza Generalizada, desarrollado en la Rusia de los años sesenta. Está firmemente enlazado a la Teoría del Aprendizaje Estadístico, la cual se desarrolló a finales de la década de los 80's por Vapnik y Chervonenkis. Por otra parte, esta Teoría del Aprendizaje Estadístico caracteriza propiedades del aprendizaje, habilitándole el poder de generalización de datos desconocidos. El desarrollo de los VS trae consigo el surgimiento de las MSV. Esta técnica de inteligencia artificial se enmarca dentro de las Redes Neuronales de aprendizaje supervisado, y es un clasificador eficiente cuando se desconoce la dependencia existente entre los datos y le permite la generalización de los mismos.

Las MSV pertenecen a la familia de clasificadores lineales. Mediante una función matemática denominada *kernel*, los datos originales se redimensionan para buscar una separabilidad lineal de los mismos. Una característica de las MSV es que realiza un mapeo de los vectores de entrada para determinar la linealidad o no de los casos, los cuales serán integrados a los multiplicadores de LaGrange para minimizar el Riesgo Empírico y la Dimensión de Vapnik-Chervonenkis. De manera general, las MSV permiten encontrar un hiperplano óptimo que separe las clases (9, p. 6).

Kernels

Las funciones *kernel* son funciones matemáticas que se emplean en las MSV. Estas funciones son las que le permiten a las MSV convertir un problema de clasificación no lineal en el espacio dimensional original, a un problema de clasificación lineal en un espacio dimensional mayor. Para que una función

pueda ser considerada candidata a *kernel*, debe ser continua, simétrica y positiva. Existen diversos *kernels,* entre los cuales se destacan: el lineal, el RBF (Función de Base Radial), el Polinomial y el Sinusoidal (9, p. 6).

1.5.2 Tipos de SVM

Para construir un hiperplano óptimo, las MSV emplean un algoritmo de entrenamiento iterativo que se emplea para minimizar una función de error. Acorde a la forma de la función de error, los prototipos de MSV se clasifican en dos grupos diferentes (9, p. 7).

Tipo 1 (también conocido como clasificación C-SVC)

Para este tipo de MSV, el entrenamiento involucra la minimización de la función error (Ec. 1):

$$\frac{1}{2}w^t w + C\sum_{i=1}^{N}\xi_i$$

Ecuación 1

Sujeto a las restricciones:

$$y(w^t\phi(x_i)+b) \geq 1-\xi_i \quad \text{y} \quad \xi_i \in \Re \geq 0 \text{ para } i=1,\ldots,N$$

Ecuación 2

Donde C es un valor entero empírico mayor que 1, w es el vector de coeficientes, b es una constante que caracteriza el hiperplano, ϕ es el *kernel* empleado y es un parámetro que permite manejar los datos de entrada no separables. El índice *i* etiqueta los casos de entrenamiento N. Debe tenerse en cuenta que es la clase etiquetada y es la variable independiente. El *kernel* se usa para transformar los datos de entrada al espacio de nuevas características. Cabe señalar que mientras mayor es C, mayor es el error penalizado. Por lo tanto, C deber ser escogido con cuidado para evitar el sobre entrenamiento (9, p. 7).

Tipo 2 (también conocido como clasificación nu-SVC)

A diferencia de la clasificación C-SVC, el modelo de clasificación nu-SVC minimiza la función error dada por la (Ec. 3).

$$\frac{1}{2}w^T w - v\rho + \frac{1}{N}\sum_{i=1}^{N}\xi_i$$

Ecuación 3

Sujeto a las restricciones:

$$y_i(w^T\phi(x_i) + b) \geq \rho - \xi_i, \xi_i \geq 0, i = 1, \ldots, N \quad \text{y} \quad \rho \geq 0$$

Ecuación 4

Para este tipo de máquina, el factor influyente es 1/N = nu, en donde N es la cantidad inicial de variables de entrenamiento (9, p. 8).

1.6 Consideraciones sobre el tratamiento de los conjuntos imbalanceados

Después de una larga lectura e investigación, se muestran algunas publicaciones que fundamentan la elección de los algoritmos con los que posteriormente se trabajará.

Japkowicz evaluó las técnicas *oversampling* y *undersampling* para conjuntos artificiales de datos desbalanceados. En este trabajo fueron considerados dos métodos de *oversampling*: aleatorio, consistente en crear más patrones de la clase menor hasta igualar el número con los de la clase mayor; y focalizado, que consistía en crear más patrones de clase-menor que estuvieran en la frontera entre ambas clases. En cuanto al *undersampling*, el método aleatorio funciona de la misma manera que para el *oversampling*, pero esta vez disminuyendo el número de patrones de la clase mayor hasta igualarse con el

de la menor; y el focalizado, consistía en eliminar los patrones de la clase mayor que estuvieran más alejados de la frontera. La autora concluyó que tanto el *oversampling* como el *undersampling* eran efectivos y, además, que ninguna técnica nueva y más sofisticada de crear o eliminar patrones daría mejores resultados, lo que muestra que son métodos a tener en cuenta para la realización del trabajo investigativo (10).

Por otra parte, Provost ya apuntó que el rebalanceo puede llevar a la pérdida o ganancia de información. La técnica de *undersampling* provoca una pérdida de información y el *oversampling* incrementa el conjunto de datos de entrenamiento, pero sin aportar ninguna ganancia de información o, incluso peor, simplemente replicando o introduciendo información falsa. Concluye que, considerando este hecho, la mejor estrategia de investigación se centra en qué situación los diferentes algoritmos de aprendizaje se muestran más eficaces (11). Siendo este uno de los objetivos de la investigación, seleccionar los algoritmos más eficaces para solucionar el problema de las muestras desbalanceadas en la Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos (alasGRATO).

Chawla, Bowyer, Hall y Kegelmeyer publicaron un método llamado Synthetic Minority Oversampling Technique (SMOTE). (12) Esta técnica consiste en la creación de nuevos patrones de la clase-menor. Para cada ejemplo de la clase minoritaria, nuevos ejemplos son creados aleatoriamente en el espacio que hay entre este y sus vecinos más próximos de la misma clase, balanceando así las clases con un método que crea ejemplos sintéticos en lugar de hacer un sobremuestreo con reemplazo. SMOTE es uno de los métodos más utilizados hasta la actualidad por la comunidad científica internacional en el tratamiento de los conjuntos imbalanceados, por lo que se convierte en un algoritmo potencial a elegir para la solución del problema tratado en la investigación.

Barandela, Valdovinos, Sánchez y Ferri presentaron otro estudio donde combinan diferentes técnicas de rebalanceo para conjuntos desbalanceados. Los resultados son aplicados sobre varios conjuntos de datos del repositorio UCI y sus conclusiones son que, si el desbalance no es muy severo, el *undersampling* sobre la clase-mayor puede ser una buena opción, mientras que en caso contrario, el *oversampling* es una buena opción (13).

Hall propuso que es mejor combinar *oversampling* y *undersampling* que aplicar sólo *undersampling*, para conjuntos de datos grandes y con mucho desbalance (14). Por esta razón, en el epígrafe 1.4 se presentan también algoritmos híbridos como son el SMOTE + *Tomek - Links* y el SMOTE + ENN, métodos que combinan el submuestreo y el sobremuestreo.

Batista, Prati y Monard presentan una comparación entre varios métodos de rebalanceo usados directamente o como parte de un método de aprendizaje. Su trabajo concluye que el aplicar métodos combinados, como SMOTE + *Tomek - links* o SMOTE + ENN son aplicables cuando los datos son muy imbalanceados o cuando hay pocos ejemplos de la clase-menor. También en (2004) analizaron ambos métodos con respecto a la relación entre el error y el desbalance. La conclusión en este segundo artículo es que ambos métodos son buenos para trabajar con datos imbalanceados y que, además, pueden ayudar a extraer *small disjuncts* o pequeñas disyuntivas indeseables, que entorpecen y complican el proceso de aprendizaje (15).

Guiándose por las fundamentaciones de estos autores, se puede decir que hay 3 algoritmos a tener en cuenta, ellos son: SMOTE, SMOTE + *Tomek - Links* y SMOTE + ENN.

1.7 Programas vinculados a los conjuntos imbalanceados a nivel mundial

En la actualidad existen múltiples herramientas software de aprendizaje y minería de datos. Características que no tienen estas herramientas en muchos casos son: la inclusión de algoritmos de aprendizaje evolutivo y la inclusión de herramientas estadísticas para el análisis de algoritmos. A continuación se presentan tres herramientas software que incluyen los requerimientos anteriormente indicados y están vinculadas al problema del desbalance.

1.7.1 RapidMiner (Anteriormente YALE)

Es un software de código abierto para el análisis inteligente de datos, descubrimiento de conocimientos, minería de datos, aprendizaje automático, visualización, etc.; con numerosas características y funciones para las clases desbalanceadas. Constituye además un entorno de aprendizaje automático y de extracción de datos para todo tipo de experimentos. Permite que los experimentos sean realizados con un

gran número de variables arbitrarias, las cuales se escriben en archivos XML que son fácilmente creados con la interfaz gráfica de *RapidMiner*. Ofrece más de 400 operadores para los principales procedimientos de aprendizaje de máquinas, incluidos los de entrada, salida, pre procesamiento de datos y visualización de los mismos.

Está escrito en el lenguaje de programación Java y, por tanto, pueden trabajar en todos los sistemas operativos populares. También integra todos los sistemas de aprendizaje y de atributo de los evaluadores Weka (3, p. 18).

1.7.2 Keel

Es un software para evaluar la evolución de los algoritmos de minería de datos y problemas de regresión, entre ellos: clasificación, agrupamiento y patrón de la minería. Contiene una gran colección de algoritmos clásicos de extracción de conocimientos, técnicas de pre procesamiento (selección de instancias, selección de características, discretización, métodos de imputación de valores), Inteligencia Computacional de aprendizaje basado en algoritmos, incluido el estado evolutivo de algoritmos de aprendizaje basados en diferentes enfoques (*Pittsburgh*, *Michigan* y IRL) y modelos híbridos como sistemas difusos genéticos, redes neuronales evolutivas, etc. Nos permite realizar un análisis completo de cualquier modelo de aprendizaje en comparación con los existentes, incluido un módulo de prueba estadística para la comparación entre ellos.

El uso más común de esta herramienta para un investigador será la ejecución automatizada de los experimentos y el análisis estadístico de sus resultados. Esta herramienta no está diseñada para ofrecer un tiempo real del progreso de los algoritmos. Trabaja muy bien en ambiente distribuido de sistemas.

Fue diseñado con doble objetivo: la investigación y la educación. Cuenta con licencia comercial, lo que lo convierte Software propietario (16).

1.7.3 Weka

Es un paquete de software de Java para la extracción de conocimientos desde bases de datos, incluye además una recopilación de algoritmos de aprendizaje automático para tareas de minería de datos. Este software ha sido desarrollado en la universidad de Waikato (Nueva Zelanda) bajo licencia GPL (Licencia Pública General) lo cual ha impulsado que sea una de las suites más utilizadas en el área en los últimos años.

Este ave da nombre a una extensa colección de algoritmos de Máquinas de conocimiento desarrollados por la Universidad de Waikato (Nueva Zelanda) implementados en Java; útiles para ser aplicados sobre datos mediante las interfaces que ofrece o para embeberlos dentro de cualquier aplicación.

Además, Weka contiene las herramientas necesarias para realizar transformaciones sobre los datos, tareas de clasificación, regresión, *clustering* (agrupamiento), asociación y visualización. Está diseñado como una herramienta orientada a la extensibilidad por lo que añadir nuevas funcionalidades es una tarea sencilla (3, p. 19).

Sin embargo, y pese a todas las cualidades que Weka posee, tiene un gran defecto y este es la escasa documentación orientada al usuario que tiene junto a una usabilidad bastante pobre, lo que la hace una herramienta difícil de comprender y manejar sin información adicional. La licencia de Weka es GPL, lo que significa que este programa es de libre distribución y difusión. Además, ya que Weka está programado en Java, es independiente de la arquitectura, ya que funciona en cualquier plataforma sobre la que haya una máquina virtual Java disponible.

Una de las propiedades más interesantes de este software, es su facilidad para añadir extensiones y modificar sus métodos (9, p. 28).

1.8 Conclusiones

En este capítulo se realizó un estudio del desbalance para su mejor comprensión, analizando las diferentes estrategias en las que se agrupan los métodos, seleccionándose la Estrategia de Rebalanceo

para el desarrollo de la investigación. Dentro de los algoritmos identificados, los que mejores soluciones aportan son: el SMOTE, SMOTE + *Tomek* - *Links* y SMOTE + ENN que serán desarrollados en el presente trabajo de diploma. Se decide utilizar para la clasificación supervisada de los mismos las Máquinas de Soporte Vectorial (MSV).

CAPÍTULO 2: MÉTODOS Y MATERIALES

En este capítulo se presentan las muestras utilizadas para validar el funcionamiento de los algoritmos seleccionados y el procedimiento que emplean los mismos para balancearlas. Posteriormente se explican las herramientas, lenguajes y la metodología que se usará para el desarrollo de la investigación.

2.1 Muestras Utilizadas

Para la realización de la investigación se emplearon dos muestras de datos reales (fragmentos y mespectrales) tomadas de la base de datos del Centro Nacional de Información Biotecnológica (NCBI) de los Estados Unidos. Ambas muestras incluyen sustancias inorgánicas y sales que se eliminaron del análisis y son lo suficientemente diversas desde el punto de vista estructural. El índice IR representa la relación entre el numero de instancias de la clase mayoritaria y minoritaria, se considera que IR > 10 clasifica a una muestra como altamente desbalanceada. En ambos casos las muestras se clasifican como altamente desbalanceadas (17).

2.2 Procedimientos para balancear

La estrategia de rebalanceo puede realizarse mediante técnicas de *undersampling, oversampling* o híbridos. Ya se conoce que tanto el submuestreo como el sobremuestreo aplicados de forma aleatoria tienen sus desventajas. También utilizar algoritmos de forma individual presenta sus inconvenientes, incluso aplicar el potente algoritmo SMOTE. Otro procedimiento consiste en aplicar combinaciones de *undersampling* con *oversampling*, donde en casi todas estas combinaciones efectivas se encuentra presente el algoritmo SMOTE. El procedimiento a usar en esta tesis será la utilización del SMOTE con sus hibridaciones, este consiste en aplicar primeramente SMOTE y luego se le aplican los métodos de limpieza *Tomek Links* o ENN, de ahí el nombre de los algoritmos SMOTE + *Tomek Links* y SMOTE + ENN.

2.3 SMOTE con sus hibridaciones

2.3.1 SMOTE

El algoritmo SMOTE sobremuestrea la clase minoritaria con la creación de nuevos casos a partir de casos ya existentes. Los principales pasos en su ejecución son los siguientes:

1- Recibe como parámetro el por ciento de ejemplos a sobre-muestrear.
2- Calcula el número de ejemplos que tiene que generar.
3- Calcula los k vecinos más cercanos de los ejemplos de la clase minoritaria.
4- Genera los ejemplos siguiendo este proceso:
    - Para cada ejemplo de la clase minoritaria, elige aleatoriamente el vecino a utilizar para crear el nuevo ejemplo.
    - Para cada atributo del ejemplo a sobremuestrear, calcula la diferencia entre el vector de atributos muestra y el vecino elegido.
    - Multiplica esta diferencia por un número aleatorio entre 0 y 1.
    - Suma este último valor al valor original de la muestra.
    - Regresa el conjunto de ejemplos sintéticos.

Como se conoce SMOTE presenta algunos inconvenientes, para remediar los mismos se aplican métodos de limpieza (8, p. 11).

2.3.2 Métodos de limpieza

Los casos de una base de datos pueden ser divididos en cuatro categorías: casos ruidosos en cuanto a clases, casos límites, casos redundantes y casos seguros. Estas cuatro categorías se relacionan con el área de decisión, en la que cada caso es visto como un punto en el espacio R. La presencia de casos ruidosos en cuanto a clases, casos límites y casos redundantes no es recomendada ya que degrada el rendimiento de los clasificadores. A continuación, la figura 2.1 representa visualmente el método de limpieza de datos.

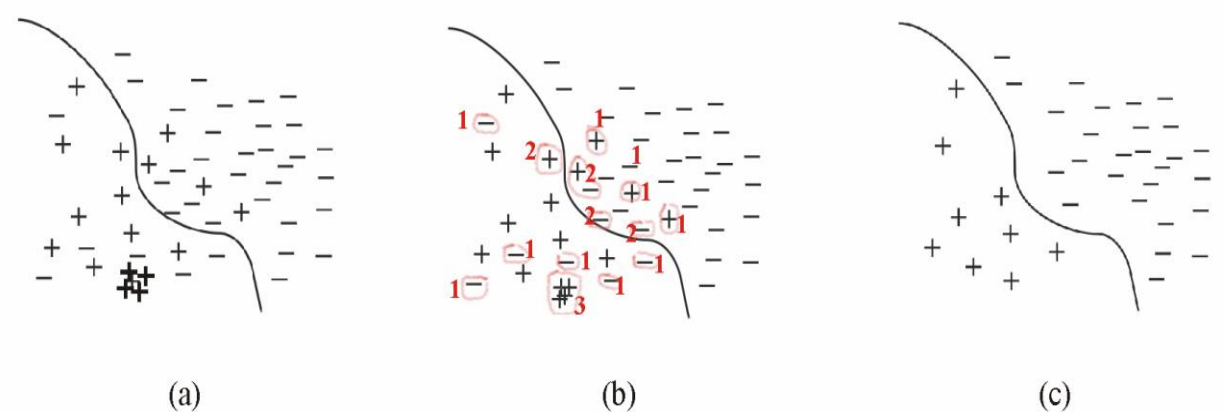


Figura 2.1: *Los datos originales (figura 2.1 a); los casos etiquetados de acuerdo a las categorías (figura 2.1 b), -1- casos ruidosos en cuanto a clases, -2- casos límite, -3- casos redundantes y los no etiquetados son los casos considerados seguros; los datos limpios con la retirada de los casos etiquetados de 1 a 3 (figura 2.1 c).*

Para esta limpieza de datos se seleccionaron dos algoritmos, ellos son: *Tomek Link* y ENN, a continuación se explican los mismos (7, p. 20).

2.3.2.1 Tomek Links

Este algoritmo puede ser utilizado para hacer *undersampling* a la clase mayoritaria o para hacer la limpieza de datos. (7, p. 21) A continuación se presenta los pasos para su ejecución.

1- Recibe como parámetro la variable tipo, la cual menciona la forma en que funcionará (¨limpieza¨ o ¨reducción¨).
2- Forma todos los pares de vecinos más cercanos que existan, a estos se les denomina par *Tomek – Links*.
3- Para cada par *Tomek – Links*, que las clases de sus vecinos sean diferentes.

- Si tipo = limpieza, elimina ambos vecinos.
- Si tipo = reducción, elimina el vecino que pertenece a la clase mayoritaria.

2.3.2.2 ENN

El algoritmo ENN aplica *undersampling* a un conjunto de entrada S. Un caso x_i del conjunto S, se elimina si y solo si es diferente a sus k vecinos más cercanos, utilizando el clasificador KNN (por lo general k = 3) (7, p. 21). Seguidamente se muestran los pasos para su ejecución.

1- Recibe como parámetro el número de vecinos a buscar para cada clase.
2- Calcula los k vecinos más cercanos de cada instancia.
3- Si la mayoría de los vecinos son de clase distinta, marca la instancia.
4- Elimina las instancias marcadas.

2.4 Herramientas y metodología utilizadas

Para lograr el objetivo general de la investigación se hace necesario el uso de una metodología que guíe el proceso de desarrollo, así como de herramientas y lenguajes que permitan el desarrollo del mismo. El presente trabajo está enmarcado dentro del proyecto “Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos” (alasGRATO), el cual cuenta con una arquitectura muy bien definida que se describe a continuación.

2.4.1 Metodología de desarrollo de software: OpenUP

La metodología seleccionada fue *OpenUP,* por ser un proceso iterativo para el desarrollo de software que es (18):

- Mínimo: Solo incluye el contenido del proceso fundamental.
- Completo: Puede ser manifestado como proceso entero para construir un sistema.
- Extensible: Puede ser utilizado como base para agregar o para adaptar más procesos.

También tiene múltiples beneficios como son:

- Ya que es apropiado para proyectos pequeños y de bajos recursos, permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito.
- Permite detectar errores tempranos a través de un ciclo iterativo.
- Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP (Proceso Unificado de Rational).
- Por ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas.

Sus principios son:

- Colaborar para alinear intereses y para compartir conocimiento.
- Balancear las prioridades para maximizar las necesidades de los *stakeholder* (clientes).
- Centrado en la Arquitectura.
- Desarrollo Iterativo.

2.4.2 Lenguaje de modelado: UML

El lenguaje de modelado seleccionado es UML (Lenguaje Unificado de Modelado), el cual está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo; permite modelar, construir y documentar los elementos que forman un producto de software que responde a un enfoque orientado a objetos. Este lenguaje fue creado por un grupo de estudiosos de la Ingeniería de Software formado por: Ivar Jacobson, Grady Booch y James Rumbaugh, en el año 1995.

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, aún cuando todavía no es un estándar oficial, está respaldado por el OMG (Grupo de Objeto de Gestión). Mediante UML es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código (18, p. 25).

La decisión de utilizar UML como notación para el desarrollo del software, está apoyada además en que se ha convertido en un estándar con muchas características favorables como las siguientes (3, p. 36):

- Permite modelar sistemas utilizando técnicas orientadas a objetos.

- Permite especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos.
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).
- Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.
- A pesar de tener gran expresividad esta notación es fácil de aprender.
- UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

2.4.3 Herramienta Case: Visual Paradigm

La herramienta CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadoras) seleccionada es *Visual Paradigm*, que pertenece a una compañía del mismo nombre y se encuentra entre las principales compañías de herramientas CASE. Tiene disponible distintas versiones: *Enterprise, Professional, Standard, Modeler, Personal y Community* (que es gratuita). La compañía facilita licencias especiales para fines académicos. Está diseñada para una amplia gama de usuarios que incluye a: ingenieros de software, analistas de sistemas, analistas de negocios y arquitectos de sistemas; interesados en la creación de grandes sistemas de software de manera confiable, a través del paradigma orientado a objetos.

Además, la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto y genera la documentación del proyecto automáticamente en varios formatos como Web o .PDF. También permite control de versiones; ingeniería inversa; generación de código; importación desde *Rational Rose*; exportación/importación XMI; generador de informes; editor de figuras; integración con *MS Visio* y *plug-ins*; integración con Eclipse, *Borland® JBuilder®, NetBeans IDE/Sun™ ONE, IntelliJ IDEA™, Oracle JDeveloper y BEA WebLogic Workshop™*. Las transiciones del análisis al diseño, y de éste a la implementación, están adecuadamente integradas dentro de la herramienta CASE, de manera que reduce

significativamente los esfuerzos de todas las etapas del ciclo de desarrollo de software. Cabe destacar igualmente su robustez, usabilidad y portabilidad (19).

2.4.4 Lenguaje de programación: Java

Como lenguaje de programación se escogió Java; el cual es un lenguaje de propósito general, concurrente, basado en clases y orientado a objetos. Su diseño fue concebido para que los programadores puedan lograr fluidez con el lenguaje y fue desarrollado por *Sun Mycrosystems* a principios de los años 90. El lenguaje Java ofrece la funcionalidad de un lenguaje potente y en sí mismo toma mucha de su sintaxis de C y C++, pero elimina muchas de sus características para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el *garbage collector* (reciclador de memoria dinámica o recolector de basura). No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello y como es un *thread* (hilo) de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de esta (19, p. 19).

Además de lo planteado hasta ahora, el lenguaje de programación Java también presenta otras características como son: que permite la ejecución de un mismo programa en múltiples sistemas operativos; incluye por defecto soporte para trabajo en red; está diseñado para ejecutar código en sistemas remotos de forma segura; es independiente de la plataforma en la que se ejecuta, esto significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware, tal como reza el axioma de Java, ‘‘*write once, run everywhere*‘‘ (“escrito una vez, corre donde sea”) (3, p. 34).

2.4.5 Entorno de desarrollo: Eclipse

Eclipse es un poderoso Entorno Integrado de Desarrollo (IDE), que permite la construcción de aplicaciones en Java. Admite la incorporación de otros *plug-ins* para obtener un mayor número de funcionalidades (3, p. 34).

Además presenta los siguientes beneficios:

- Es una herramienta de código abierto.
- Soporta herramientas que manipulan diferentes tipos de lenguajes, como por ejemplo: Java, C y C++.
- Corre en una gran cantidad de sistemas operativos, incluyendo Windows y Linux.
- Provee a los desarrolladores, herramientas (ej.- PDE) que facilitan la creación de *plug-ins*.

A los beneficios mencionados anteriormente, se suman estas características: la capacidad de ser soportado para distintas arquitecturas; control de versiones con *subversión*; resaltado de sintaxis; autocompletado; tabulador de un bloque de código seleccionado; asistentes (*wizards*) para la creación, exportación e importación de proyectos, para generar esqueletos de códigos (*templates*); permite la integración con la herramienta CASE Visual Paradigm (19, p 20).

Por estas razones fue seleccionado Eclipse como IDE para el desarrollo de la tesis.

CAPÍTULO 3: RESULTADOS Y DISCUSIÓN

En este capítulo se presenta un modelo conceptual o de dominio para un mejor entendimiento del sistema, además del diagrama de clases del diseño y los patrones utilizados en el mismo. En el epígrafe *Funcionamiento del Plug-in,* se explican las principales funcionalidades del mismo. Seguidamente se describe detalladamente la validación de los algoritmos implementados a través de pruebas con datos reales para comprobar el correcto funcionamiento de la herramienta y para concluir se le realiza una comparación estadística a dichos algoritmos.

3.1 Modelo de dominio o conceptual

Un modelo de dominio o conceptual es presentado como uno o más diagramas de clases, que contiene, no conceptos propios de un sistema de software sino la propia realidad física. Los modelos de dominio se emplean para capturar y expresar el entendimiento adquirido en un área como paso previo al diseño de un sistema, ya sea de software o de otro tipo.

El modelo de dominio puede ser tomado como el punto de partida para el diseño del sistema. Cuando se realiza la programación orientada a objetos, se supone que el funcionamiento interno del software va a imitar en alguna medida a la realidad, por lo que el mapa de conceptos del modelo de domino constituye una primera versión del sistema (20).

Las cualidades más significativas del esquema conceptual están orientadas a lograr según (21):

- Claridad y simplicidad: Significación no ambigua.
- Coherencia: Ausencia de contradicciones o confusión.
- Completitud: Sin buscar la exhaustividad, se representa lo esencial de los fenómenos.
- Fidelidad: Representación sin desviaciones y sin deformaciones.
- No redundancia: Sólo se representan elementos estrictamente necesarios, y únicamente una vez.

Por ocupar un rol protagónico en el desarrollo moderno de software y no encontrarse bien definidos los actores y fronteras del negocio, además de estar definido su uso en la metodología de software utilizada *OpenUP*, se utiliza en el presente trabajo el modelo de dominio o concepto.

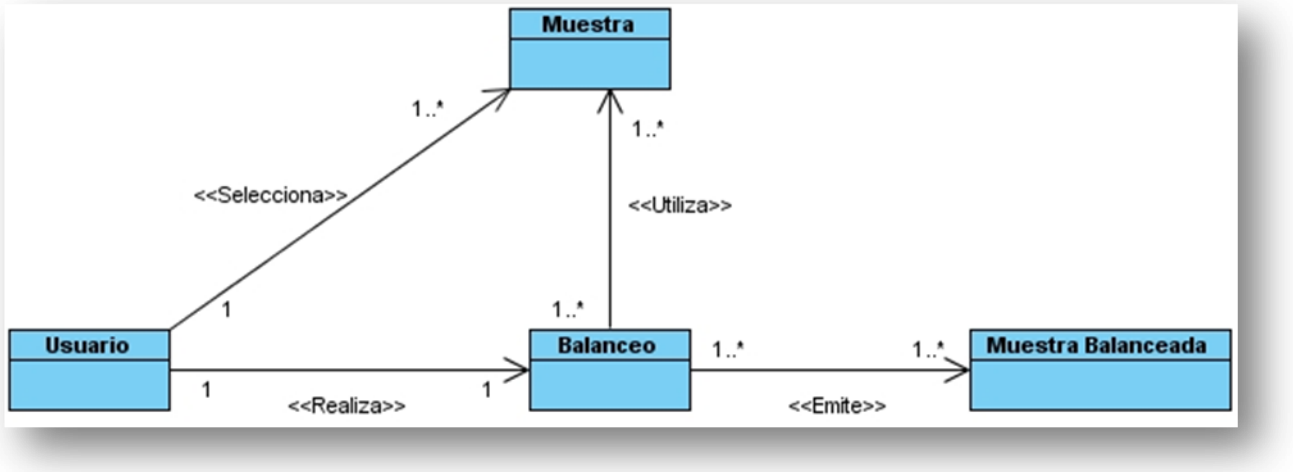


***Figura 3.1**: Modelo de dominio*

Conceptos asociados al modelo de dominio de la Figura 3.1:

- Usuario: Cliente o persona que interactúa con el sistema.
- Selecciona: Acción de tomar un fichero de un directorio raíz donde este se encuentre.
- Realiza: Acción de pulsar el botón Balancear para ejecutar el algoritmo de balanceo seleccionado.
- Balanceo: Aplicación informática que ejecuta los algoritmos de balanceo.
- Utiliza: Acción de aplicar los algoritmos de balanceo a la muestra seleccionada.
- Emite: Creación de nuevos ficheros como resultado del rebalanceo de la muestra.
- Muestra Balanceada: Fichero resultante del balanceo de la muestra seleccionada.

En la Figura 3.1 se muestra el modelo de dominio correspondiente a la investigación realizada. El mismo representa como el Usuario selecciona una Muestra para balancearla, identificada esta acción por el concepto Balanceo, que hace uso del concepto Muestra y emitiendo la Muestra Balanceada.

3.2 Patrones de Diseño de Software

Según (22), los patrones de diseño son soluciones simples a problemas específicos y comunes del diseño. Es una unidad de información nombrada, instructiva e intuitiva que agrupa exitosas soluciones, probadas en un problema recurrente dentro de un cierto contexto.

La utilización de patrones en el diseño de software nos proporciona ventajas como:

- Contribuir a reutilizar diseño, identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones. La importancia de la reutilización del diseño no es despreciable, ya que provee numerosas ventajas: reduce los esfuerzos de desarrollo y mantenimiento, mejora la seguridad, eficiencia y consistencia de nuestros diseños, y proporciona un considerable ahorro en la inversión.
- Mejorar (aumentar y elevar) la flexibilidad, modularidad y extensibilidad, factores internos e íntimamente relacionados con la calidad percibida por el usuario.
- Incrementar el vocabulario de diseño, ayudando a diseñar desde un mayor nivel de abstracción.

3.2.1 Patrones GRASP

El acrónimo GRASP se origina de las siglas en ingles de Patrones de Software para la Asignación General de Responsabilidades (*General Responsability Assignment Software Patterns*) (23). El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos.

Se pueden destacar cinco patrones principales que son (3, p. 43):

- Experto.
- Creador.
- Alta cohesión.

- Bajo acoplamiento.
- Controlador.

Y cuatro patrones GRASP adicionales que son:

- Fabricación Pura.
- Polimorfismo.
- Indirección.
- No hables con extraños.

Cada patrón permite que algunos aspectos de la estructura del sistema puedan cambiar independientemente de otros aspectos, además facilitan la reusabilidad, extensibilidad y mantenimiento del diseño, por eso se decide utilizar los siguientes patrones para el diseño de la herramienta implementada.

Experto

El patrón Experto en información es el principio básico de asignación de responsabilidades. Consiste en asignar la responsabilidad de creación de un objeto o la implementación de un método al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad (24).

***Figura 3.2**: Patrón Experto*

En la Figura 3.2 se ejemplifica el patrón Experto utilizado en el sistema, asignándosele la responsabilidad de ejecutar el método de balanceo SMOTE a la clase del mismo nombre siendo esta la clase que contiene la información necesaria para ejecutar esa acción.

Creador

Según (24), este patrón asigna a la clase B la responsabilidad de crear una instancia de la clase A si alguna de las siguientes premisas es cierta:

- B agrega los objetos de A
- B contiene los objetos de A
- B registra las instancias de los objetos de A.
- B tiene los datos de inicialización que serán enviados a A cuando este objeto sea creado.

Permite que el bajo acoplamiento sea soportado, lo que implica bajo mantenimiento y altas oportunidades de reutilización.

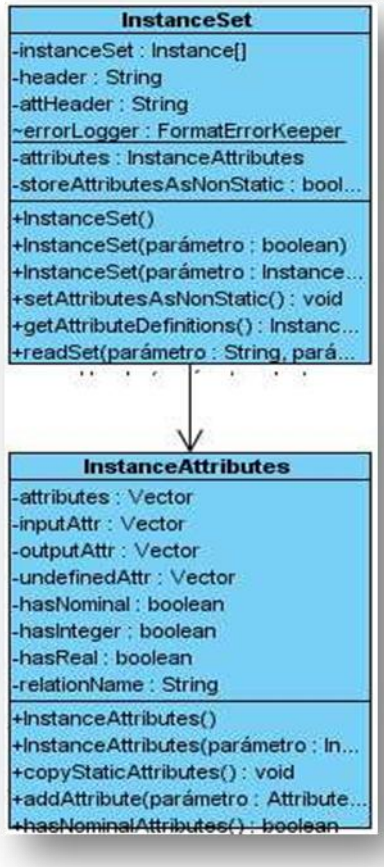


***Figura 3.3**: Patrón Creador*

La figura 3.3 representa la forma en que la clase InstanceSet utiliza una instancia de la clase InstanceAttributes reflejándose en el atributo attributes de tipo InstanceAttributes.

Alta Cohesión

La alta cohesión es una medida con la que se relacionan las clases y el grado de responsabilidades de un elemento. Este patrón mejora la claridad y facilidad con que se entiende el diseño, simplifica el mantenimiento y las mejoras de funcionalidad, soportando mayor capacidad de reutilización (3, p. 46).

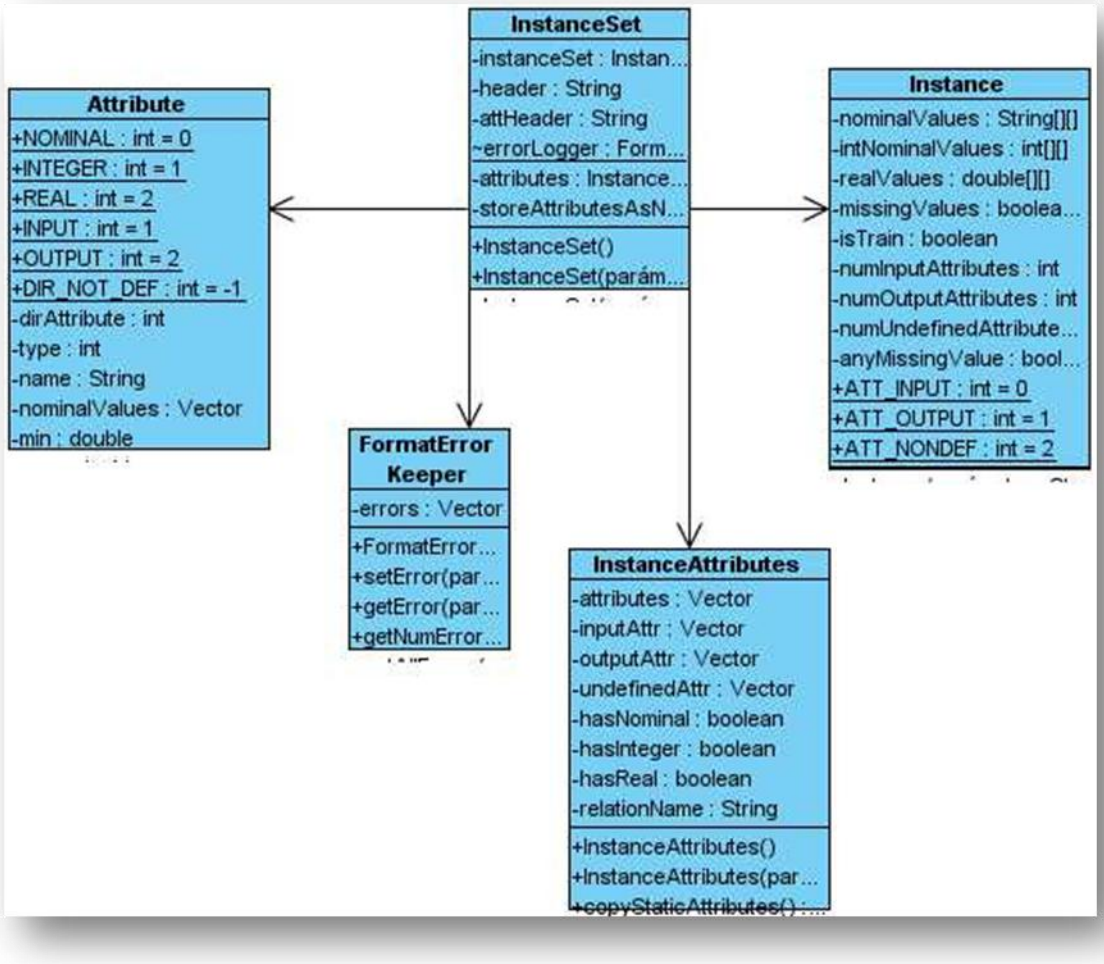


***Figura 3.4**: Patrón Alta Cohesión*

En la Figura 3.4 se evidencia como la clase InstanceSet asigna a cada clase sus responsabilidades, para así no afectar la claridad o facilidad con que se entiende el diseño, además de simplificar el mantenimiento y las mejoras de funcionalidad del sistema.

Controlador

Este patrón es responsable de asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades como validación y seguridad. El

controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión (25).

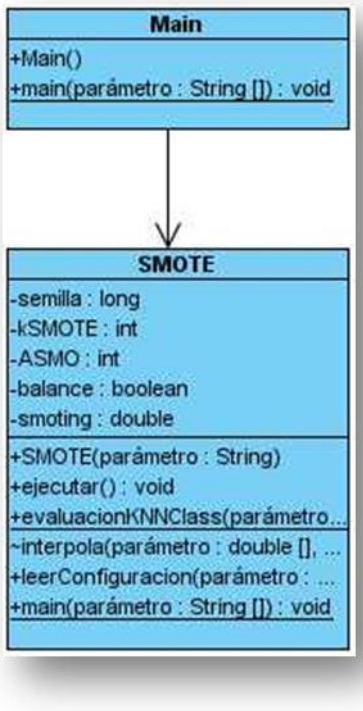


***Figura 3.5**: Patrón Controlador*

En la Figura 3.5 se evidencia como la clase Main controla la acción de ejecutar el algoritmo SMOTE mediante la llamada del método ejectar() implementado en la clase SMOTE.

Bajo Acoplamiento

Es la idea de tener las clases lo menos relacionadas entre sí. De esta forma, si se produce una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases (26).

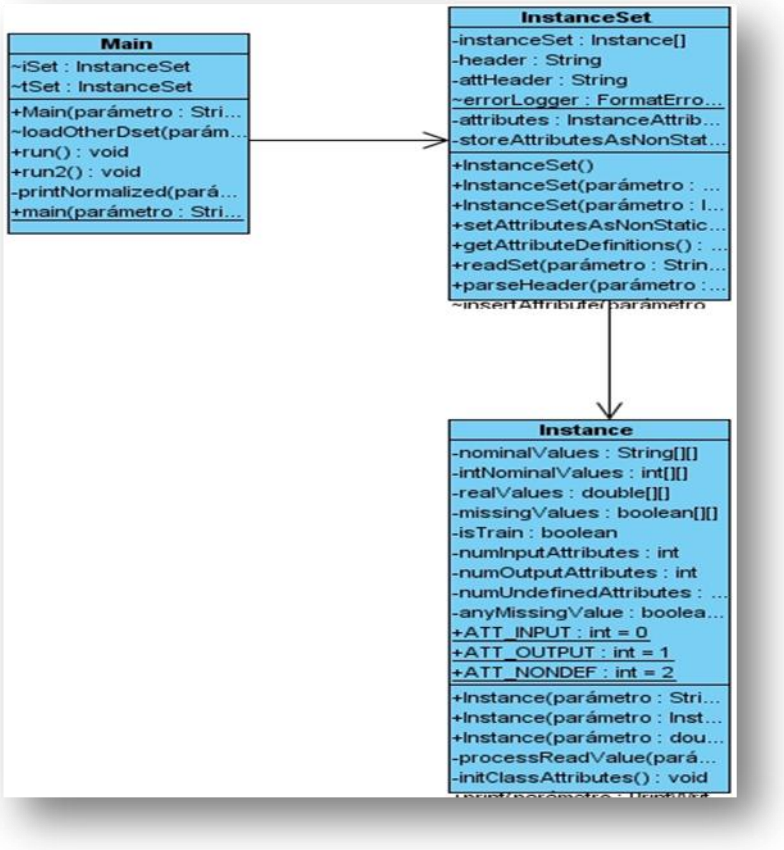


***Figura 3.6**: Patrón Bajo Acoplamiento*

En la Figura 3.6 se evidencia como la clase Instance puede presentar cualquier error sin que la clase Main sea afectada, ejemplificándose así el patrón Bajo Acoplamiento, potenciando la reutilización y disminuyendo la dependencia entre las clases.

Polimorfismo

El polimorfismo significa asignar el mismo nombre a servicios en varios objetos, cuando los servicios se parecen o están relacionados entre sí (27).

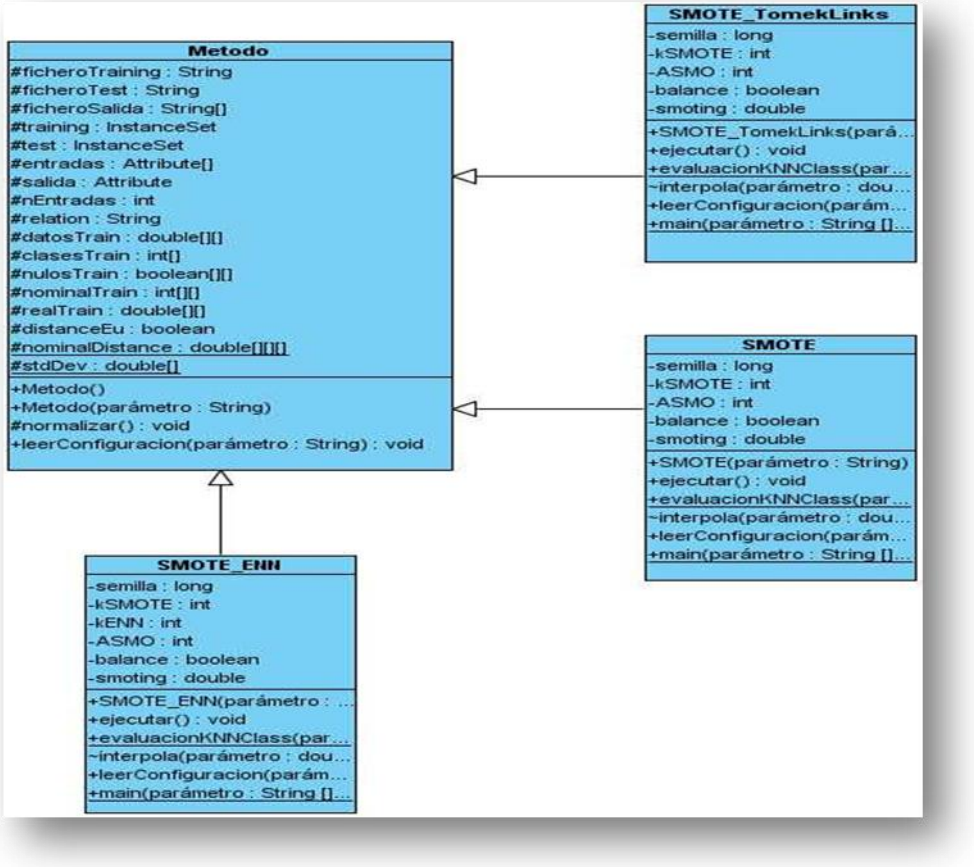


***Figura 3.7**: Patrón Polimorfismo*

La Figura 3.7 muestra como heredan las clases SMOTE, SMOTE_TomekLinks, y SMOTE_ENN de la clase Método evidenciando el polimorfismo utilizado en el código de la herramienta.

3.3 Diagrama de Clases del Diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases y las relaciones entre ellas. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se

manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

También estos diagramas son los más comunes en el modelado de sistemas orientados a objetos y no solo para la visualización, especificación y documentación del modelo estructural, sino también para la construcción de sistemas ejecutables y para la realización de Ingeniería hacia adelante e ingeniería inversa. (3, p. 40)

La Figura 3.8 muestra el diagrama de clases del diseño elaborado para la herramienta, se consideró ocultar los atributos y métodos correspondientes a las clases del diagrama, pues son muchos por cada clase y esto afectaría la comprensión del diagrama, al hacerse este muy grande.

El diagrama de clases representa los paquetes en los que está dividido el sistema, un subsistema Balance que está formado por las clases BalanceView y ConvertARFF que son las interfaces de la aplicación. La clase BalanceView hace sus llamadas al método ejecutar() de las clases SMOTE, SMOTE_TomekLinks y SMOTE_ENN, que heredan de la clase Método que se encuentra en el subsistema Basic junto a la clase KNN, relacionada con las clases del subsistema Instance_Selection además de las clases Attribute y Attributes, estas presentes en el subsistema Database, relacionadas por una composición de uno a muchos. En este subsistema también se encuentra la clase InstanceSet que evidenciando el patrón Alta Cohesión delega en sus clases vecinas las responsabilidades pertinentes a cada una de ellas. Además se representa también el subsistema Core que contiene la clase Fichero utilizada por las clases SMOTE, SMOTE_TomekLinks y SMOTE_ENN.

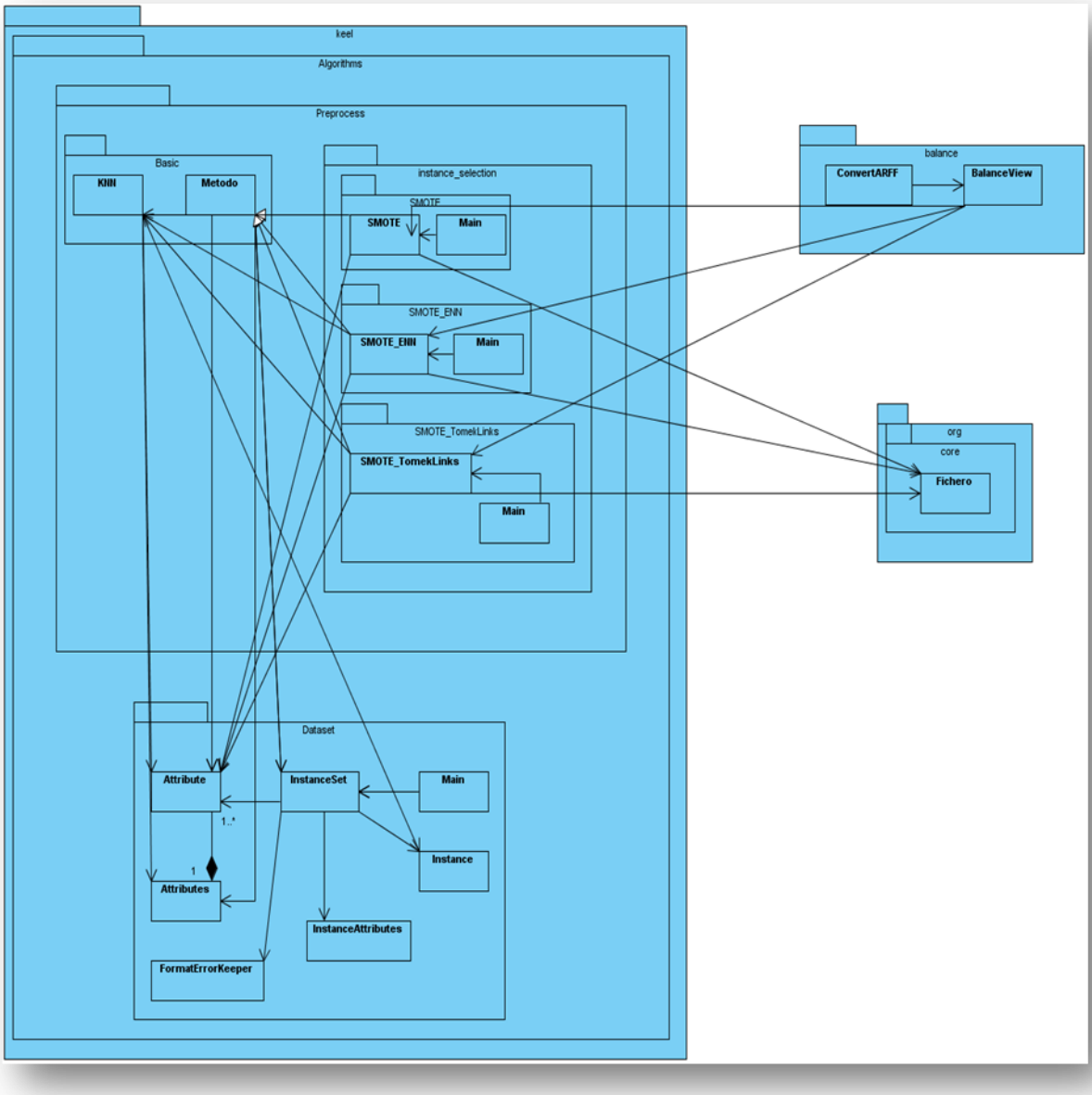


Figura 3.8: *Diagrama de clases del diseño*

3.4 Funcionamiento del plug-in

Los plug-ins le brindan a una aplicación la capacidad de agregar funcionalidades nuevas en tiempo de ejecución. Como resultado de la investigación se logró implementar un plug-in, realizado con el objetivo de reestructurar las muestras desbalanceadas de la plataforma alasGRATO.

El plug-in cuenta con una interfaz visual que permite cargar las muestras y proponer un directorio para tomar los resultados del balanceo (Ver anexo 1), también permite variar los parámetros de la operación, pues muestra los parámetros por defecto que propone cada algoritmo, el cual puede elegirse en la parte superior de la aplicación. (Ver anexo 2)

Los parámetros que se muestran son:

- Semilla
- Número de vecinos
- Número de vecinos ENN (Habilitado solo para SMOTE + ENN)
- Sobre-muestreo con vecinos de la clase
- Equilibrio exacto de las clases
- Cantidad de balanceos
- Función distancia

Además existe un botón Balancear, el cual ejecuta la acción del balanceo de la muestra elegida, mostrando un mensaje de terminada la operación y la dirección donde se encuentra el archivo resultante. (Ver Anexo 3)

El sistema cuenta también con una funcionalidad adicional, que permite convertir los ficheros .arff a formato .dat, dividiéndolos en dos subconjuntos, un 90% de la muestra para entrenamiento y un 10% para pruebas. Se accede a esta a través del menú Archivo/Importar archivo ARFF. (Ver Anexo 4)

3.5 Procedimiento general para balancear la muestra

Los datos de actividad biológica reportados para los ensayos fragmentos y mespectrales, presentan un elevado grado de desbalance. Teniendo en cuenta esto se realizaron los siguientes pasos:

1. Se aplicaron pruebas de clasificación con MSV a las muestras, antes de balancearse.
2. Se balancearon las muestras con cada uno de los algoritmos de la herramienta.
3. Se les aplicó nuevamente pruebas de clasificación con MSV.

3.5.1 Pruebas de clasificación con MSV antes del balanceo

Para la clasificación de las muestras se emplean las máquinas de soporte vectorial del tipo C-SVC perteneciente a la librería libSVM en su versión 2.8. Los parámetros usados por este clasificador toman valores por defecto, estos son: *cost* que recibe 1.0, *nu* que recibe 0.5, *gamma* que recibe 0.0, *kernelType* que recibe *radial basis function* y *prababilityEstimates* que recibe *false*. El método para determinar las muestras de entrenamiento y prueba que se empleó es la validación cruzada (*cross validation*). En la tabla 3.1 se muestran los resultados.

Muestra	**% de clasificación**
Fragmentos	58
Mespectrales	74

Tabla 3.1*: Resultados de la clasificación con C-SVC y cross validation 10*

Los resultados obtenidos fueron de 58% de clasificación para fragmentos y 74% para meesprectales, los cuales según criterios internacionales no proporcionan los mejores modelos de clasificación. Esto se debe a que las muestras analizadas son altamente desbalanceadas ya que IR > 10.

3.5.2 Balanceamiento de las muestras

Conocido el por ciento de clasificación correcto de cada muestra, se balancearon las mismas con cada uno de los algoritmos presentes en la herramienta, para algunas combinaciones de sus parámetros. Los tres algoritmos presentan parámetros iguales, estos son: la semilla, que puede ser cualquier número aleatorio y este no influye en el resultado; sobremuestreo con vecinos de la clase, es un parámetro que se le pasa por defecto ya que en todos los algoritmos está presente el SMOTE y este como ya se conoce sobremuestrea mediante la técnica del vecino más cercano; cantidad de balanceos, otro parámetro que también se le pasa por defecto, al igual que la función distancia. El número de vecinos es otro parámetro que presentan estos algoritmos, pero este puede variar en dependencia a la cantidad de sobremuestreo que se requiera hacer y toma un valor máximo de 5. En el caso del algoritmo SMOTE + ENN recibe un parámetro adicional, que es el número de vecinos del algoritmo ENN, el cual se recomienda que sea tres.

3.5.3 Pruebas de clasificación con MSV después de balanceada la muestra

Las muestras se clasificaron nuevamente después de ser balanceadas. Para todos los algoritmos se realizaron pruebas usando los parámetros que reciben por defecto y en el caso del número de vecinos (k) variándolo con k = 1, k = 3 y k =5.

En la tabla 3.2 se muestran los resultados obtenidos con el algoritmo SMOTE.

Muestra	k	% de clasificación
Fragmentos	1	60
Fragmentos	3	60
Fragmentos	5	61
Mespectrales	1	75
Mespectrales	3	75
Mespectrales	5	77

***Tabla 3.2**: Resultados de la clasificación con C-SVC y cross validation 10, para el algoritmo SMOTE*

Para la muestra de fragmentos el algoritmo SMOTE en todos los casos brinda un resultado superior al 2%, con respecto a la muestra original de la tabla 3.1; mientras que para la muestra de mespectrales brinda un resultado superior al 1%. Siendo el número de vecinos k = 5 donde el algoritmo brinda sus mejores resultados, para ambas muestras.

En la tabla 3.3 se muestran los resultados obtenidos con el algoritmo SMOTE + *Tomek – Links.*

Muestra	k	% de clasificación
Fragmentos	1	62
Fragmentos	3	62
Fragmentos	5	63
Mespectrales	1	76
Mespectrales	3	76
Mespectrales	5	77

***Tabla 3.3**: Resultados de la clasificación con C-SVC y cross validation 10, para el algoritmo SMOTE + Tomek - Links.*

Para la muestra de fragmentos el algoritmo SMOTE + *Tomek - Links* en todos los casos brinda un resultado superior al 4%, con respecto a la muestra original de la tabla 3.1; mientras que para la muestra de mespectrales brinda un resultado superior al 2%. Siendo el número de vecinos k = 5 donde el algoritmo brinda sus mejores resultados, para ambas muestras.

En la tabla 3.4 se muestran los resultados obtenidos con el algoritmo SMOTE + ENN.

Muestra	**k**	**% de clasificación**
Fragmentos	1	62
Fragmentos	3	62
Fragmentos	5	64
Mespectrales	1	76
Mespectrales	3	76
Mespectrales	5	77

***Tabla 3.4**: Resultados de la clasificación con C-SVC y cross validation 10, para el algoritmo SMOTE + ENN.*

Para la muestra de fragmentos el algoritmo SMOTE + ENN en todos los casos brinda un resultado superior al 4%, con respecto a la muestra original de la tabla 3.1; mientras que para la muestra de mespectrales brinda un resultado superior al 2%. Siendo el número de vecinos k = 5 donde el algoritmo brinda sus mejores resultados, para ambas muestras.

Como se puede observar para los tres algoritmos los resultados de clasificación después de balancear las muestras se encuentran en un rango de 60% a 64% y de 75% a 77%, para los ensayos fragmentos y mespectrales respectivamente, por lo que se considera un resultado aceptable para los modelos de clasificación de MSV. Se pudo notar también que estos algoritmos utilizando el número de vecinos k = 5

ofrecen los mejores por cientos de clasificación. Además se puede ver que los dos algoritmos híbridos son más efectivos que el algoritmo SMOTE aplicado de forma individual. No obstante se decidió aplicar también pruebas no paramétricas para tener una comparación estadística de estos algoritmos.

3.6 Pruebas no Paramétricas

Para poder establecer una diferencia clara entre los algoritmos, en caso de que exista, se decidió utilizar el test de Wilcoxon, el cual se aplica cuando los algoritmos que se comparan están relacionados. Este consiste en realizar una prueba de comparación por parejas, para detectar diferencias más precisas mediante comparaciones de algoritmos dos a dos entre SMOTE, SMOTE + Tomek – Links y SMOTE + ENN (17, p 42). Para este test se tomaron los resultados con el número de vecinos k = 5, que es la cantidad de vecinos con la que estos algoritmos ofrecen los mejores resultados. En la tabla 3.5 se muestran los resultados obtenidos.

Comparación	p-value
SMOTE vs SMOTE + Tomek – Links	0.180
SMOTE vs SMOTE + ENN	0.180
SMOTE + ENN vs SMOTE + Tomek - Links	1.000

Tabla 3.5: *Resultados de la evaluación de las predicciones con el test de Wilcoxon.*

Se considera que si p-value es menor que 0.05 existen diferencias significativas. Como se puede observar, los tres algoritmos presentan resultados equivalentes desde el punto de vista estadístico, por lo que pueden emplearse indistintamente.

3.7 Conclusiones

En este capítulo se clasificaron las muestras antes y después de ser balanceadas, llegando a la conclusión que para ambos ensayos los algoritmos brindan mejores resultados cuando usan el número de vecinos k = 5. Aunque en las pruebas no paramétricas los tres algoritmos presentan resultados equivalentes desde el punto de vista estadístico, se pudo observar que los dos algoritmos híbridos (SMOTE + *Tomek* – *Links* y SMOTE + ENN) brindan mejores por cientos de clasificación que el algoritmo SMOTE.

CONCLUSIONES

Para el presente trabajo de diploma se realizó una investigación exhaustiva desde noviembre del 2009 hasta mayo del 2010 arribando a las siguientes conclusiones:

- Se realizó una búsqueda en la bibliografía, referente al tema de las muestras desbalanceadas, identificándose los algoritmos utilizados a nivel mundial para la resolución de este tipo de problema.
- Al término del análisis a fondo de los algoritmos identificados se seleccionaron los algoritmos de rebalanceo SMOTE, SMOTE + *Tomek Links* y SMOTE + ENN; para su siguiente implementación. Resultando un sistema informático capaz de balancear muestras desbalanceadas, incorporado como *plug-in* en la versión 1.0 de la plataforma alasGRATO.
- Se evaluaron los algoritmos implementados, con los cuales se obtuvieron resultados aceptables, siendo los algoritmos híbridos los más efectivos. Con resultados entre 62% y 63% para la muestra fragmentos y para la muestra mespectrales entre 76% y 77%, con el algoritmo SMOTE + *Tomek – Links*; mientras que con el algoritmo SMOTE + ENN para la muestra fragmentos los resultados obtenidos estuvieron entre el 62% y el 64% y para la muestra mespectrales entre 76% y 77%.

RECOMENDACIONES

Como colofón a esta investigación, recomendamos continuar trabajando en:

- Incorporar a la herramienta la técnica de validación cruzada.
- Buscar nuevos métodos y medidas de evaluación que contribuyan al mejoramiento de los resultados obtenidos con los algoritmos implementados.

REFERENCIAS BIBLIOGRÁFICAS

1. **Calzadilla, G. V.** Los medicamentos en el mundo asimétrico actual: del no acceso al acceso. Parte II. [En línea] 2007 (1)[Citado el: 15 de Noviembre de 2009.] http://bvs.sld.cu/revistas/mil/vol36_2_07/mil07207.html.

2. **José Hernández Orallo, M.José Ramírez Quintana, Cèsar Ferri Ramírez.** Introducción a la Minería de Datos. [En línea] 2004. [Citado el: 15 de Noviembre de 2009.] http://users.dsic.upv.es/~flip/LibroMD/.

3. **Rosa, Tonisé de la.** *Propuesta de Algoritmos para la Reducción del Espacio Muestral.* Ciudad de la Habana : s.n., 2008.

4. **Garcia, Alvaro Barreiro.** El algoritmo de aprendizaje. [En línea] 1997. [Citado el: 16 de Noviembre de 2009.] http://www.dc.fi.udc.es/ai/~barreiro/cogdocen/cctema11/node7.html.

5. **Ruiz, Vicenç Soler.** *Lógica Difusa aplicada a Conjuntos Imbalanceados: Aplicación a la detección del Síndrome de Down.* Barcelona : s.n., 2007.

6. **Gustavo E. A. P. A. Batista, Ronaldo C. Prati, Maria Carolina Monard.** *A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data.* SP, Brazil : s.n., 2002.

7. **Machado, Emerson López.** *Um Estudo de Limpeza em Base de Dados Desbalanceada e com Sobreposição de Classes.* Brasília : s.n., 2007.

8. **González, Eduardo F. Morales y Jesús A.** *El Problema de las Clases Desbalanceadas.* 2004.

9. **Díaz, Yaikiel Hernández.** *Desarrollo de Modelos de Clasificación de Actividad Biológica empleando Máquinas de Soporte Vectorial.* Ciudad de la Habana, Cuba : s.n., 2010.

10. **Japkowicz, N.** *The Class Imbalance Problem: Significance and Strategies. .* Las Vegas : s.n., 2000.

11. **Provost, F.** *Learning with Imbalanced Data Sets 101.* Florida : s.n., 2000.

12. **Chawla N.V., K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer.** *SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research.* 2002.

13. **Barandela R., R.M. Valdovinos, J.S. Sánchez.** *New applications of ensembles of classifiers. Pattern Analysis and Applications.* 2003.

14. **L.O, Hall.** *Data mining from extreme data.* Boston, USA : s.n., 2001.

15. **Batista G.E.A.P.A., R.C. Prati, M.C. Monard.** *A study of the behavior of several methods for balancing machine learning training data.* 2004.

16. **Jesús Alcalá-Fdez, María José del Jesús, Josep M. Garrell, Francisco Herrera, Cesar Hervás, Luciano Sánchez.** *Proyecto KEEL: Desarrollo de una Herramienta para el Análisis e Implementación de Algoritmos de Extracción de Conocimiento Evolutivos.* Granada, España : s.n., 2004.

17. **Entenza, Julio Omar Prieto.** *Desarrollo de modelos de softcomputing para la predicción de actividad biológica.* Ciudad Habana, Cuba : s.n., 2010.

18. **Yadira Marrero López, Adonis Ricardo Rosales García.** *Propuesta del diseño arquitectónico de la Plataforma bioGRATO.* Ciudad de la Habana, Cuba : s.n., 2008.

19. **Martínez, Julio Antonio Villaverde.** *Un nuevo Front-End para la plataforma alasGRATO.* Ciudad de la Habana, Cuba : s.n., 2009.

20. **Garcerant, Iván.** Tecnología y Synergix. [En línea] 10 de Julio de 2008. [Citado el: 16 de Abril de 2010.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

21. **De Miguel, A, Martínez, P.** Diseño de Bases de Datos. [En Línea] 2008. [Citado el: 20 de Abril de 2010.] http://basesdatos.uc3m.es/fileadmin/Docencia/DBD/Curso0607/Teoria/MODELO_ER.pdf.

22. **Gracia, Joaquin.** Ingeniero Software. [En línea] 27 de Mayo de 2005. [Citado el: 20 de Abril de 2010.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

23. *Patrones GRASP.* **Diana Paola Hurtado Bustamante, Diana Patricia Gutiérrez Valencia, Juan Pablo Suárez Valencia, Gabriel Asakawa, Eudo Quevedo Pantoja.** 2009.

24. **Erick Salazar, Anaís Aponte.** *Patrones de Diseño.* 1999.

25. **Mora, Roberto Canales.** Adictos al trabajo. [En línea] 22 de Diciembre de 2003. [Citado el: 2 de Mayo de 2010.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=grasp.

26. **Marcello Visconti, Hernán Astudillo.** Fundamentos de Ingeniería de Software. http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

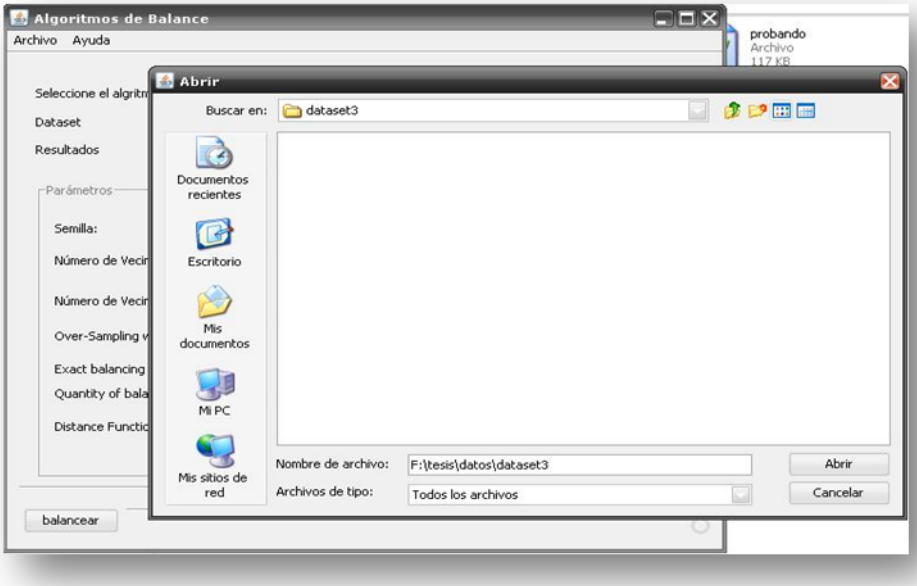
27. **Murano M., Amore S., Puelles M., Bartoloni V., Borinsky M., Martínez M.** Defensa técnica de Patentes de Invención (II). *Polimorfismo*.

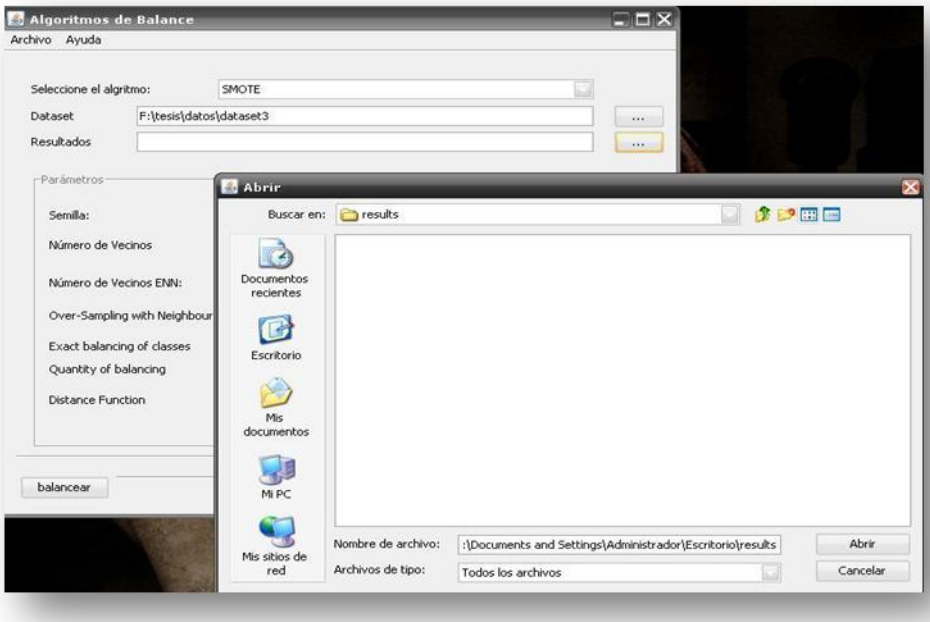
BIBLIOGRAFÍA

- Acuña, E. y Daza, L. (2004) (activo Enero 2010): Métodos para Mejorar la Calidad de un Conjunto de Datos en Descubrimiento de Conocimiento. http://sci2s.ugr.es/docencia/asignatura.php?id_asignatura=11
- Batista, GE. Ronaldo, PC. Monard, MC. (2004) (activo Enero 2010): A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. http://sci2s.ugr.es/docencia/doctoM6/batista.pdf
- Chawla, N. (2002): SMOTE: Synthetic Minority Over-sampling TEchnique.
- De La Rosa Martín, T. y Herrera Martínez HL. (2008): Propuesta de Algoritmos para la Reducción del Espacio Muestral.
- García, S. y Herrera, F. (2008) (activo Febrero 2010): Evolutionary Under-Sampling for Classification with Imbalanced Data Sets: Proposals and Taxonomy. http://sci2s.ugr.es/docencia/doctoM6/2008-Garcia-EC.pdf
- Hernández Díaz, Y. (2010): Desarrollo de Modelos de Clasificación de Actividad Biológica empleando Máquinas de Soporte Vectorial.
- Herrera, F. (2006) (activo Enero 2010): Nuevas Tendencias en Minería de Datos. http://sci2s.ugr.es/docencia/doctoM6/Curso-M6-Clasificacion%20con%20datos%20no%20balan.pdf
- Herrera, F. y Otero, J. (2000): Un estudio empírico preliminar sobre los testsestadísticos más habituales en el aprendizaje automático.
- Morales, E. F. y González JA. (2007): El Problema de las Clases Desbalanceadas.
- Prieto Entenza, JO. (2010): Desarrollo de modelos de softcomputing para la predicción de actividad biológica.
- Soler Ruiz, V. (2007): Lógica Difusa aplicada a Conjuntos Imbalanceados: Aplicación a la Detección del Síndrome de Down.

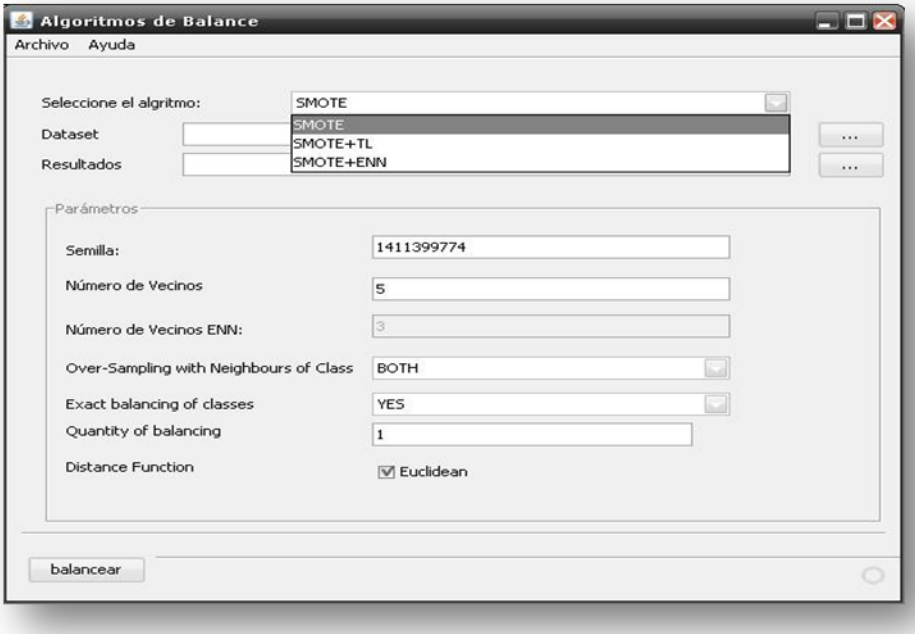
ANEXOS

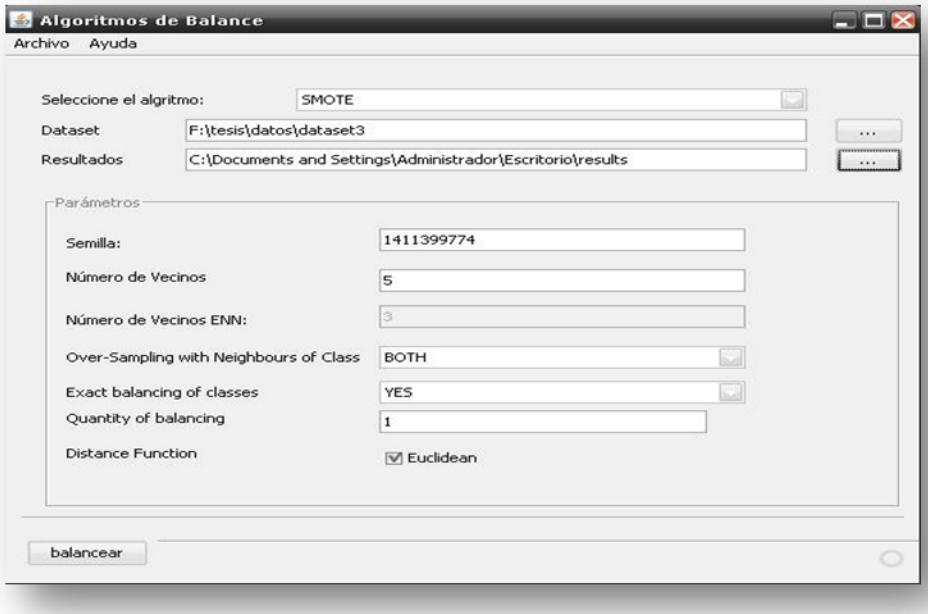
Anexo1: Funcionamiento del sistema.



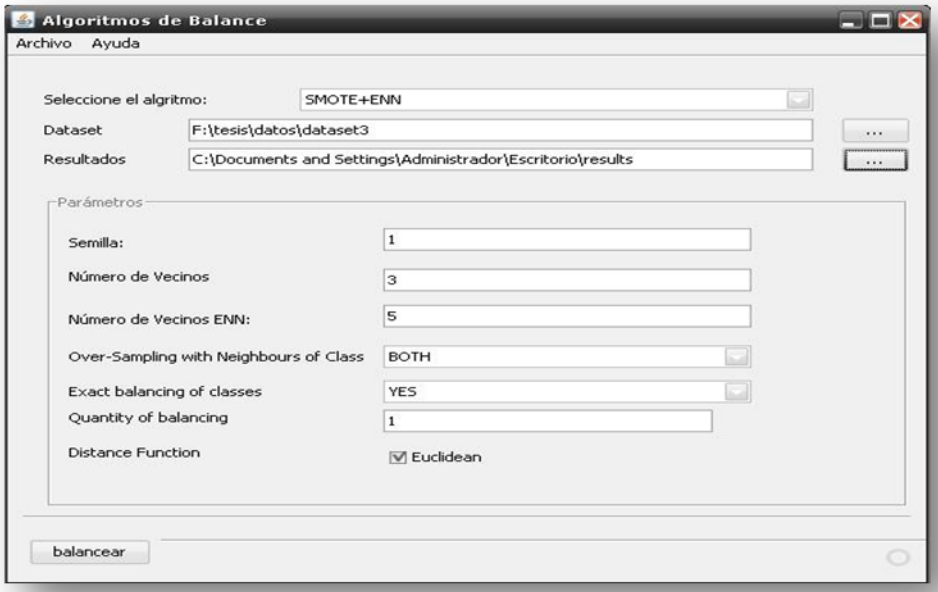


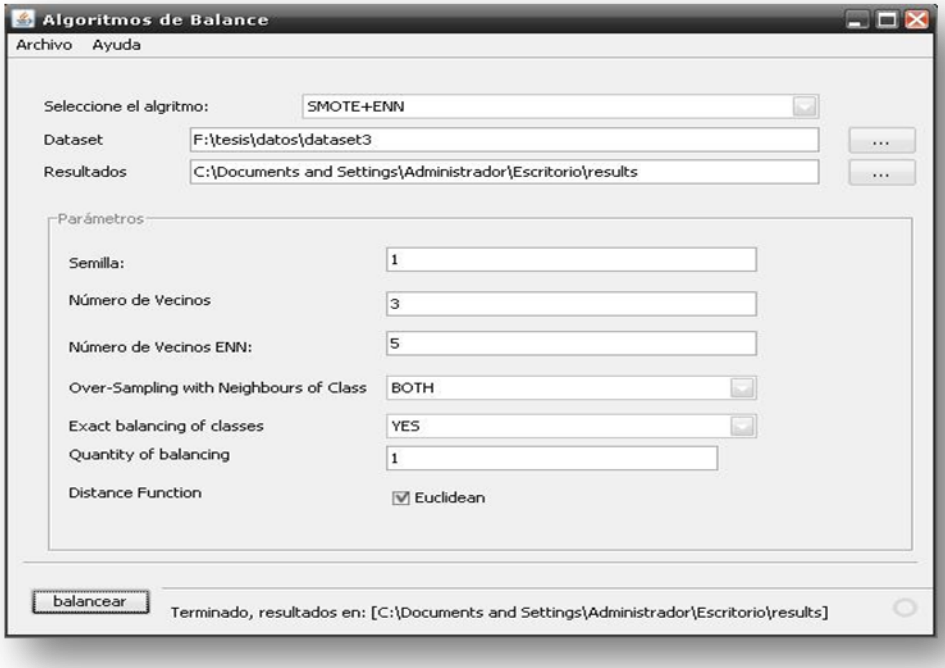
Anexo 2: Funcionamiento del sistema.



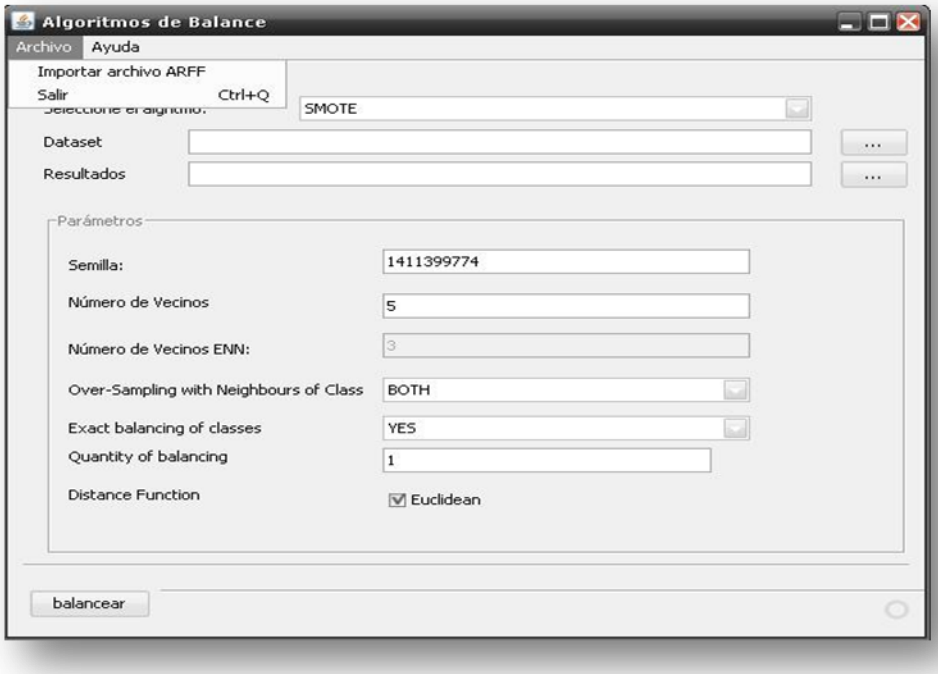


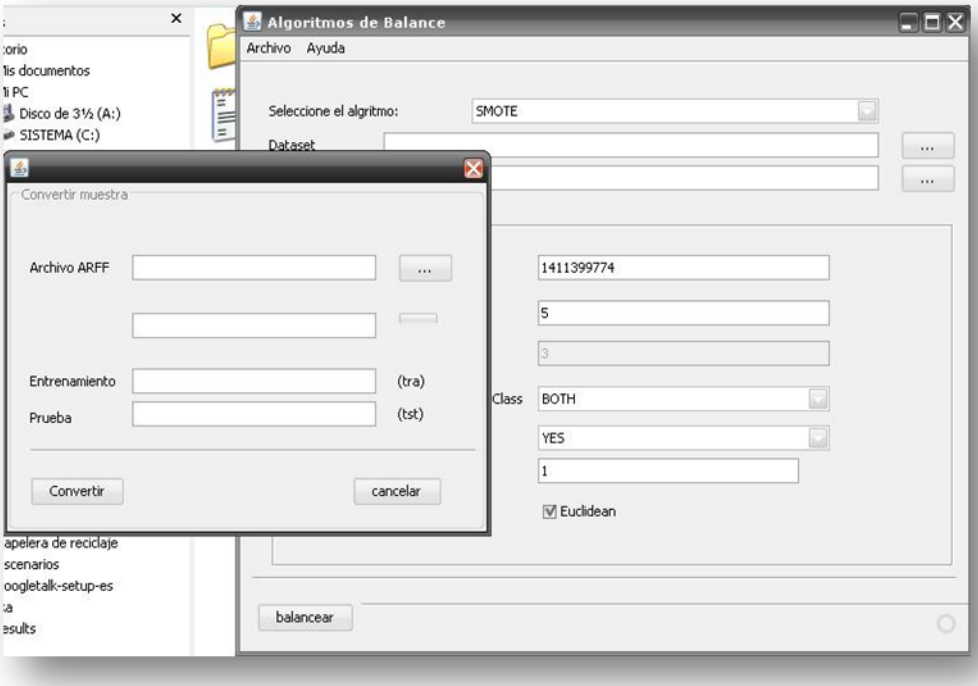
Anexo 3: Funcionamiento del sistema.





Anexo 4: Funcionamiento del sistema.





GLOSARIO DE TÉRMINOS

A

Actividad biológica: Actividad que caracteriza el comportamiento biológico en compuestos químicos (Molécula o Fragmento).

Atributo: Descripción de alguna medida existente en el universo de discurso que toma valores en un determinado dominio.

B

Bioinformática: Es la aplicación de los ordenadores y los métodos informáticos en el análisis de datos experimentales y simulación de los sistemas biológicos.

C

Compuestos Orgánicos: Compuestos cuya composición fundamental es sobre la base del elemento químico carbono.

D

Descriptor: Número que caracteriza estructuralmente la molécula.

F

Farmacología: ciencia que estudia el origen, las acciones y las propiedades que las sustancias químicas ejercen sobre los organismos vivos.

G

GPL: Acrónimo de General Public Licence (Licencia pública general de GNU).

P

Plug-in: Aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.

Polimorfismo: Capacidad que tienen los objetos de una clase de responder al mismo mensaje o evento en función de los parámetros utilizados durante su invocación. Un objeto polimórfico es una entidad que puede contener valores de diferentes tipos durante la ejecución del programa.

Predicción: Valor estimado que caracteriza una propiedad o fenómeno obtenido de un modelo.