Universidad Central "Marta Abreu" de Las Villas

Facultad de Matemática, Física y Computación

Licenciatura en Ciencia de la Computación

TRABAJO DE DIPLOMA

Métodos avanzados de preprocesamiento de datos para mitigar el problema de clases desbalanceadas en la clasificación multinstancia.

Autor: Jorge Luis Borges Jiménez

Tutor: Dr. Dánel Sánchez Tarragó

Santa Clara

2015

Agradecimientos

A mi familia por haberme apoyado durante estos cinco años, a todos les estoy agradecidos porque de una forma u otra han ayudado en mi formación como profesional, en especial a mis padres porque sé el esfuerzo y los sacrificios que han hecho por mí. A mi tía Norbelia porque si no fuera por su apoyo, todo hubiera sido muy difícil. Haré lo imposible con tal de que se sientan orgullosos de mi.

A todos los amigos que han estado a mi lado desde el comienzo, así como los que no pudieron terminar y a los que siempre he tenido presente.

A todos los profesores que han contribuido a mi preparación profesional.

Resumen

El problema de clasificación es uno de los tipos de problema más importante que busca resolver el aprendizaje automático. Específicamente la clasificación multinstancia ha provocado gran interés debido fundamentalmente a su capacidad para modelar problemas con datos ambiguos y con la relaciones uno a muchos, tanto así que múltiples estudios empíricos han mostrado la superioridad del enfoque multinstancia sobre el tradicional.

Uno de los problemas que afecta a la clasificación multinstancia es el problema del desbalance de clases que ocurre cuando hay mucha diferencia en el tamaño de las clases, provocando un modelo de aprendizaje erróneo a partir de los datos. Este problema ha recibido poca atención hasta el momento, y solo aparecen escasas discusiones sobre él y aisladas soluciones en la literatura. En la clasificación tradicional, uno de los métodos más usados para solucionar este problema consiste en modificar la distribución de los datos de entrenamiento a través de técnicas de muestreo para equilibrar el balance de las clases.

El submuestreo de la clase mayoritaria y sobremuestreo de la clase minoritaria son técnicas de muestreo que equilibran el tamaño de las clases extrayendo y agregando datos respectivamente en conjuntos desbalanceados. La combinación de estas técnicas han mejorado la calidad de la clasificación en la clasificación tradicional.

En esta tesis se implementa un método de preprocesamiento de datos basados en técnicas de muestreo que combina sobremuestreo de la clase minoritaria y submuestreo de la clase mayoritaria en la clasificación multinstancia. Las pruebas experimentales validadas por pruebas estadísticas demostraron que existen mejoras en la calidad de la clasificación en determinadas variantes del método.

En la tesis también se modifica un algoritmo de preprocesamiento de datos multinstancia llamado MISMOTE que sobremuestrea la clase minoritaria produciendo ejemplos sintéticos. Se diseñaron varias alternativas al algoritmo original que lograron mejoras en la clasificación en conjuntos de datos que cumplen con determinados requisitos.

Palabras claves: clasificación, multinstancia, desbalance, sobremuestreo, submuestreo, MISMOTE.

Abstract

The problem of classification is one of the most important problem types that seeks to solve the machine learning. Specifically, the multi-instance classification has caused great interest mainly because of its ability to model problems with ambiguous data and with one to many relationships, so much so that many empirical studies have shown the superiority of multi-instance over the traditional approach.

One problem affecting the multi-instance learning is the class imbalanced problem that occurs when exist much difference in the size of classes, causing erroneous learning model from the data. This problem has received little attention so far, and only few discussions about it and isolated solutions in the literature appear. In the traditional classification, one of the most used methods to solve this problem is to change the distribution of the training data through sampling techniques to equalize the balance of classes.

The sub-sampling the majority class and oversampling of the minority class are sampling techniques that balance class sizes by removing and adding respectively unbalanced data sets. The combination of these techniques have improved the quality of the classification in the traditional classification.

This thesis presents a preprocessing method based on sampling techniques combining the minority class oversampling and undersampling of the majority class in the multi-instance data classification. Experimental evidence validated by statistical tests showed that there are improvements in the quality of the classification in certain variants of the method.

In the thesis was modified an algorithm of multi-instance data preprocessing called MISMOTE that oversampled producing synthetic minority class examples. Several alternatives to the original algorithm were designed that shown improvements in classification process in some datasets that meet certain criteria.

Índice general

Introducción **1**
Situación problémica . . . 2
Hipótesis . . . 3
Preguntas de investigación . . . 3
Objetivo general . . . 3
Objetivos específicos . . . 4
Estructura . . . 4

1. Fundamentación teórica **5**
1.1. Clasificación Multinstancia . . . 5
1.1.1. Aprendizaje Automático . . . 5
1.1.2. Aprendizaje Supervisado . . . 6
1.1.3. Aprendizaje multinstancia . . . 7
1.1.4. Clasificación multinstancia . . . 7
1.1.5. Otros modelos donde está presente el aprendizaje multinstancia . . . 8
1.2. Importancia de la clasificación multinstancia . . . 9
1.2.1. Surgimiento del aprendizaje multinstancia . . . 9
1.2.2. Ejemplos donde se puede aplicar la clasificación multinstancia . . . 11
1.2.3. La ambigüedad como fuente de problemas multinstancia . . . 12
1.3. Problema del desbalance de clases . . . 12
1.3.1. ¿Cómo medir el desempeño de la clasificación cuando las clases están desbalanceadas? . . . 13
1.3.2. Soluciones existentes en la clasificación tradicional . . . 15
1.3.3. Técnicas de muestreo . . . 16
1.3.4. Soluciones existentes en la clasificación multinstancia . . . 18

2. Combinación submuestreo con sobremuestreo **20**
2.1. Submuestreo de la clase mayoritaria . . . 20
2.2. Sobremuestreo de la clase minoritaria . . . 21
2.3. Remuestreo en ambas clases . . . 23
2.4. Experimentos . . . 24
2.4.1. Conjuntos de datos . . . 26
2.4.2. Algoritmos de clasificación . . . 27
2.4.3. Validación de la experimentación . . . 27
2.5. Resultados . . . 28

2.5.1. Pruebas estadísticas generales . . . 30
2.5.2. Pruebas estadísticas por dominio de aplicación . . . 32
2.6. Conclusiones parciales . . . 36

3. Modificación del Algoritmo MISMOTE **38**
3.1. ¿Cómo mejorar MISMOTE? . . . 38
3.2. Probabilidad de una instancia de ser positiva . . . 39
3.2.1. Densidad Diversa . . . 39
3.2.2. Selección de una instancia teniendo en cuenta su probabilidad de ser positiva . . . 41
3.3. Nueva versión de MISMOTE . . . 42
3.4. Experimentos . . . 43
3.4.1. Conjuntos de Datos . . . 44
3.4.2. Algoritmos de clasificación . . . 45
3.4.3. Método de validación y medidas de desempeño . . . 46
3.5. Resultados con datos artificiales . . . 46
3.6. Resultados con datos reales . . . 52
3.7. Conclusiones parciales . . . 53

Bibliografía **55**

Índice de figuras

1.1.1.Composición del (a) aprendizaje supervisado tradicional y (b) aprendizaje supervisado multinstancia. 7
1.2.1.Diferentes conformaciones de una molécula de butano (C4H10) 10
1.2.2.Ejemplo de representación multinstancia en las variaciones en el trazado de los dígitos . 11
1.3.1.Ejemplo de curva *ROC* y el cálculo del *AUC* 15
1.3.2.Ejemplo de interpolación aleatoria hecha por SMOTE 17

2.4.1.Diseño del experimento para determinar cuál es la mejor combinación de submuestreo y sobremuestreo 25
2.5.1.Gráfica con los resultados de la tabla 2.5.1 30
2.5.2.Prueba de Friedman para los resultados generales 31
2.5.3.Prueba de Friedman para los resultados con los conjuntos de datos WIR 33
2.5.4.Prueba de Friedman para los resultados con los conjuntos de datos Corel 35

3.4.1.Diseño del experimento para determinar cuál es la mejor variante de MISMOTE 44
3.5.1.Prueba de Friedman para los resultados con los conjuntos de datos con IR=3 47
3.5.2.Prueba de Friedman para los resultados con los conjuntos de datos con IR=9 48
3.5.3.Prueba de Friedman para los resultados con los conjuntos de datos con IR=19 50
3.5.4.Prueba de Friedman para los resultados con los conjuntos de datos con IR=99 51
3.6.1.Prueba de Friedman para los resultados con los conjuntos de datos reales 53

Índice de cuadros

1.1.1.Representación de la información en el aprendizaje supervisado tradicional. 6
1.1.2.Representación de los datos en el aprendizaje multinstancia 8
1.3.1.Matriz de confusión para un problema de dos clases 13
1.3.2.Variantes de AdaBoost sensible al costo usadas para clasificación multinstancia en problemas con clases desbalanceadas. 18

2.4.1.Conjuntos de datos utilizados para las pruebas con el método de muestreo combinado . 26
2.4.2.Clasificadores multinstancia utilizados en los experimentos con el método de muestreo combinado 27
2.5.1.Resultados generales de las pruebas del algoritmo de muestreo combinado en la medida de desempeño AUC 29
2.5.2.Post Hoc con la prueba de Shaffer para encontrar diferencias entre pares de combinaciones de muestreos en los resultados con todos los conjuntos de datos 32
2.5.3.Resultados para los conjuntos de datos WIR de las pruebas del algoritmo de muestreo combinado 33
2.5.4.Post Hoc con la prueba de Shaffer para encontrar diferencias entre pares de combinaciones de muestreos en los resultados con los conjuntos de datos WIR 34
2.5.5.Resultados para los conjuntos de datos Corel de las pruebas del algoritmo de muestreo combinado 35
2.5.6.Post Hoc con la prueba de Shaffer para encontrar diferencias entre pares de combinaciones de muestreos en los resultados con los conjuntos de datos Corel 36

3.4.1.Cantidad de bolsas de cada clase según el IR 45
3.4.2.Clasificadores multinstancia utilizados en las pruebas con las distintas variantes de MISMOTE 45
3.5.1.Resultados de AUC promedio para los conjuntos de datos artificiales con IR=3 46
3.5.2.Post hoc con prueba de Shaffer para encontrar diferencias entre pares de variantes de MISMOTE en conjuntos de datos con IR=3 47
3.5.3.Resultados de AUC promedio para los conjuntos de datos artificiales con IR=9 48
3.5.4.Post hoc con prueba de Shaffer para encontrar diferencias entre pares de variantes de MISMOTE en conjuntos de datos con IR=9 49
3.5.5.Resultados de AUC promedio para los conjuntos de datos artificiales con IR=19 49
3.5.6.Post hoc con prueba de Shaffer para encontrar diferencias entre pares de variantes de MISMOTE en conjuntos de datos con IR=19 50
3.5.7.Resultados de AUC promedios para los conjuntos de datos artificiales con IR=99 51

3.6.1.Resultados de AUC con el clasificador MITI en los conjuntos de datos reales para las cuatro variantes de MISMOTE. 52

List of Algorithms

1. Submuestreo aleatorio 21
2. MISMOTE: distribución de la generación 22
3. MISMOTE: Generación de la bolsa sintética 23
4. Remuestreo en ambas clases 24

5. MISMOTE: Generación de la bolsa sintética 43

Introducción

La inteligencia artificial es una disciplina que está en el centro de la revolución informática. Una de las ramas de la inteligencia artificial que ha alcanzado mayor auge es el aprendizaje automático, debido a su gran aplicabilidad: prácticamente todo conocimiento es susceptible de ser aprendido, si bien el uso del aprendizaje automático en determinadas aplicaciones puede estar limitado por razones técnicas, económicas, legales u otras, los métodos del aprendizaje automático son usados como herramientas básicas en otras ramas de la inteligencia artificial, como la minería de datos y el reconocimiento de patrones.

El problema de clasificación es uno de los tipos de problema más importante que busca resolver el aprendizaje automático, en el cual se parte de un conjunto de objetos de clasificación, cada uno descrito por un vector de valores (cada valor describiendo a un atributo) y una etiqueta de clase. Los objetos de clasificación se toman como ejemplo para crear un modelo matemático capaz de predecir las etiquetas de clase de nuevos objetos carentes de clasificación. El aprendizaje automático ha producido una variedad importante de soluciones enfocadas mayormente a problemas donde los objetos de clasificación están descritos por un único vector de atributos, identificando a este tipo de problemas de clasificación con el epíteto *tradicional*.

Los problemas de clasificación tradicional pueden encontrarse en un número importante de aplicaciones debido a que son relativamente simples de modelar. Sin embargo, otras aplicaciones que demandan formas distintas y más complejas de modelar han llevado a un nivel superior el problema de clasificación. Una de las alternativas que han surgido recientemente al modelo de clasificación tradicional es la clasificación multinstancia, en la cual cada objeto de clasificación está descrito por múltiples vectores de atributos en lugar de uno. Cada vector de atributos representa una instancia del objeto, y cada objeto se conoce comúnmente como bolsa. Las etiquetas de clase se asocian con las bolsas pero no con las instancias, sin embargo es la información contenida en las instancias la que determina en último caso la etiqueta de clase de las bolsas. El objetivo de la clasificación multinstancias es aprender un modelo matemático que represente la relación que existe entre las instancias y la etiqueta de la bolsa para predecir las etiquetas de clase de nuevas bolsas carentes de clasificación.

El aprendizaje multinstancia ha provocado gran interés debido fundamentalmente a su capacidad para modelar problemas de clasificación con datos ambiguos y problemas con la relación uno a muchos entre sus datos,característica presente en el aprendizaje relacional. Múltiples estudios empíricos han mostrado, en tipos de problemas, la superioridad del enfoque multinstancia sobre los algoritmos de clasificación tradicional, los cuales tratan a todos los datos por igual sin tomar en cuenta las relaciones que puedan existir entre ellos. Se han diseñado algunos algoritmos específicos para el aprendizaje de problemas de clasificación multinstancia y muchos de estos basados en métodos tradicionales como los árboles y reglas de decisión, las máquinas de soporte vectorial, la máxima verosimilitud y los métodos basados en instancias. Varios de estos algoritmos han sido usados con éxito en aplicaciones de campos

tan diversos como la bioinformática, el diagnóstico médico, control de finanzas o la recuperación de información y cada día aparecen nuevos problemas de clasificación que pueden beneficiarse con los algoritmos del enfoque multinstancia. A pesar de los avances obtenidos en el desarrollo y aplicación de métodos de clasificación multinstancia, ésta es un área de investigación muy joven dentro del aprendizaje automático, y tiene aún muchos aspectos sin resolver que requieren ser estudiados.

Particularmente, uno de los problemas que afecta a la clasificación multinstancia es el problema del desbalance de clases. Se dice que las clases están desbalanceadas cuando hay diferencia en el tamaño de las clases, esto provoca que el modelo de aprendizaje inducido a partir de los datos no represente adecuadamente el concepto que se pretende aprender y provoque muchos errores de clasificación[36].

El problema de clases desbalanceadas fue identificado originalmente en el escenario de la clasificación tradicional. En este escenario ha sido propuesta una gran variedad de soluciones para el problema de clases desbalanceadas. Lamentablemente, debido a las diferencias entre estos dos enfoques del problema de clasificación, los algoritmos propuestos en estas soluciones no pueden ser aplicados para enfrentar el problema de clases desbalanceadas en la clasificación multinstancia. El problema de clases desbalanceadas en la clasificación multinstancia ha recibido poca atención hasta el momento, y solo aparecen escasas discusiones sobre él y aisladas soluciones en la literatura. En el escenario de la clasificación tradicional, uno de los métodos más usados para tratar el problema de clases desbalanceadas consiste en modificar la distribución de los datos de entrenamiento a través de técnicas de muestreo para equilibrar el balance de las clases. Esta variante podria ser aplicada para dar solución a este problema en la clasificación multinstancia. En esta tesis se propone un nuevo método basado en técnicas de muestreo para enfrentar el problema de clases desbalanceadas y la modificación de un método ya realizado con el objetivo de obtener mejores resultados.

Situación problémica

El problema de desbalance de clases surge en los problemas de clasificación cuando la diferencia en el tamaño de las clases es muy acusado, provocando un deterioro importante en la calidad de la clasificación, tanto en la clasificación tradicional como en la clasificación multinstancia. Aunque el problema de desbalance de clases ha sido principalmente estudiado en la clasificación simple-instancia, la clasificación multinstancia también sufre el problema y se han hecho muy pocos aportes en esta línea con vista a solucionarlo. Un enfoque de solución es equilibrar el tamaño de ambas clases aplicando técnicas de muestreo, que fundamentalmente son el submuestreo de la clase mayoritaria y el sobremuestreo de la clase minoritaria.

Con el submuestreo de la clase mayoritaria se eliminan las bolsas que pertenecen a esta clase, dejando el conjunto de datos con n instancias menos, donde n representa la diferencia entre la cantidad de bolsas negativas y las positivas. Con el sobremuestreo de la clase minoritaria se añaden nuevas instancias positivas al conjunto de datos hasta tener la misma cantidad de bolsas que la clase mayoritaria. Para la creación de una nueva bolsa se escogen dos bolsas positivas y se interpolan sus instancias para dar lugar a las instancias que pertenecerán a la nueva bolsa.

Recientemente se desarrolló un método de preprocesamiento de datos multinstancia, llamado MISMOTE [36], que permite mejorar la calidad de clasificación de datos multinstancia con clases desbalanceadas. MISMOTE hace un sobremuestreo de la clase minoritaria mediante la introducción de ejemplos sintéticos con el objetivo de equilibrar el tamaño de las clases. Sin embargo el método de submuestreo de la clase mayoritaria en conjuntos de datos multinstancia para mejorar la calidad de la clasificación en datos desbalanceados no ha sido estudiado.

Algunos estudios sobre clasificación tradicional (simple instancia) sugieren que la combinación de ambas técnicas (sobremuestreo y submuestreo) devuelven mejores resultados que aplicar cada técnica por separado. La combinación de ambas técnicas en la clasificación multinstancia podría potencialmente beneficiar el tratamiento de los datos con sus respectivas bondades. Sin embargo no se conoce en qué proporción la combinación de ambos es mejor.

Por otro lado, el algoritmo MISMOTE genera una bolsa aleatoria a partir de la selección de instancias de bolsas vecinas que usa para generar instancias sintéticas por medio de interpolación. La selección de las instancias la hace mediante muestreo aleatorio simple con remplazo con el objetivo de conservar en las bolsas generadas las características originales de las bolsas de la clase positiva. Sin embargo, el muestreo aleatorio con distribución uniforme de MISMOTE, puede provocar en algunos casos una falta de representatividad de las instancias positivas y por tanto podrían generarse bolsas negativas en la clase minoritaria. Es deseable introducir mejoras en los métodos de preprocesamiento de datos multinstancia con clases desbalanceadas para obtener desempeños superiores en la clasificación.

Hipótesis

- Una solución híbrida que implemente sobremuestreo de la clase minoritaria y submuestreo de la mayoritaria para datos multinstancia podría mejorar los resultados de aplicar cada una de estas técnicas por separado.
- Si en lugar de muestrear con distribución uniforme, MISMOTE muestrea con distribución proporcional a la probabilidad de cada instancia de ser positiva, entonces la bolsa generada podría tener más probabilidad de pertenecer a la clase minoritaria (positiva), lo cual mejoraría la calidad de la clasificación en algunos tipos de conjuntos de datos.

Preguntas de investigación

1. ¿Mejora el submuestreo aleatorio de la clase mayoritaria la clasificación de conjuntos de datos multinstancia desbalanceados?
2. ¿Existe un balance óptimo entre sobremuestro de la clase minoritaria y submuestreo de la clase mayoritaria en la clasificación multinstancia?
3. ¿Es posible mejorar el MISMOTE usando muestreo con distribución proporcional a la probabilidad de cada instancia de ser positiva en lugar de muestreo con distribución uniforme?
4. ¿En qué medida estos algoritmos mejorarán la clasificación de los datos?

Objetivo general

Desarrollar nuevos métodos de preprocesamiento de datos multinstancia basados en técnicas de muestreo para mitigar el problema de clases desbalanceadas en la clasificación multinstancia.

Objetivos específicos

1. Implementar un método híbrido de preprocesamiento de datos multinstancia que combine MISMOTE de la clase minoritaria y submuestreo de la clase mayoritaria.
2. Determinar cuál es el balance óptimo entre MISMOTE de la clase minoritaria y submuestreo de la clase mayoritaria en la clasificación multinstancia para distintos dominios de aplicación.
3. Desarrollar una variante del método MISMOTE de forma tal que en el proceso de interpolación realice una selección dirigida de instancias según la probabilidad de que la instancia sea positiva, en lugar de una selección aleatoria.
4. Verificar la eficacia de las soluciones propuestas y comparar los métodos desarrollados entre sí y con otras soluciones al problema de desbalance de clases en la clasificación multinstancia para determinar las mejores soluciones.

Estructura

Para una mejor comprensión del presente documento, la estructura del contenido queda conformada de la siguiente manera:

- Capítulo 1. Fundamentación Teórica: Marco teórico relativo a los conceptos necesarios para el desarrollo de este trabajo.
- Capítulo 2. Análisis e implementación del método híbrido que aplica submuestreo de la clase mayoritaria y sobremuestreo de la clase minoritaria con MISMOTE : En este se realiza el análisis e implementación de este método y los resultados que el mismo devuelve al ser probado en distintos casos y en diferentes combinaciones.
- Capítulo 3. Modificación del método MISMOTE: Análisis detallado de como funciona el método MISMOTE y en que medida se puede mejorar el mismo, también se le realizarán pruebas y se le compararán los resultados con los del método sin modificar.

Capítulo 1

Fundamentación teórica

1.1. Clasificación Multinstancia

La clasificación multinstancia es una tareas enmarcada dentro del aprendizaje multinstancia. Cada conjunto de datos está formado por bolsas, que en realidad son conjunto de instancias(vectores atributo-valor), La etiqueta de cada bolsa, es de tipo nominal y representa la clase o categoría de la bolsa. El objetivo es construir un modelo a partir de las bolsas de entrenamiento que permita predecir las etiquetas de clase de nuevas bolsas. Esta es una de las tareas que ha recibido mayor atención en el aprendizaje multinstancia debido a su gran aplicabilidad en los más variados campos de la actividad humana.

Este capítulo expone los fundamentos de la clasificación multinstancia.Presentando primeramente el contexto en el que se inserta la misma y luego se formalizándola introduciendo la notación matemática que sera usada en este informe. Seguidamente, se aborda la importancia que tiene el aprendizaje multinstancia, y con este la clasificación multinstancia. Por ultimo se introduce el tema del problema del desbalance de clases, mencionando algunas de las soluciones existentes en la clasificación tradicional y multinstancia.

1.1.1. Aprendizaje Automático

El aprendizaje automático puede definirse como: *Un programa de computadora aprende de la experiencia E respecto a alguna clase de tarea T y una medida del desempeño D, si su desempeño en la tarea T, medido a través de D, aumenta con la experiencia E*[29].

Existen varios tipos de aprendizaje automático. El más ampliamente utilizado es llamado *aprendizaje supervisado*. En este tipo de aprendizaje, los algoritmos requieren el suministro de un conjunto de ejemplos, cada ejemplo descrito por un conjunto de atributos y una etiqueta asociada que corresponde a alguna propiedad importante o decisión relativa al ejemplo. La tarea del algoritmo de aprendizaje es construir un modelo que genere predicciones precisas de las etiquetas de futuros ejemplos.

El *aprendizaje no supervisado* es un tipo de aprendizaje automático donde los ejemplos no presentan etiquetas de clase. Una forma de aprendizaje no supervisado, llamado agrupamiento, intenta dividir el conjunto de datos dado en grupos de instancias relacionadas. El grado de relación se mide típicamente utilizando una medida de proximidad, tal como la distancia Euclideana entre vectores de atributos.

El *aprendizaje semisupervisado* se aplica a conjuntos de datos donde algunas instancias están etiquetadas y otras no. Por tanto, puede considerarse que es un tipo intermedio entre el aprendizaje supervisado y el aprendizaje no supervisado. El etiquetado de los datos es con frecuencia un proceso manual y costoso, mientras que los datos sin etiquetar pueden ser adquiridos automáticamente y relativamente a bajo costo en algunos dominios de problemas.

Los tipos de aprendizaje descritos anteriormente trabajan con datos organizados en forma de tabla plana, pero existen muchos problemas de aprendizaje que tienen una estructura relacional, es decir, que están formados por varias tablas que se relacionan entre sí a través de atributos claves. Este tipo de problemas de aprendizaje se conoce como *aprendizaje relacional*, y se contrapone con los problemas que involucran una tabla plana, a los cuales se les llama problemas de aprendizaje atributo-valor o *aprendizaje proposicional*.

1.1.2. Aprendizaje Supervisado

En el aprendizaje supervisado una instancia es un par (x; y), donde $x = \langle x_1, x_2, x_3, \ldots, x_N \rangle \in$ X es un vector de N atributos y y$\in$Y es la etiqueta de decisión de la instancia. Los atributos y las etiquetas de decisión son típicamente o bien elementos de los números reales (atributos numéricos) o bien conjuntos de nombres de un dominio específico (atributos nominales). El espacio N dimensional X de donde x toma valores se conoce como espacio de instancias o espacio de atributos, y Y es el conjunto de etiquetas de decisión. La Tabla 1.1.1 muestra la forma en que se representa la información en el aprendizaje supervisado. Cada ejemplo x_i es descrito por un vector de atributos y una etiqueta de decisión y_i.

Ejemplos	Atributo 1	Atributo 2	...	Atributo N	Decisión
x_1	$x_{1,1}$	$x_{1,2}$	...	$x_{1,N}$	y_1
x_2	$x_{2,1}$	$x_{2,2}$	...	$x_{2,N}$	y_2
...	...	...	...	...	...
x_M	$x_{M,1}$	$x_{M,2}$	...	$x_{M,N}$	y_M

Cuadro 1.1.1: Representación de la información en el aprendizaje supervisado tradicional.

La tarea del aprendizaje supervisado es encontrar $f(x) = y$, basado en un conjunto de instancias de entrenamiento $D = \{(x_1, y_1), (x_2, y_2), ..., (x_M, y_M)\}$. Generalmente, $f(x)$ es una función que asigna a cada elemento del dominio una etiqueta de decisión. Cuando la etiqueta es un atributo nominal, este proceso es llamado clasificación. Cuando la etiqueta es un atributo numérico, el proceso es llamado regresión. El proceso de clasificación subyacente $f(x)$ se conoce como un concepto en la terminología del aprendizaje automático. Dado un conjunto de ejemplos de entrenamiento D a partir de los cuales aprender, un algoritmo de aprendizaje supervisado devuelve un modelo de los datos $h(x)$, el cual se pretende que sea la mejor aproximación a $f(x)$. Dicho modelo se conoce como hipótesis o descripción del concepto.

1.1.3. Aprendizaje multinstancia

El aprendizaje multinstancia es una variación del aprendizaje proposicional en el que cada ejemplo está descrito, ya no por un único vector, sino por muchos vectores atributo-valor. El enfoque multinstancia se ha aplicado a distintos tipos de aprendizaje proposicional, sin embargo es mucho más popular en el aprendizaje supervisado.

En el aprendizaje supervisado estándar cada ejemplo es una instancia que consiste en un vector de atributos, junto a una etiqueta de decisión. La tarea es aprender una función que prediga la etiqueta de decisión de un ejemplo, dado el vector de atributos.Por otra parte, en el aprendizaje multinstancia, cada ejemplo consiste en una bolsa de instancias que contiene las instancias sin orden ni concierto, pudiendo estar incluso repetida la misma instancia dentro de la bolsa. Cada bolsa tiene asociada una etiqueta de decisión, pero las instancias en sí no están etiquetadas. El problema de aprendizaje es construir un modelo, basado en las bolsas de ejemplo dadas, que pueda predecir la etiqueta de decisión de futuras bolsas.

Como se puede apreciar en la Figura 1.1.1 la el aprendizaje multinstancia es una generalización del aprendizaje proposicional tradicional ya que cualquier ejemplo de este último puede representarse en el aprendizaje multinstancia como una bolsa que contiene una única instancia.

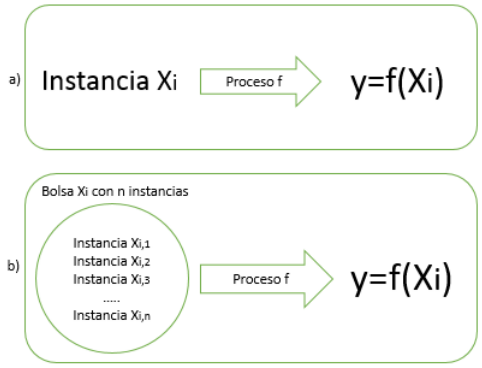

Figura 1.1.1: Composición del (a) aprendizaje supervisado tradicional y (b) aprendizaje supervisado multinstancia.

1.1.4. Clasificación multinstancia

En la clasificación multinstancia un ejemplo es un par $(X; y)$, donde$X = \{x_1, x_2, ..., x_T\} \in \mathbb{N}^x$es un multiconjunto (bolsa) de T instancias y $y \in Y$ es la etiqueta de clase del ejemplo. Se dice que es un multiconjunto y no simplemente un conjunto porque el multiconjunto, a diferencia de este, permite tener instancias repetidas. En notación matemática, $X \in \mathbb{N}^x$ significa que X es un multiconjunto (una bolsa) de elementos de x donde $\mathbb{N}$ representa el conjunto de los números naturales. Las instancias $x_i \in x, i = 1....T$ son vectores de N atributos que, a diferencia del aprendizaje supervisado estándar,

carecen de etiquetas de decisión. El conjunto x es un espacio N-dimensional formado por el producto vectorial de los N atributos que describen a las instancias, y Y es el conjunto de etiquetas de clase.

La mayoría de los trabajos desarrollados en clasificación multinstancia asumen que el atributo de decisión es binario,y={+,-}, pero, en general, y puede tener mas de dos elementos, entonces se dice que el problema de clasificación es multiclase. La Tabla 1.1.2 muestra la forma en que se representan los datos en el aprendizaje multinstancia. Cada ejemplo X_i es descrito por T_i vectores de atributos y una etiqueta de decisión y_i. Observe que en nuestra notación los ejemplos se representan con letra mayúscula, para indicar que es una bolsa (un conjunto de vectores), mientras que las instancias dentro de la bolsa se representan con letra minúscula, indicando que son vectores simples. La tarea de la clasificación multinstancia es encontrar $f(X) = y$, basado en un conjunto de instancias de entrenamiento $D = \{(X_1, y_1)...(X_M, y_M)\}$, donde$f : \mathbb{N}^x \rightarrow Y$ es un concepto multinstancia. Un algoritmo de aprendizaje multinstancia construye, a partir de un conjunto de ejemplos de entrenamiento D, un clasificador o modelo de los datos $h(x)$, el cual es una hipótesis de $f(x)$.

Ejemplos	Instancias	Atributo 1	Atributo 2	...	Atributo N	Decisión
x_1	$x_{1,1}$	$x_{1,1,1}$	$x_{1,1,2}$	...	$x_{1,1,N}$	y_1
	...	...	...	...	...	
	x_{1,T_1}	$x_{1,T_1,1}$	$x_{1,T_1,2}$	...	$x_{1,T_1,N}$	
...		...	...	...	...	...
x_M	$x_{M,1}$	$x_{M,1,1}$	$x_{M,1,2}$	...	$x_{M,1,N}$	y_M
	...	...	...	...	...	
	x_{M,T_1}	$x_{M,T_1,1}$	$x_{M,T_1,2}$	...	$x_{M,T_1,N}$	

Cuadro 1.1.2: Representación de los datos en el aprendizaje multinstancia

Muchos algoritmos de clasificación multinstancia asumen la existencia de etiquetas de clase ocultas en las instancias y la existencia de una determinada relación entre las etiquetas de clase de las instancias y las etiquetas de clase de la bolsa. A esta relación asumida se le llama hipótesis multinstancia. Se han propuesto varias hipótesis multinstancia, pero la más ampliamente usada es conocida como hipótesis estándar. La hipótesis estándar establece que una bolsa es positiva si tiene al menos una instancia positiva y que una bolsa es negativa si todas sus instancias son negativas. Dado un conjunto de datos multinstancia no es posible saber a priori qué hipótesis multinstancia puede ofrecer mejores resultados en la clasificación.

1.1.5. Otros modelos donde está presente el aprendizaje multinstancia

Aunque en el aprendizaje multinstancia una gran parte de la investigación este centrada en los problemas de clasificación, no es esta la única linea investigativa que aborda este aprendizaje. Las mas notables de estas son la regresión multinstancia[1, 35], el agrupamiento multinstancia[22, 52] y el aprendizaje multinstancia multietiqueta[54].

Regresión multinstancia

En la regresión multinstancia las bolsas están asociadas con etiquetas de valores reales en lugar de las etiquetas de clase binarias usados en la clasificación, y la tarea es igualmente predecir estas

etiquetas[1, 35].Al igual que la clasificación multinstancia, la regresión multinstancia esta motivada por el problema de la predicción de la actividad farmacológica, ya que muchos desarrolladores de fármacos prefieren predicción de niveles de actividad de las drogas en vez de una clasificación en activo/inactivo.Otras áreas de aplicación identificadas es la predicción de la profundidad óptica de aerosoles para la investigación climática[47] y modelacion del rendimiento de las cosechas [43].

Agrupamiento multinstancia

Cuando en los problemas no se cuenta con las etiquetas de las bolsas de entrenamiento, el agrupamiento multinstancia permite formar grupos de bolsas según criterios de semejanza.Es uno de los métodos de aprendizaje mas populares dentro del aprendizaje no supervisado ya que puede ayudar a encontrar la estructura inherente a los datos y tener una comprensión de su distribución,por lo que puede ser usado como un paso de preprocesamiento para otras tareas, como la clasificación o la regresión[52].

Aprendizaje multinstancia multietiqueta

En el aprendizaje supervisado tradicional, los problemas de aprendizaje multiclase contienen más de dos categorías de clasificación, pero cada ejemplo pertenece a solo una de esas categorías.En el aprendizaje multietiqueta las categorías no son mutuamente exclusivas,por lo que cada ejemplo puede pertenecer a varias clases a la vez. Zhou & Zhang[54] formalizaron el aprendizaje multinstancia multietiqueta (MIME), donde cada bolsa multinstancia puede tener múltiples etiquetas de clase.

1.2. Importancia de la clasificación multinstancia

El aprendizaje multinstancia fue probado inicialmente con los algoritmos de rectángulos paralelos a los ejes para la predicción de la actividad farmacológica, pero ya hoy existen decenas de métodos de solución y siguen apareciendo nuevas propuestas que mejoran aspectos específicos de este problema. También ha crecido significativamente el número y la variedad de los campos de aplicación que se han beneficiado con el aprendizaje multinstancia. Algunos ejemplos de aplicaciones recientes son la anotación automática de imágenes[55, 39, 13], recuperación de imágenes/videos basados en su contenido[18, 33, 25, 26], seguimiento visual[53, 51, 30, 3], bioinformática[42, 14, 28], diagnóstico médico asistido por ordenador[10, 40, 31], imaginología[45, 34, 12] , reconocimiento de emociones[24, 41, 21, 23], detección de anomalías[50] , detección de minas terrestres[8, 7], entre otras.Por lo que ha despertado un gran interés entre los investigadores y profesionales de este ramo. En esta sección se argumentara con mas detalles sobre la importancia del aprendizaje multinstancia y, particularmente, de la clasificación multinstancia.

1.2.1. Surgimiento del aprendizaje multinstancia

El aprendizaje multinstancia fue definido formalmente en 1997 por Dietterich et al.[11]. Ellos trabajaban en el problema de predicción de la actividad de fármacos en la Corporación Farmacéutica Arris. Este problema es de gran importancia, pues permite ahorrar tiempo y dinero al concentrar los esfuerzos de biólogos y químicos en la síntesis y prueba de compuestos para los cuales se predice una alta actividad. Este problema consiste en encontrar moléculas que se enlacen estrecha y selectivamente

a una proteína dada. El factor mas importante para determinar si una molécula se enlaza a una proteína es su forma. Empleando el paradigma de aprendizaje supervisado se podría aprender la forma adecuada de una molécula a partir de una serie de ejemplos positivos y negativos. Un ejemplo positivo sera una molécula que se sabe que se enlaza a la proteína dada, mientras que un ejemplo negativo sería una molécula que no se enlaza con la proteína. Sin embargo, el principal problema de este enfoque es que las moléculas son objetos flexibles. Por tanto, una misma molécula puede tomar muchas conformaciones diferentes. La Figura 1.2.1 muestra un ejemplo de varias conformaciones que puede tomar una molécula de butano al rotar uno de sus enlaces.

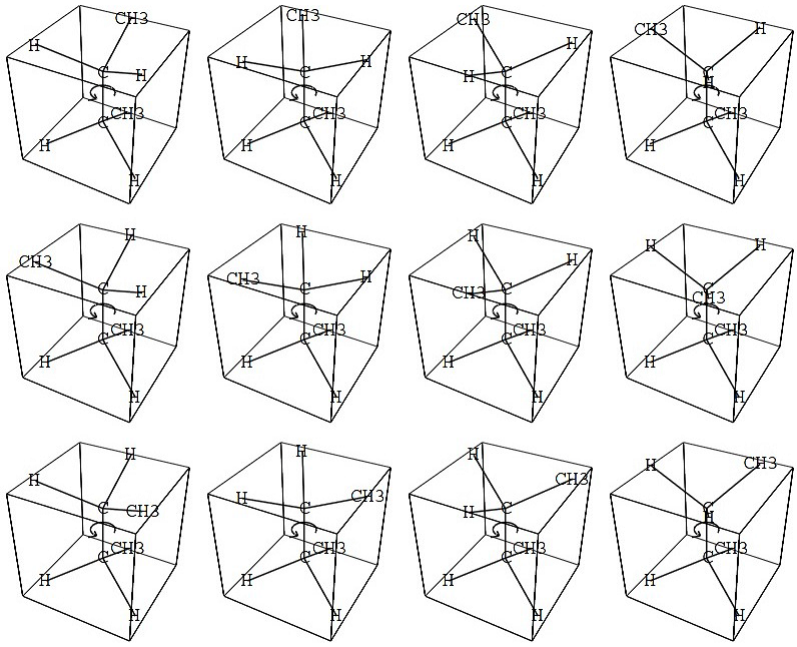

Figura 1.2.1: Diferentes conformaciones de una molécula de butano (C4H10)

Ya que no existe una única descripción para una molécula, es que se buscaron variantes para la representación de estos datos, dando paso así a la aparición del problema multinstancia. Por tanto, un ejemplo de entrenamiento esta formado por una colección de conformaciones. Si una molécula es positiva, sabemos que al menos una de sus posibles conformaciones es la correcta, aunque no sabemos cuál . Si una molécula es negativa, entonces ninguna de sus conformaciones es correcta.

El trabajo de Dietterich et al.[11] no solo ofreció una mejor solución al problema de predicción de la actividad de fármacos sino que fundo un nuevo paradigma del aprendizaje automatizado, importante tanto desde el punto de vista práctico, con aplicaciones a nuevos y más complejos problemas, como desde el punto de vista teórico por su conexión con el aprendizaje relacional y su capacidad para manejar la ambigüedad en los datos.

1.2.2. Ejemplos donde se puede aplicar la clasificación multinstancia

A continuación se muestran algunos ejemplos de aplicaciones de clasificación multinstancia donde se ilustran cómo este enfoque puede ser usado para solucionar determinados problemas de clasificación.

En la recuperación de imágenes basada en su contenido (CBIR, siglas del termino ingles content-based image retrieval) es una de las áreas de aplicación más frecuentes en el estudio del aprendizaje multinstancia. En CBIR, los usuarios presentan ejemplos de las imágenes que desean y la tarea es determinar si una nueva imagen sera de su interés. Una imagen es representada por un conjunto de segmentos (regiones de píxeles) que están caracterizadas por descriptores de color, textura y forma. Una bolsa representa una imagen y una instancia dentro de una bolsa representa un segmento de la imagen. No se conoce cuales segmentos y atributos de una imagen están relacionados con el contenido deseado, las etiquetas de clase de las instancias individuales son desconocidas[9, 27].

La clasificación multinstancia tiene también aplicaciones en la categorización de texto, donde la tarea es asignar etiquetas semánticas a los documentos de texto. Un documento puede ser representado como una bolsa y las instancias son obtenidas al dividir el documento en fragmentos más pequeños. Se pueden extraer atributos tales como las frecuencias de ocurrencia de las palabras en cada fragmento para formar las instancias.

Figura 1.2.2: Ejemplo de representación multinstancia en las variaciones en el trazado de los dígitos

Otra aplicación donde se puede hacer un buen uso de este modelo de clasificación es en el reconocimiento de números(o letras) manuscritos, debido a la diversidad de formas en las que un numero o letra puede ser dibujada(cada persona escribe de forma distinta). En la Figura 1.2.2 se puede apreciar un ejemplo limitado a números y con pocas formas de escribir cada uno,donde en cada fila hay múltiples representantes (instancias) de una misma clase de dígito.Este importante problema del campo del reconocimiento de patrones ha sido enfocado de muchas maneras, por ejemplo, en[38, 37] un conjunto de N transformaciones aplicadas a un patrón se interpreta como una variedad matemática N-dimensional del patrón. Los autores presentan una red neuronal de retropropagación para aprender el conjunto de variaciones de cada patrón. Desde el enfoque multinstancia cada clase de dígito (digamos, un 5) se corresponde con un ejemplo, y las distintas transformaciones de un dígito se pueden entender como distintas variaciones del mismo, por tanto, serán múltiples instancias del mismo ejemplo.

1.2.3. La ambigüedad como fuente de problemas multinstancia

Debido al polimorfismo, en el aprendizaje multinstancia se modelan los problemas a partir de entradas ambiguas.Un caso común de entrada polimorfa se ve reflejada cuando un objeto puede tener múltiples apariencias diferentes en el espacio de entrada y no se conoce cuál de ellas es responsable de la etiqueta del objeto.También cuando un objeto está compuesto por piezas o partes, cada una de las cuales tiene una representación en el espacio de entrada, y la etiqueta se asigna al objeto como un todo y no a las partes, aunque solo una o unas pocas sean responsables de la etiqueta.

1.3. Problema del desbalance de clases

Cuando hay una diferencia significativa entre el tamaño de las clases en los problemas de clasificación se dice que las clases están desbalanceadas.Tanto en el aprendizaje tradicional como en el multinstancia cuando estamos en presencia de este problema el desempeño de los sistemas de clasificación baja, afectando fundamentalmente la clase con menor cantidad de representantes(minoritaria).Por ejemplo, cuando se quieren encontrar enfermedades severas que afectan a una entre miles de personas(caso típico de desbalance de clases), aunque la enfermedad sea rara, es muy importante identificarla en la persona que la padece (acertar el caso positivo) para poder tratarla. También cuando se desea detectar operaciones financieras fraudulentas, aunque la gran mayoría de las operaciones sean legales, es sumamente importante detectar cualquier operación fraudulenta, por las grandes perdidas económicas que una sola de ellas puede ocasionar.

Aunque a este fenómeno se le ha dado en llamar *problema de las clases desbalanceadas*, en realidad la diferencia entre el tamaño de las clases no es la esencia del problema, sino una condición que favorece la manifestación de otros problemas que subyacen en segundo plano. Entre las causas primarias que dan lugar a los efectos observados tras el problema de las clases desbalanceadas, algunos de los más conocidos son los siguientes:

- Uso de medidas del desempeño inadecuadas para guiar el proceso de aprendizaje: Los algoritmos de aprendizaje que usen de forma explícita o implícita heurísticas basadas en maximizar medidas del desempeño tales como la exactitud de la clasificación (accuracy) tenderán a ignorar los ejemplos de la clase minoritaria. Como estas medidas tratan los aciertos de ejemplos positivos de igual manera que los aciertos de ejemplos negativos, al ser estos últimos mucho más abundantes, los aciertos de ejemplos positivos pasan a ser irrelevantes. Consideremos un conjunto de datos desbalanceado, que tenga, digamos, un ejemplo positivo por cada 100 negativos. Un clasificador que trate de maximizar el *accuracy* de su regla de clasificación podrá obtener una exactitud del 99 % con solo ignorar los ejemplos de la clase positiva, clasificando todos los ejemplos como negativos.

- Pequeño tamaño de la muestra: Para que el aprendizaje sea efectivo se necesitan conjuntos de ejemplos representativos de cada clase. Si el número de ejemplos de la clase minoritaria es muy bajo el desempeño de la clasificación puede afectarse por falta de representatividad [20].

- Solapamiento entre clases: Las clases pueden estar (en alguna medida) solapadas por una pobre definición intrínseca de los conceptos que ellas representan, o porque los atributos seleccionados no son suficientes para diferenciarlas adecuadamente[4]. Naturalmente, mientras mayor es el solapamiento entre las clases mas difícil se hace su separación. Sin embargo, las clases desbalanceadas amplifican el efecto: menores niveles de solapamiento son suficientes para observar un marcado deterioro del desempeño de la clasificación [32, 17].

- Presencia de subconceptos en la clase minoritaria: La presencia de subconceptos es otro factor que aumenta la complejidad de un problema de clasificación debido a que implica una descripción más larga de la frontera de decisión. Las fronteras de decisión más largas requieren más ejemplos para su especificación. Por eso, aunque los subconceptos pueden aparecer en cualquier clase, cuando están en la clase minoritaria, puede ser particularmente difícil su identificación por estar más esparcida la muestra [19].

Usualmente la tasa de desbalance (IR) es el medidor que marca cuán grande es el desbalance entre las clases, que se define como el cociente del número de ejemplos en la clase mayoritaria entre el número de ejemplos en la clase minoritaria . Mientras mayor es el IR, mayor es el desbalance entre las clases. Sin embargo, no puede decirse explícitamente a partir de que valor de IR es apreciable el deterioro de la clasificación, debido al carácter heterogéneo de la complejidad de los conjuntos de datos.

La mayoría de los algoritmos de aprendizaje tradicionales no son apropiados para datos con clases desbalanceadas, generando modelos de clasificación subóptimos en casos de una buena cobertura de los ejemplos de la clase mayoritaria, pero con frecuentes errores en la clasificación de los ejemplos de la clase minoritaria.

Actualmente es ampliamente reconocida la necesidad de emplear medidas robustas para evaluar la calidad de la clasificación cuando las clases están desbalanceadas, medidas que no sean falseadas por la prevalencia de una clase, y cuantifiquen fielmente cuán bueno es un clasificador, ya sea identificando ejemplos de la clase minoritaria, o de ambas clases, como mejor convenga a la aplicación. En la sección siguiente se presentan algunas de estas medidas, las cuales son usadas en este estudio.

1.3.1. ¿Cómo medir el desempeño de la clasificación cuando las clases están desbalanceadas?

En un problema de clasificación de dos clases hay cuatro posibles salidas, las cuales están representadas por la matriz de confusión que se muestra en la Tabla 1.3.1. Esta puede considerarse la información más básica sobre el desempeño de la clasificación, a partir de la cual se definen medidas más avanzadas.

	Clasificados como positivo	Clasificados como negativos
Realmente Positivo (AP)	Verdaderos Positivos (TP)	Falsos Negativos (FN)
Realmente Negativo (AN)	Falsos Positivos (FP)	Verdaderos Negativos (TN)

Cuadro 1.3.1: Matriz de confusión para un problema de dos clases

En este trabajo se usan las siguientes medidas de desempeño:

- *Precision y recall* : estas medidas se calculan a partir de la matriz de confusión. *Precision* es una medida de exactitud, ya que da la fracción de ejemplos clasificados como positivos que son realmente positivos. Por otro lado *recall*, también llamada exactitud de la clase positiva, es una medida que da la fracción de ejemplos de la clase positiva que han sido clasificados correctamente. Observe que estas medidas se centran en la identificación de la clase minoritaria.

$$Precision = \frac{TP}{TP+FP} \tag{1.3.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{1.3.2}$$

- *F1*: es también una medida usada en el campo de la recuperación de información. Se define como la media armónica de *precision* y *recall*. Por lo general, el objetivo principal de un algoritmo de aprendizaje es maximizar el *recall*, sin sacrificar la *precision*. Sin embargo, ambos objetivos entran en conflicto frecuentemente, y es difícil decidir el mejor entre dos algoritmos si uno aventaja al otro en *precision* y el otro aventaja al primero en *recall*. *F1* incorpora el equilibrio costo-beneficio de *recall* y *precision*, dando igual peso a ambas medidas, para evaluar el desempeño de un clasificador a través de un solo número. Al igual que *precision* y *recall*, *F1* se enfoca solo en la clase positiva.

$$F1 = \frac{2}{{}^{1}/_{recall} + {}^{1}/_{precision}} = \frac{2 \times recall \times precision}{recall + precision} \tag{1.3.3}$$

- AUC: es el area bajo una curva ROC. La curva ROC describe equilibrios entre beneficios (TP) y costos (FP) a lo largo de un rango de umbrales de un modelo de clasificación. Los gráficos ROC son consistentes para un problema dado aun si la distribución de instancias positivas y negativas esta altamente desbalanceada. El AUC permite medir el desempeño de un clasificador para evaluar cuál modelo es en promedio mejor usando un único valor numérico obtenido a partir de la curva ROC. Su valor representa la probabilidad de que un ejemplo positivo seleccionado aleatoriamente sea evaluado correctamente con más seguridad que un ejemplo negativo seleccionado aleatoriamente. AUC tiene un sólido significado estadístico ya que es equivalente a la prueba de rangos de Wilcoxon dados n puntos (TP_i; FP_i) de una curva ROC creada a partir de AP ejemplos positivos y AN ejemplos negativos usados para el cálculo exacto del AUC basado en la regla trapezoidal. El proceso de cálculo se ilustra en la figura 1.3.1 , donde $\triangle TP_i = TP_i - TP_{i+1}$, $\triangle FP_i = FPi - FP_{i+1}$ y $S_i = AN - FP_i$.AUC toma en cuenta la identificación correcta de ambas clases.

$$AUC = \frac{1}{AP \times AN} \sum_{i=1}^{n-1} \triangle TP_i \left(S_i + \frac{1}{2} FP_i \right) \tag{1.3.4}$$

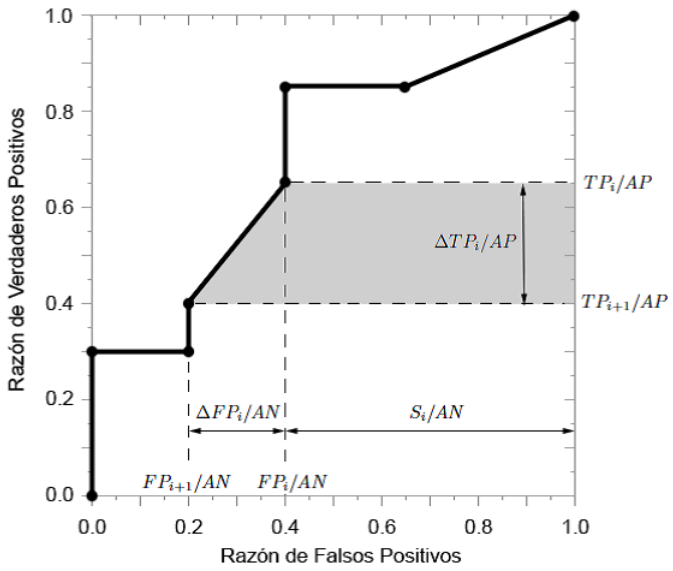

Figura 1.3.1: Ejemplo de curva ROC y el cálculo del AUC

- GMean: Es la media geométrica entre TPRate y TNRate. Donde TPRate es la tasa de instancias clasificadas correctamente como positivas respecto al total de positivas y TNRate la tasa de instancias clasificadas correctamente como negativas respecto al total de negativas.

$$TPRate=\frac{TP}{TP+FN} \tag{1.3.5}$$

$$TNRate=\frac{TN}{TN+FP} \tag{1.3.6}$$

$$GMean=\sqrt{TPRate\times TNRate} \tag{1.3.7}$$

1.3.2. Soluciones existentes en la clasificación tradicional

El problema de clases desbalanceadas en la clasificación monoinstancia puede enfrentarse por medio de cuatro tipos principales de soluciones:

1. Soluciones a nivel de algoritmo: Los algoritmos tradicionales de aprendizaje son adaptados para favorecer el aprendizaje de la clase pequeña. Se ha probado que estas soluciones son efectivas para ciertos algoritmos de aprendizaje de clasificadores y en algunos dominios de aplicaciones, sin embargo son difíciles de extender a otros contextos porque requieren un profundo conocimiento del algoritmo de aprendizaje y del por que fallan cuando la distribución de clases de los datos disponibles esta desbalanceada. Por este motivo, son preferibles las soluciones en las que la estrategia de aprendizaje subyacente permanece sin cambios.

2. Soluciones sensibles al costo: Se basan en los diferentes costos asociados a los errores de clasificación en las clases o incluso en ejemplos específicos, ya que el costo del error en la clasificación de

un ejemplo de la clase más pequeña es, por lo general, más elevado. La estrategia en este tipo de solución es minimizar el costo de la clasificación general. La principal desventaja del aprendizaje sensible al costo es que está basado en una matriz de costo para diferentes tipos de errores o ejemplos. Sin embargo, muchas veces no se dispone de una matriz de costo para un conjunto de datos dado.

3. Soluciones mediante técnicas de muestreo: Alteran la distribución de clases mediante un sobremuestreo o submuestreo del espacio de datos buscando el balance de las clases.

4. Soluciones de ensambles: Los ensambles construyen múltiples clasificadores a partir de los datos originales y luego agregan sus predicciones cuando clasifican ejemplos desconocidos. Los ensambles pueden utilizar técnicas de muestreo o sensibles al costo, los primeros se basan en la aplicación de una técnica de muestreo a una parte de los datos antes del entrenamiento de cada uno de los clasificadores que forman parte del ensamble. En los ensambles sensibles al costo el peso de los ejemplos se calcula tomando en cuenta, no solo la frecuencia de las clasificaciones erradas, sino también los costos de los errores.

1.3.3. Técnicas de muestreo

Para solucionar el problema del desbalance de clases existen varias opciones que implementan técnicas de muestreo. Se puede aplicar sobremuestreo (*oversampling),* que consiste en balancear la distribución de las clases añadiendo ejemplos de las clase minoritaria, algunos de los algoritmos que utilizan esta técnica son SMOTE, que genera nuevas instancias de la clase minoritaria interpolando los valores de las instancias minoritarias más cercanas a una dada y Resampling que duplica al azar instancias de la clase minoritaria.

Por lo contrario si se eliminan ejemplos de la clase mayoritaria también se puede llegar a solucionar este problema, a esta técnica se le conoce como submuestreo (*undersampling*) y la implementan algoritmos como Random undersampling, que elimina instancias al azar de la clase mayoritaria Tomek Links, que elimina sólo instancias de la clase mayoritaria que sean redundantes o que se encuentren muy cerca de instancias de la clase minoritaria y Wilson Editing o ENN(Editing Nearest Neighbor) que elimina aquellas instancias donde la mayoría de sus vecinos pertenecen a otra clase.

SMOTE

Para comprender como funciona SMOTE es necesario conocer el algoritmo de la regla del vecino más cercano, que consiste en suponer que instancias próximas entre sí tienen mayor probabilidad de pertenecen a la misma clase. El principal inconveniente del algoritmo es el alto coste computacional que tiene.

SMOTE (Syntetic Minority Over-sampling Technique) es un algoritmo de oversampling que genera instancias "sintéticas" o artificiales para equilibrar la muestra de datos basado en la regla del vecino más cercano. La generación se realiza extrapolando nuevas instancias en lugar de duplicarlas como hace el algoritmo de Resampling . Para cada una de las instancias minoritarias se buscan las instancias minoritarias vecinas (más cercanas) y se crean N instancias entre la línea que une la instancia original y cada una de las vecinas. El valor de N depende del tamaño de oversampling deseado. Para un caso del 200 % por cada instancia de la clase minoritaria deben crearse dos nuevas instancias genéricas.

Dependiendo de la cantidad de sobremuestreo que se vaya a realizar, se seleccionan aleatoriamente varios ejemplos de entre los k vecinos más cercanos. Sea x_0 un ejemplo de la clase minoritaria, y x_i uno

de los vecinos mas cercanos de x_0 seleccionado aleatoriamente, entonces se genera un nuevo ejemplo sintetico s_{0i} a partir de x_0 y x_i, calculando su valor mediante la siguiente fórmula .

$$s_{0i}=x_0 + \lambda(x_i - x_0) \tag{1.3.8}$$

Donde λ es un número aleatorio entre 0 y 1 generado para cada componente del vector s_{0i}. Esto causa la selección de un punto aleatorio a lo largo del segmento de línea entre los valores de dos atributos específicos. El proceso se ilustra en la figura 1.3.2 .

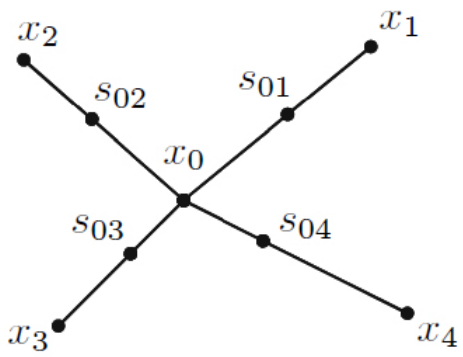

Figura 1.3.2: Ejemplo de interpolación aleatoria hecha por SMOTE

Alguna de las características que posee este algoritmo de sobremuestreo son :

- 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.

SMOTE es el algoritmo herramienta 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.
- Sólo funciona con variables continuas.
- No tiene una forma clara de decidir cuántos ejemplos generar.

Para minimizar los inconvenientes anteriores, después de usar SMOTE pueden aplicarse algoritmos de limpieza para reducir el ruido y el solapamiento de las clases como Tomek Links yWilson Editing (ENN).

1.3.4. Soluciones existentes en la clasificación multinstancia

El uso de amplificadores sensibles al costo en conjuntos de datos multinstancia desbalanceados es una de las lineas que se estudian para solucionar el problema de las clases desbalanceadas.Algunos de los más utilizados son algoritmos ensambles secuenciales que recalculan el peso de los datos después de cada iteración, de forma que el próximo clasificador débil en ser entrenado se centre más en los ejemplos que fueron mal clasificados la vez anterior.

El algoritmo ampliador más popular para clasificación multinstancia es el AdaBoost, el cual se describe a continuación.

Sea $T = \{(x_1, y_1); \cdots (x_M, y_M)\}$ un conjunto de ejemplos de entrenamiento donde cada instancia x_i pertenece a un espacio de instancias X y, enfocándose en problemas de clasificación de dos clases, cada etiqueta y_i pertenece al conjunto$Y = \{-1, +1\}$. El ampliador usa un algoritmo de aprendizaje base (también llamado débil) que acepta como entrada un conjunto de ejemplos de entrenamiento T junto con una distribución de pesos $D = \{1, \cdots, M\}$, sobre los índices de T. Dada esa entrada, el algoritmo de aprendizaje base calcula una hipótesis base (o débil) h. El signo de $h(x)$ es interpretado como la etiqueta que debe ser asignada a x. La idea de la amplificación es usar el algoritmo de aprendizaje base para formar una regla de predicción de gran exactitud al invocar el algoritmo de aprendizaje base repetidamente en diferentes distribuciones sobre los ejemplos de entrenamiento. Inicialmente los pesos son uniformemente distribuidos con valor $1/M$. En cada iteración el peso de los ejemplos correcta (o incorrectamente) clasificados por h es disminuido (o aumentado).

Los amplificadores pueden usarse con cualquier algoritmo de aprendizaje base siempre que este pueda manejar pesos en los ejemplos. Si se usa un algoritmo de aprendizaje multinstancia como algoritmo de aprendizaje base, entonces el amplificador será capaz de aprender a partir de conjuntos de datos multinstancia (en lugar de conjuntos de datos monoinstancia). En [46] se propone la estrategia de usar un clasificador multinstancia como algoritmo de aprendizaje base de un ensamble amplificador sensible al costo. Ellos incluyen en su estudio diversas variantes de ensambles amplificadores sensibles al costo. Cada variante introduce, de una forma diferente, información de costo dentro de la formula de actualización de pesos de AdaBoost, tabla 1.3.2 .

Nombre de la variante	modificación en la formula de actualización de pasos
Ab2	$K_t(X_i, y_i) = C_i exp(-\alpha_t y_i h_t(X_i))$
Ab3	$K_t(X_i, y_i) = C_i exp(-C_i \alpha_t y_i h_t(X_i))$
Ab4	$K_t(X_i, y_i) = C_i^2 exp(-C_i^2 \alpha_t y_i h_t(X_i))$

Cuadro 1.3.2: Variantes de AdaBoost sensible al costo usadas para clasificación multinstancia en problemas con clases desbalanceadas.

Los clasificadores base usados en [46] para los algoritmos amplificadores sensibles al costo son MITI[6] y MIRI[5]. En [46]muestran que en los conjuntos de datos multinstancia estudiados los algoritmos amplificadores sensibles al costo mejoran el desempeño de la clasificación tanto de los clasificadores base como de los amplificadores estándares.

Por otra parte una solución viable para enfrentar el problema de desbalance de clases en la clasificación multinstancia es utilizar técnicas de muestreo(sobremuestreo o submuestreo). En [36] se implementa el equivalente a SMOTE de la clasificación monoinstancia tradicional para la clasificación multinstancia, este algoritmo de preprocesamiento de datos es llamado MISMOTE y genera bolsas sintéticas del tipo de la clase minoritaria interpolando instancias escogidas aleatoriamente de pares

de bolsas hasta formar tantas bolsas como hagan falta para equilibrar el tamaño de las clases. En el capítulo 3 se explica más detalladamente como funciona.

Capítulo 2

Combinación submuestreo con sobremuestreo

Estudios realizados en la clasificación tradicional sugieren que la combinación del sobremuestreo de la clase minoritaria con el submuestreo de la clase mayoritaria puede mejorar la calidad de la clasificación. Siguiendo esta idea surge la interrogante de qué pasaría si utilizáramos esta técnica en la clasificación multinstancia. En este capítulo se implementa la combinación de submuestreo de la clase mayoritaria, usando submuestreo aleatorio, con sobremuestreo de la clase minoritaria, usando MISMOTE. Primeramente se explica en que consiste cada uno de los métodos de muestreo, luego se describe el proceso de combinación de las dos técnicas y después se realizan experimentos con distintas combinaciones de submuestreo y sobremuestreo para determinar si existe una combinación óptima que mejore significativamente la calidad de la clasificación de conjuntos de datos multinstancia afectados por el problema de clases desbalanceadas. .

2.1. Submuestreo de la clase mayoritaria

El submuestreo de la clase mayoritaria consiste en seleccionar ejemplos del conjunto que pertenece a la clase negativa o mayoritaria para formar un nuevo subconjunto de entrenamiento con un número reducido de los ejemplos de la clase mayoritaria y de esta forma disminuir el desbalance entre las clases positiva y negativa. El muestreo se hace sin remplazo, es decir, una vez que una instancia ha sido seleccionada para formar parte del subconjunto de la clase negativa, no vuelve a ser considerada en la próxima selección.

El submuestreo es equivalente a descartar una parte de los datos. Considere que hay N ejemplos en la clase mayoritaria. El submuestreo de Q ejemplos es equivalente a descartar N-Q ejemplos. Es por ello que este procedimiento debe aplicarse a clases que tengan un número considerablemente alto de instancias, ya que al submuestrear podría perderse información representativa de la clase en cuestión.

En general las técnicas de submuestreo pueden ser aleatorias o dirigidas. En el submuestreo dirigido se emplean diversas heurísticas para seleccionar los ejemplos de la clase mayoritaria. Por ejemplo, algunos se enfocan en seleccionar instancias que se encuentran en la frontera que divide las clases, otros eliminan los que más vecinos tengan de la clase contraria ya que estas son consideradas instancias ruidosas. La principal función del submuestreo dirigido es la de limpiar los datos y mejorar la definición

de la frontera de decisión, pero como el submuestreo dirigido no se puede usar de forma masiva no es útil para reducir significativamente el tamaño de la clase mayoritaria. Por este motivo, cuando se trata de enfrentar el desbalance de clases equilibrando el tamaño de las clase se usa por lo general el submuestreo aleatorio en lugar del dirigido. En el submuestreo aleatorio se seleccionan instancias al azar siguiendo una distribución uniforme, es decir todas las instancias tienen la misma probabilidad de ser seleccionadas, de esta forma se tiene una alta probabilidad de conservar las características de la clase representadas en el conjunto de datos original.

En este trabajo se utiliza el submuestreo aleatorio para reducir el tamaño de la clase mayoritaria en conjuntos de datos multinstancia con clases desbalanceadas. El objetivo es verificar si esta técnica de submuestreo permite mejorar la clasificación de conjuntos de datos multinstancia con clases desbalanceadas, ya que hasta donde se conoce no han sido reportado estudios sobre este método de solución. Para cumplir con este objetivo se realizaron los experimentos descritos en la sección 2.4.

El pseudocódigo del submuestreo aleatorio se muestra en el algoritmo 1. A partir de las entradas $D = \{B_1, B_2, \ldots, B_M\}$ y N, donde D es un conjunto de ejemplos(bolsas) y N la cantidad de ejemplos a eliminar, el objetivo es obtener D' que es un subconjunto de D representado por las $(M - N)$ bolsas restantes luego de que se escogieran aleatoriamente y eliminaran N ejemplos del conjunto D.

Input:
conjunto $D = \{X_1, \ldots, X_M\}$ de bolsas
número N de bolsas a eliminar
Output:
conjunto D' de bolsas con N bolsas menos

```
1 D' ←D
2 for i ← 1 to N do
3 |  Seleccionar una bolsa aleatoriamente de D' y quitarla de D'
4 return D'
```

Algorithm 1: Submuestreo aleatorio

2.2. Sobremuestreo de la clase minoritaria

El sobremuestreo de la clase minoritaria es todo lo contrario al submuestreo de la clase mayoritaria ya que con este se busca aumentar la cantidad de ejemplos que representan la clase minoritaria. Uno de los ejemplos más conocidos en la clasificación tradicional es el SMOTE, al cual se le han desarrollado algunas variantes. El SMOTE toma cada ejemplo de la clase minoritaria e introduce ejemplos sintéticos a lo largo del segmento de línea que une a algunos de los k vecinos más cercanos de la clase minoritaria, es decir, a partir de un ejemplo de entrenamiento P se genera un nuevo ejemplo sintético seleccionando aleatoriamente uno de los k vecinos más cercanos de P. Para crear una instancia sintética a partir de dos instancias, una de entrenamiento y una de sus k vecinas más cercana, se realiza el proceso de interpolación entre los dos vectores respectivos de cada instancia.

En el aprendizaje multinstancia existen muy pocos estudios sobre algoritmos de muestreo. MISMOTE es un algoritmo de sobremuestreo de la clase minoritaria, el equivalente de SMOTE pero para la clasificación multinstancia, introducido en [36]. En MISMOTE para generar una nueva bolsa se interpolan, tantas veces como instancias vaya a tener la bolsa, pares de instancias escogidas aleatoriamente, una de cada bolsa. La bolsa sintética resultante van a tener un tamaño igual al promedio de los tamaños de las k bolsas más cercanas a la bolsa de entrenamiento original. El algoritmo 2 representa

el método MISMOTE. Donde se refleja todo el proceso ocurrido para la generación de un conjunto de bolsas sintéticas.

Para una mejor comprensión del algoritmo se ilustra su funcionamiento con un ejemplo. Dado un conjunto de bolsas positivas con 5 bolsas que quiere sobremuestrear hasta 23 bolsas, primeramente el algoritmo el la línea 2 y 3 inicializan cinco conjuntos de vecinos más cercanos . De la línea 4 a la 8 genera 4 bolsas sintéticas por cada bolsa del conjunto de entrenamiento a través del algoritmo GenerarBolsa que se le pasa como parámetro una bolsa y el conjunto de sus k vecinos más cercanos. Por ultimo de la línea 9 a la 17 se generan las 3 bolsas restantes seleccionando para esto 3 instancias aleatorias de las 5 del conjunto de entrenamiento inicial y sus k vecinos más cercanos.

Input:
conjunto $D = \{X_1, \ldots, X_M\}$ de bolsas de entrenamiento positivas
número N de bolsas sintéticas a generar
número k de vecinos más cercanos

Output:
conjunto D' de bolsas sintéticas positivas

```
1  D' ← ∅ inicializar conjunto de bolsas sintéticas
2  for i ← 1 to M do
3      NN_i ← ∅ inicializar conjunto de vecinos más cercanos
4  if N ≥ M then
5      for i ← 1 to M do
6          NN_i ← calcular los k vecinos más cercanos de X_i
7          for j ← 1 to N div M do
8              GenerarBolsa (X_i,NN_i) y añadirla a D'
9  if N mod M > 0 then
10     L ← ∅ inicializar lista de bolsas positivas
11     L ←D
12     for j ← 1 to N mod M do
13         seleccionar aleatoriamente un X_i de L
14         remover X_i de L
15         if N < M then
16             NN_i ← calcular los k vecinos más cercanos de X_i
17         GenerarBolsa (X_i,NN_i) y añadirla a D'
18 return D'
```

Algorithm 2: MISMOTE: distribución de la generación

El algoritmo 3genera una bolsa sintética a partir de dos bolsas, una de entrenamiento y la otra seleccionada de sus k vecinos más cercanos. En las líneas 7 y 8 se observa que la selección de instancias

a interpolar dentro de una bolsa se hace de forma aleatoria.

Input:
bolsa X de partida para la generación de una nueva bolsa
conjunto NN de los k vecinos más cercanos de X

Output:
bolsa sintética Z

```
1  NumAttrs ← número de atributos
2  minClass ← valor de la clase positiva o minimal
3  Z ← ∅ inicializar una nueva bolsa sintética
4  Y ← escoger aleatoriamente una bolsa de NN
5  Size ← calcular el tamaño promedio de las bolsas en NN
6  for r ← 1 to Size do
7      x ← Seleccion aleatoria de una instancia de X
8      y ← Seleccion aleatoria de una instancia de Y
9      z ← inicializar instancia sintética
10     for a ← 1 to NumAttrs do
11         x_a ← valor del atributo a-ésimo de x
12         y_a ← valor del atributo a-ésimo de y
13         λ ← número aleatorio entre 0 y 1
14         z_a ← x_a + λ (y_a − x_a)
15     agregar z a Z
16 return Z
```

Algorithm 3: MISMOTE: Generación de la bolsa sintética

El algoritmo 2 permite aplicar la técnica de sobremuestreo al escenario multinstancia por medio de la generación de bolsas del algoritmo 3 con características similares a aquellas que le rodean. Los elementos principales de su diseño son la definición del tamaño de la bolsa sintética como el promedio de los tamaños de las bolsas más cercanas, y la selección de las instancias a ser interpoladas a través de muestreo simple aleatorio con remplazo. La selección del tipo de muestreo se hizo con la intensión de que la bolsa generada mantuviera las mismas características de las bolsas originales al tener instancias representativas del conjunto de entrenamiento. En la línea 3 y 4 se calculan la cantidad de bolsas a generar por cada bolsa positiva y la cantidad extra a generar para lograr el balance óptimo de ambas clases.

2.3. Remuestreo en ambas clases

Debido a que cada uno de los algoritmos de muestreo trabajan con una parte del conjunto de datos. El submuestreo se le aplica a la clase mayoritaria mientras que el sobremuestreo a la minoritaria, ninguno interfiere en el trabajo del otro.

Los únicos parámetros que necesitan estos algoritmos como entrada son los conjuntos de ejemplos de entrenamiento y el número de bolsas de la clase positiva debe construir MISMOTE y la cantidad de bolsas que debe seleccionar el submuestreo aleatorio de la clase mayoritaria. El resultado final es

un conjunto de datos balanceado donde el IR sea igual a 1.

Input:
conjunto desbalanceado $D = \{B_1, \ldots, B_N\}$
$pSubmuestreo$ porciento a aplicar submuestreo
$pSobremuestreo$ porciento a aplicar sobremuestreo

Output:
conjunto D balanceado

1 $cantPositivas \leftarrow$ cantidad de bolsas positivas en D
2 $cantNegativas \leftarrow$ cantidad de bolsas negativas en D
3 $cantToSub \leftarrow pSubmuestreo \times \frac{(cantNegativas - cantPositivas)}{100}$
4 $cantToSob \leftarrow pSobremuestreo \times \frac{(cantNegativas - cantPositivas)}{100}$
5 $D \leftarrow$MISMOTE(D,$cantToSob$) se sobremuestrea
6 $D \leftarrow$SubmuestreoAleatorio(D,$cantToSub$) se submuestrea
7 **return** D

Algorithm 4: Remuestreo en ambas clases

En el algoritmo 4, en la línea 3 se calcula el parámetro cantidad de bolsas a submuestrear, que utilizará en la línea 5 el algoritmo 1 y en la línea 4 se calcula la cantidad de bolsas a generar utilizada en el algoritmo 2, parámetros necesarios para garantizar el balance de las clases.

La experimentación con distintas combinaciones de submuestreo y sobremuestreo permitirá conocer si existe una combinación óptima, y si existe, cuál es para distintos dominios de aplicación. Los resultados se muestran en la sección 2.4.

2.4. Experimentos

Los experimentos se organizaron en un conjunto de pasos que se muestran en la figura 2.4.1.Primeramente se tomaron conjuntos de datos desbalanceados para luego aplicarles las técnicas de preprocesamiento, submuestreo y sobremuestreo el las distintas proporciones explicadas más adelante en esta sección. Como resultado se obtienen 6 variantes de preprocesamiento. A la izquierda de la figura, la variante identificada como 0-100 es aquella donde se aplica solamente submuestreo de la clase mayoritaria. A la derecha de la figura, la variante identificada como 100-0 es aquella donde se aplica solamente sobremuestreo de la clase minoritaria. El resto de las variantes, desde 20-80 hasta 80-20, tienen distintas combinaciones de muestreo. En los 6 casos los conjuntos de datos originales han quedado balanceados, pero en cada caso el balance se obtuvo de manera diferente.

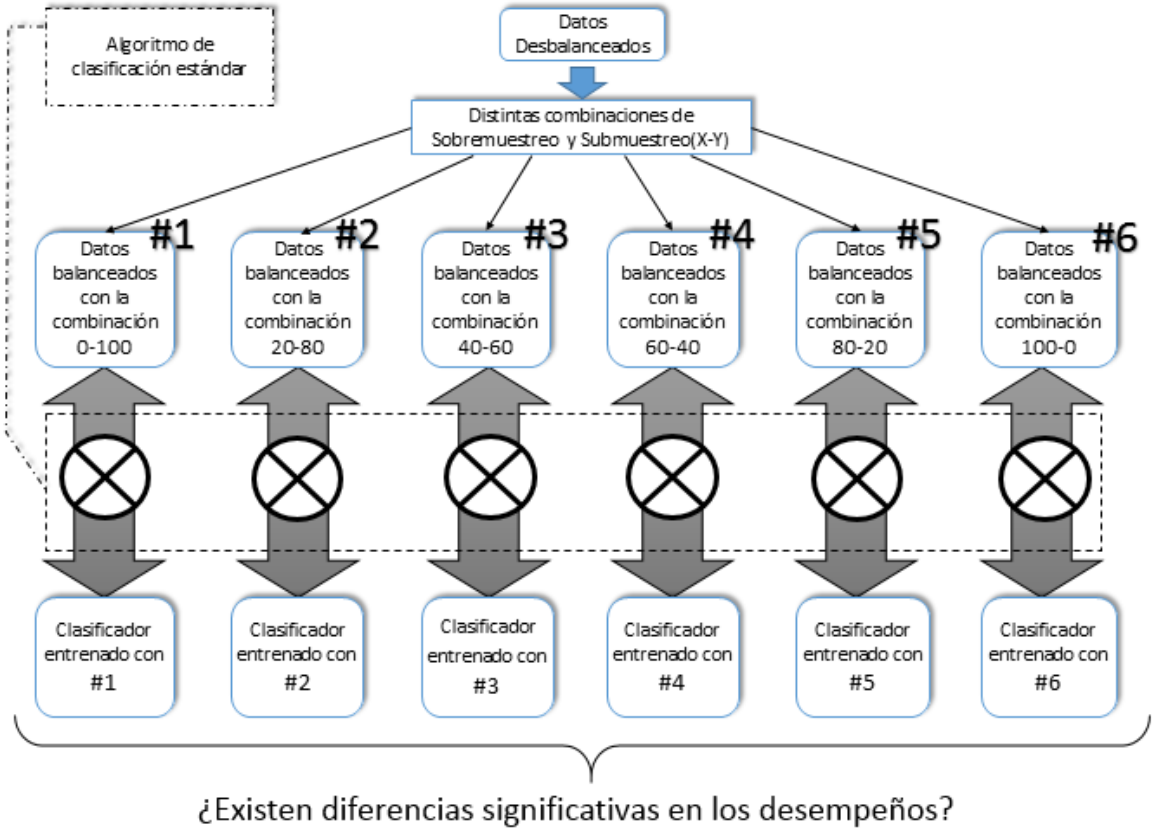

Figura 2.4.1: Diseño del experimento para determinar cuál es la mejor combinación de submuestreo y sobremuestreo

En el paso siguiente se entrena un algoritmo de clasificación multinstancia con cada una de las 6 variantes de preprocesamiento y se evalúa el clasificador resultante con un conjunto de datos desbalanceado de prueba para conocer su desempeño. Este paso se repite con varios algoritmos de clasificación y los desempeños de los clasificadores resultantes se promedian para obtener un desempeño medio en cada variante de preprocesamiento. El objetivo principal del experimento es determinar si existen diferencias significativas entre los desempeños medios de los clasificadores en las distintas variantes de preprocesamiento. También es de interés determinar si la variante 0-100, que representa el uso exclusivo de submuestreo de la clase mayoritaria, mejora significativamente la clasificación de los conjuntos de datos desbalanceados.

En el experimento vamos a comparar distintas combinaciones de sobremuestreo y submuestreo. De esta forma al preprocesar un conjunto de datos con la configuración 30-70 significa que vamos a aplicar sobremuestreo a un 30 porciento de la diferencia de clase y el 70 porciento restante va a ser submuestreado de la clase mayoritaria. Las diferentes combinaciones utilizadas en los experimentos son 0-100, 20-80, 40-60, 60-40, 80-20 y 100-0 aunque se pudieran escoger otros más y se obtendrían

resultados más exactos y detallados, pero entonces los experimentos serían más largos y complejos.

Dado que para distintos dominios de aplicación podría ser diferente el resultado de la combinación de MISMOTE de la clase minoritaria y submuestreo de la clase mayoritaria, replicamos este experimento para los dos dominios de aplicación presentes en el estudio.

A continuación se describes los conjuntos de datos, los algoritmos de clasificación multinstancia, los métodos de validación y las pruebas estadísticas utilizadas para la realización de los experimentos.

2.4.1. Conjuntos de datos

Para las pruebas se utilizan un total de 27 conjuntos de datos MIL desbalanceados y que pertenecen a tres dominios de aplicación distintos: texto, imagen y farmacológico. seis pertenecientes al problema de clasificación textual WIR por sus sigles en inglés Web Index page Recommendation, veinte del problema de clasificación de imágenes COREL y uno llamado Thioredoxin relacionado con la actividad farmacológica. En la tabla 2.4.1 se muestran algunas de las características más representativas de los conjuntos de datos utilizados, se observa que la taza de desvalance va desde 3,7 hasta 19,0.

Name	attrs	-inst	+inst	inst	-bags	+bags	bags	IR
Thioredoxin	8	23270	3341	26611	168	25	193	6,7
WIR1	304	2867	556	3423	92	21	113	4,4
WIR2	298	2867	556	3423	92	21	113	4,4
WIR3	303	2867	556	3423	92	21	113	4,4
WIR4	303	757	2666	3423	24	89	113	3,7
WIR5	302	757	2666	3423	24	89	113	3,7
WIR6	304	757	2666	3423	24	89	113	3,7
Corel1	9	7463	484	7947	1900	100	2000	19,0
Corel2	9	7593	354	7947	1900	100	2000	19,0
Corel3	9	7637	310	7947	1900	100	2000	19,0
Corel4	9	7188	759	7947	1900	100	2000	19,0
Corel5	9	7747	200	7947	1900	100	2000	19,0
Corel6	9	7645	302	7947	1900	100	2000	19,0
Corel7	9	7501	446	7947	1900	100	2000	19,0
Corel8	9	7558	389	7947	1900	100	2000	19,0
Corel9	9	7609	338	7947	1900	100	2000	19,0
Corel10	9	7223	724	7947	1900	100	2000	19,0
Corel11	9	7567	380	7947	1900	100	2000	19,0
Corel12	9	7667	280	7947	1900	100	2000	19,0
Corel13	9	7428	519	7947	1900	100	2000	19,0
Corel14	9	7595	352	7947	1900	100	2000	19,0
Corel15	9	7454	493	7947	1900	100	2000	19,0
Corel16	9	7691	256	7947	1900	100	2000	19,0
Corel17	9	7717	230	7947	1900	100	2000	19,0
Corel18	9	7515	432	7947	1900	100	2000	19,0
Corel20	9	7582	365	7947	1900	100	2000	19,0

Cuadro 2.4.1: Conjuntos de datos utilizados para las pruebas con el método de muestreo combinado

2.4.2. Algoritmos de clasificación

Como se representa en la gráfica 2.4.1 es necesario la utilización de algoritmos de clasificación en la realización de los experimentos. Estos deberán ser enfocados al aprendizaje multinstancia y cada uno será utilizado con cada uno de los conjuntos de datos mencionados anteriormente. Se Seleccionaron seis algoritmos de clasificación estándar. La selección fue hecha sobre la base de la diversidad en la forma de representación de la hipótesis de aprendizaje y el enfoque del método de aprendizaje. En la Tabla 2.4.2 se nombra cada algoritmo y los parámetros con que fueron usados en el experimento y se indica brevemente la justificación de su selección. En los algoritmos SimpleMI y MIWrapper se usa C4.5 como clasificador base debido a que es un clasificador ampliamente reconocido y ha sido incluido como uno de los 10 algoritmos más importantes en la minería de datos[48]. Para todos los algoritmos de la Tabla y el clasificador base C4.5 se usan las implementaciones disponibles en el paquete de aprendizaje automático WEKA . En cada algoritmo se establece la configuración por defecto establecida en WEKA, sino se especifica lo contrario.

Clasificador	Parámetros	Justificación de la selección
CitationKNN[44]	Citadores: 3 Referencias: 3	Encarna el enfoque de aprendizaje basado en instancias
MITI[6]		Encarna el enfoque de aprendizaje basado en árboles de decisión
MIEMDD[49]		Usa el enfoque de densidad diversa e hipótesis multinstancia estándar
SimpleMI	Clasificador base: C4.5	Transforman los datos multinstancia en simple instancia basado en distintas formas de relacionar instancias con bolsas
MIWrapper		
MISVM[2]	Kernel: lineal	Encarna el enfoque de aprendizaje basado en máximo margen

Cuadro 2.4.2: Clasificadores multinstancia utilizados en los experimentos con el método de muestreo combinado

2.4.3. Validación de la experimentación

La evaluación de la clasificación se llevo a cabo mediante un esquema de validación cruzada (CV, por las siglas del término ingles cross validation) 5 CV. Lo que significa que cada conjunto de datos fue dividido en cinco particiones distintas en las cuales se registro y promedió la medida de desempeño AUC(área bajo la curva), presentada en la sección 1.3.1. Para evaluar el desempeño de un clasificador en un conjunto de datos se guardaron los resultados de la clasificación (valores de la matriz de confusión) de cada una de las cinco particiones del conjunto de datos , y se calculó la medida del desempeño a partir de la matriz de confusión.

Se usaron pruebas estadísticos para probar la existencia de diferencias significativas entre los algoritmos comparados. Específicamente, las pruebas no paramétricas, de acuerdo a las recomendaciones hechas en [15, 16] , porque no existen garantías de que estén satisfechas las condiciones iniciales que aseguran la fiabilidad de las pruebas paramétricas.

Se usa la prueba de rankings de Friedman que verifica la hipótesis nula de que no hay diferencias significativas entre el desempeño promedio de los clasificadores en cada una de las 6 variantes de preprocesamiento. Los valores que se comparan en este caso son las medidas de desempeño (AUC) resultantes del promedio de la ejecución de los algoritmos de la tabla 2.4.2 en cada una de las 5 combinaciones de procesamiento de los conjuntos de datos. La prueba de Friedman asigna un ranking a cada variante de preprocesamiento. Un ranking menor significa que la variante obtuvo un mejor desempeño ya que fue mejor que las demás un mayor número de veces.

La prueba de Friedman calcula un valor de significación crítica p que, al ser indeseablemente conservador, es corregido por el método de Iman y Daveport. Si el valor p calculado por Iman y Daveport es menor que 0.05 significa que existe diferencias significativas entre al menos dos variantes de preprocesamiento. Para determinar en cuál de las variantes hay diferencias significativas emplea una prueba post-hoc que permita comparar cada una de las 6 variantes contra las 5 restantes (prueba NxN). Como comparar a A con B es igual que comparar a B con A, en total debemos verificar 6x5/2=15 comparaciones. La hipótesis nula que debe verificarse en cada comparación es que el desempeño medio de los clasificadores en las dos variantes de preprocesamiento comparadas son iguales.

La prueba post-hoc que se usa en esta tesis es la de Shaffer, pues es relativamente simple y al mismo tiempo potente. En la prueba se calcula el valor de p que es la probabilidad de que las diferencias observadas hayan sido producidas por azar. El valor de p se compara con el valor de significación crítica calculado por Shaffer. Si p es menor se rechaza la hipótesis nula concluyendo que la variante de preprocesamiento que tenga menor ranking (de Friedman) entre las comparadas es significativamente mejor que la otra. Si p no es menor, se retiene la hipótesis nula y solo se puede decir que la prueba no encuentra diferencias significativas entre las variantes comparadas.

2.5. Resultados

La tabla 2.5.1 muestra los promedios de los resultados de aplicar los clasificadores a todos los conjuntos de datos, teniendo en las filas los resultados devueltos por emplear un clasificador determinado a todos los conjuntos de datos agrupándolos por las diferentes combinaciones de sobremuestreo y submuestreo con las que estos han sido preprocesadas. Se incluyó la columna "sin preprocesar" para que se puedan apreciar las posibles ventajas de los métodos de preprocesamiento y también en la última fila se presenta el promedio de las columnas, es decir de los valores de AUC de todos los clasificadores que se usaron en el experimento. Las flechas hacia arriba y hacia abajo en una fila representan con cuál variante el clasificador especificado ha alcanzado el valor más bajo y el más alto respectivamente, si en una fila hay más de una flecha de un mismo tipo significa que el valor está repetido.

Clasificador	Sin preprocesar	0-100	20-80	40-60	60-40	80-20	100-0
Citation KNN	0,82856353	0,80842449↓	0,83093179	0,8430819	0,85374496	0,85576781	0,85714329↑
MITI	0,59298036↓	0,74292279	0,7558663	0,75741248↑	0,74529074	0,75115002	0,74599887
Simple	0,70644167↓	0,75566362	0,76965966	0,79060490	0,79879139	0,79394059	0,80264708↑
MIEMDD	0,84875270↑	0,82038943	0,78085914	0,77875791	0,77282420	0,76023171	0,75846840↓
MIWrapper	0,82436722↓	0,87918897	0,87916408	0,87903810	0,88326357	0,88481306↑	0,88368219
MISVM	0,52307335↓	0,64498522	0,64500630	0,65234549	0,66219930	0,66426990	0,66803208↑
Promedio	0,72069648↓	0,77526242	0,7769146	0,78354013	0,78601902↑	0,78502885	0,78599532

Cuadro 2.5.1: Resultados generales de las pruebas del algoritmo de muestreo combinado en la medida de desempeño AUC

Como se puede apreciar en la tabla 2.5.1, hay resultados diferentes en dependencia de qué clasificador se esté usando, aunque sí se puede comprobar que los valores más bajos prevalecen en la columna donde no se realiza ningún tipo de preprocesamiento y los valores promedios en la última fila se comportan de forma muy similar para todas las combinaciones.

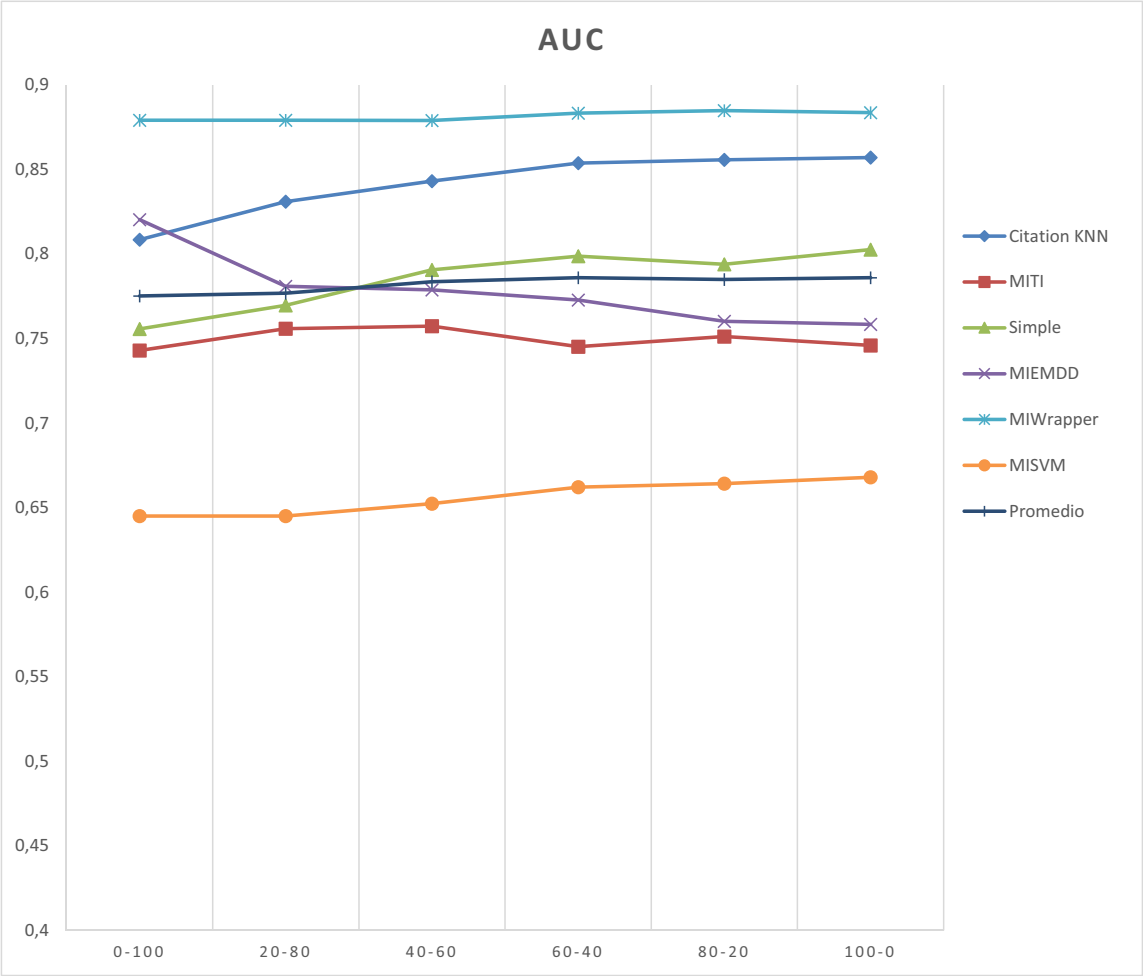

Figura 2.5.1: Gráfica con los resultados de la tabla 2.5.1

En la figura 2.5.1 se observa cómo se comportan la medida de desempeño AUC en los distintos clasificadores, si aumentan o no a medida que aumenta el porciento de datos sobremuestreado. Cada línea representa a un clasificador, estas tienen distintas alturas debido a las diferencias en los desempeños de los clasificadores en los conjuntos de datos estudiados, pero a la vez tienen poca variación en la componente vertical. La línea que representa al clasificador promedio, que está en una altura media entre el clasificador de más arriba y el de más abajo es casi completamente horizontal, lo cual indica que no existe diferencia en el comportamiento de los clasificadores en las distintas combinaciones de muestreo.

2.5.1. Pruebas estadísticas generales

Se realizan las pruebas estadísticas descritas anteriormente, los resultados obtenidos se muestran en el gráfico 2.5.2, donde mientras más pequeña es la barra, mejor es el resultado.

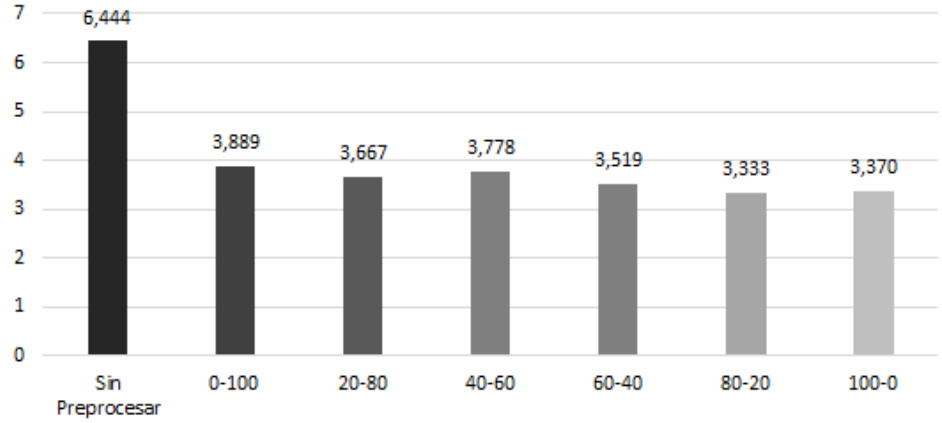

Figura 2.5.2: Prueba de Friedman para los resultados generales

Los resultados de todas las combinaciones son muy parejos aunque se diferencian del caso en que no se hace ningún procesamiento.Se observa que el submuestreo puro es mejor que no hacer ningún preprocesamiento, lo cual se verifica porque la segunda barra es menor que la primera.

La prueba de Iman y Daveport devuelve una significación de $1{,}781503 \times 10^{-8}$ que es menor que 0.05 por lo que se dice que al menos dos variables tienen diferencias significativas, que pueden ser entre cada uno de las combinaciones de muestreo y la variante sin preprocesar. Entonces se pasa a determinar cuáles son a través de la tabla 2.5.2 que presenta los resultados Post hoc.

AUC

algorithms	p	Shaffer
Sin Preprocesar vs. 80-20	**0**	**0.002381**
Sin Preprocesar vs. 100-0	**0**	**0.003333**
Sin Preprocesar vs. 60-40	**0.000001**	**0.003333**
Sin Preprocesar vs. 20-80	**0.000002**	**0.003333**
Sin Preprocesar vs. 40-60	**0.000006**	**0.003333**
Sin Preprocesar vs. 0-100	**0.000014**	**0.003333**
0-100 vs. 80-20	0.344704	0.003333
0-100 vs. 100-0	0.377822	0.003571
40-60 vs. 80-20	0.449692	0.003846
40-60 vs. 100-0	0.48835	0.004167
0-100 vs. 60-40	0.528733	0.004545
20-80 vs. 80-20	0.57075	0.005
20-80 vs. 100-0	0.614295	0.005556
40-60 vs. 60-40	0.659243	0.00625
0-100 vs. 20-80	0.705457	0.007143
60-40 vs. 80-20	0.752784	0.008333
60-40 vs. 100-0	0.801059	0.01
20-80 vs. 60-40	0.801059	0.0125
0-100 vs. 40-60	0.850107	0.016667
20-80 vs. 40-60	0.850107	0.025
80-20 vs. 100-0	0.949771	0.05

Cuadro 2.5.2: Post Hoc con la prueba de Shaffer para encontrar diferencias entre pares de combinaciones de muestreos en los resultados con todos los conjuntos de datos

En la tabla 2.5.2 las filas que están señaladas en negritas son las hipótesis rechazadas debido a que el valor de p es menor que el valor crítico de Holm o Shaffer. Entonces se puede concluir que existen diferencias significativas entre no aplicar ningún tipo de método de preprocesamiento y aplicar cualquier combinación de submuestreo y sobremuestreo, pero no existen diferencias significativas entre aplicar cualquiera de las combinaciones. Por lo que se puede decir que aplicar cualquiera de estas combinaciones de técnicas de muestreo mejora la calidad de la clasificación y que no hay una combinación mejor que otra significativamente.

En el caso particular *Sin Preprocesar vs. 0-100* se rechaza la hipótesis nula y se demuestra que el submuestreo puro es significativamente mejor que no hacer ningún preprocesamiento.

2.5.2. Pruebas estadísticas por dominio de aplicación

Ahora se pasa a probar estadísticamente los resultados como se hizo anteriormente pero para cada dominio de aplicación, primeramente para los conjuntos de datos pertenecientes a la clasificación textual WIR y luego para los Corel del dominio de clasificación de imágenes.

Conjuntos de datos WIR

Clasificador	Sin preprocesar	0-100	20-80	40-60	60-40	80-20	100-0
Citation KNN	0,7954396↑	0,58746359↓	0,65436490	0,73598154	0,76483632	0,78277205	0,79384890
MITI	0,66084999↓	0,68005820	0,70213192	0,7099923	0,71037410	0,72405774↑	0,71764362
Simple	0,62041891↓	0,63123696	0,63823372	0,68411277	0,73382912↑	0,68825021	0,71361254
MIEMDD	0,75686012	0,68092220↓	0,70444072	0,70812178	0,73449776	0,76323860	0,76364809↑
MIWrapper	0,86375920	0,85112215	0,83511897↓	0,86257425	0,87299421↑	0,86502953	0,86857614
MISVM	0,60299674↓	0,6416260	0,6362565	0,6625284	0,67301084	0,66978472	0,68974086↑
Promedio	0,7167208	0,67873818↓	0,69509111	0,72721850	0,74825706	0,74885547	0,75784502↑

Cuadro 2.5.3: Resultados para los conjuntos de datos WIR de las pruebas del algoritmo de muestreo combinado

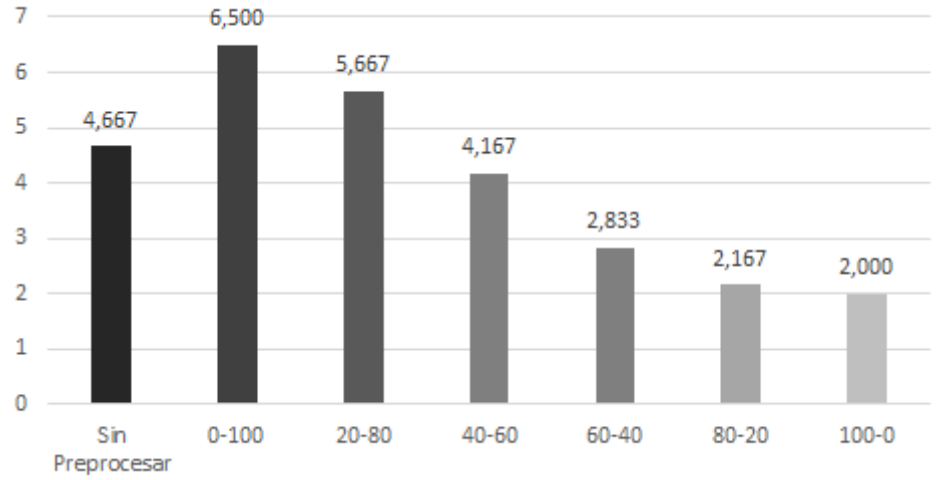

Figura 2.5.3: Prueba de Friedman para los resultados con los conjuntos de datos WIR

En este dominio de aplicación se observa un comportamiento distinto porque a medida que se disminuye el submuestreo y aumenta el sobremuestreo mejora la calidad de la clasificación. En este caso se observa que el submuestreo puro no es mejor que no hacer ningún preprocesamiento.

La prueba de Iman y Daveport devuelve una significación de $8{,}624074238497644 \times 10^{-6} < 0{,}05$ por lo que se dice que existen diferencias significativas, las cuales según el gráfico 2.5.3 se deben encontrar entre aplicar solamente submuestreo y donde se aplican los mayores porcentajes de sobremuestreo. La tabla 2.5.4 de Post hoc dá la respuesta exacta de entre cuáles combinaciones existen diferencias significativas:

AUC

algorithms	p	Shaffer
0-100 vs. 100-0	**0.000309**	**0.002381**
0-100 vs. 80-20	**0.000512**	**0.003333**
0-100 vs. 60-40	**0.003283**	**0.003333**
20-80 vs. 100-0	**0.003283**	**0.003333**
20-80 vs. 80-20	0.005012	0.003333
20-80 vs. 60-40	0.023103	0.003333
Sin Preprocesar vs. 100-0	0.032509	0.003333
Sin Preprocesar vs. 80-20	0.045021	0.003571
0-100 vs. 40-60	0.061369	0.003846
40-60 vs. 100-0	0.082352	0.004167
40-60 vs. 80-20	0.108809	0.004545
Sin Preprocesar vs. 0-100	0.141579	0.005
Sin Preprocesar vs. 60-40	0.141579	0.005556
20-80 vs. 40-60	0.229102	0.00625
40-60 vs. 60-40	0.285049	0.007143
Sin Preprocesar vs. 20-80	0.422678	0.008333
0-100 vs. 20-80	0.504036	0.01
60-40 vs. 100-0	0.504036	0.0125
60-40 vs. 80-20	0.59298	0.016667
Sin Preprocesar vs. 40-60	0.6885	0.025
80-20 vs. 100-0	0.893695	0.05

Cuadro 2.5.4: Post Hoc con la prueba de Shaffer para encontrar diferencias entre pares de combinaciones de muestreos en los resultados con los conjuntos de datos WIR

Se puede entonces plantear que en el caso particular del dominio de aplicación WIR existen diferencias entre 0-100 y 100-0, una aplica solamente submuestreo mientras que la otra sobremuestreo, demostrando que para este dominio de aplicación el sobremuestreo proporciona más calidad en los resultados de la clasificación que el submuestreo aleatorio en un 100 porciento. También existen diferencias significativas entre 0-100 y 80-20, entre 0-100 y 60-40, y entre 20-80 y 100-0, demostrando así lo mal que se comporta el submuestreo aleatorio en los conjuntos de datos WIR.

En el caso particular *Sin Preprocesar vs. 0-100* no se rechaza la hipótesis nula y no se puede demostrar que el submuestreo puro es significativamente mejor que no hacer ningún preprocesamiento en este dominio de aplicación.

Conjuntos de datos Corel

Clasificador	Sin preprocesar	0-100	20-80	40-60	60-40	80-20	100-0
Citation KNN	0,85037540↓	0,8830244	0,89336↑	0,888528	0,891227	0,885896	0,885
MITI	0,57488158↓	0,77185526	0,78334211↑	0,78369737	0,77023684	0,77036842	0,76768421
Simple	0,5017002↓	0,79808	0,81772040	0,82676974	0,82872434	0,83167303	0,83682237↑
MIEMDD	0,88081763↑	0,86694605	0,81077237	0,80209342	0,79006316	0,76800790	0,76201974↓
MIWrapper	0,81387513↓	0,89312105	0,89334474↑	0,888263	0,89094605	0,8947	0,89239276
MISVM	0,50025↓	0,65356579	0,65488158	0,65690790	0,66706579	0,67082895↑	0,66992105
Promedio	0,68698333↓	0,81109792↑	0,80890386	0,8077099	0,80637718	0,80357907	0,80234002

Cuadro 2.5.5: Resultados para los conjuntos de datos Corel de las pruebas del algoritmo de muestreo combinado

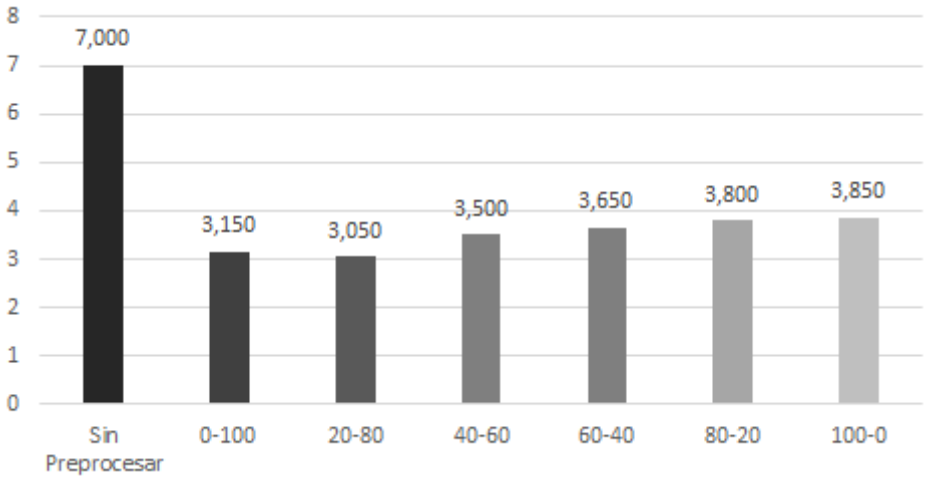

Figura 2.5.4: Prueba de Friedman para los resultados con los conjuntos de datos Corel

En este dominio de aplicación se observa un comportamiento similar al observado en el caso general porque se muestran todas las combinaciones de sobremuestreo y submuestreo muy parejas y a la vez con mejores resultados que si no se aplicara preprocesamiento. En este dominio de aplicación sí se cumple que el submuestreo puro es mejor que no hacer ningún preprocesamiento, aunque no se sabe si es significativamente mejor.

La prueba de Iman y Daveport devuelve una significación de $1{,}0219225636080678 \times 10^{-10} < 0{,}05$ por lo que se dice que existen diferencias significativas. Entonces se pasa a determinar donde están las diferencias a través de la tabla 2.5.6 :

AUC

algorithms	p	Shaffer
Sin Preprocesar vs. 20-80	**0**	**0.002381**
Sin Preprocesar vs. 0-100	**0**	**0.003333**
Sin Preprocesar vs. 40-60	**0**	**0.003333**
Sin Preprocesar vs. 60-40	**0.000001**	**0.003333**
Sin Preprocesar vs. 80-20	**0.000003**	**0.003333**
Sin Preprocesar vs. 100-0	**0.000004**	**0.003333**
20-80 vs. 100-0	0.241567	0.003333
20-80 vs. 80-20	0.272254	0.003571
0-100 vs. 100-0	0.305507	0.003846
0-100 vs. 80-20	0.341349	0.004167
20-80 vs. 60-40	0.379775	0.004545
0-100 vs. 60-40	0.464214	0.005
20-80 vs. 40-60	0.510068	0.005556
0-100 vs. 40-60	0.608408	0.00625
40-60 vs. 100-0	0.608408	0.007143
40-60 vs. 80-20	0.660549	0.008333
60-40 vs. 100-0	0.769698	0.01
40-60 vs. 60-40	0.8262	0.0125
60-40 vs. 80-20	0.8262	0.016667
0-100 vs. 20-80	0.883617	0.025
80-20 vs. 100-0	0.941653	0.05

Cuadro 2.5.6: Post Hoc con la prueba de Shaffer para encontrar diferencias entre pares de combinaciones de muestreos en los resultados con los conjuntos de datos Corel

Entonces en el caso particular del dominio de aplicación Corel existen diferencias en la medida AUC entre no aplicar técnicas de muestreo y aplicar cualquier combinación. En el caso particular de la comparación entre no aplicar ninguna técnica de preprocesamiento y el submuestreo de la clase mayoritaria puro (0-100) se demuestra que este último es significativamente mejor.

2.6. Conclusiones parciales

A partir de los experimentos realizados anteriormente se llegan a las siguientes conclusiones:

En conjuntos de datos multinstancia afectados por el problema del desbalance de clases el submuestreo de la clase mayoritaria es un método que puede ser útil para mejorar la calidad de la clasificación en algunos dominios de aplicación. Particularmente, en nuestros experimentos, el método de submuestreo mejoró la clasificación en la aplicación Corel perteneciente al dominio del reconocimiento de imágenes. Sin embargo, en la aplicación WIR del dominio de clasificación textual no mejoró la calidad de la clasificación, lo cual sugiere que la eficacia de este método puede estar condicionada por las características del problema.

De forma general no se aprecian diferencias significativas en la calidad de la clasificación obtenida tras aplicar distintas combinaciones de sobremuestreo y submuestreo. Esto significa que no encontramos

una diferencia importante entre aplicar, por ejemplo, un 40 % de sobre submuestreo con un 60 % de sobremuestreo y lo contrario, un 60 % de submuestreo con un 40 % de sobremuestreo. Aunque debemos notar que encontramos algunas diferencias en función del dominio de aplicación. Particularmente, se vió que en la aplicación WIR el submuestreo de la clase mayoritaria no favorece la calidad de la clasificación y a medida que se disminuye el submuestreo y se aumenta el sobremuestreo de la clase mayoritaria se obtienen mejores desempeños de la clasificación. Esto no sucede así en la aplicación Corel para la cual ambas técnicas (sobremuestreo y submuestreo) en cualquiera de sus combinaciones tienen aproximadamente igual desempeño, mejorando en todos los casos en una proporción semejante la calidad de la clasificación cuando no se realiza ningún tipo de procesamiento para enfrentar el problema de clases desbalanceadas.

Capítulo 3

Modificación del Algoritmo MISMOTE

Aplicar técnicas de muestreo en la clasificación multinstancia ha dado buenos resultados, al igual que en la clasificación tradicional. MISMOTE es un algoritmo de preprocesamiento que sobremuestrea la clase minoritaria hasta igualar el tamaño de ambas clases. En este capítulo se le hace una modificación al método de sobremuestreo MISMOTE para intentar mejorarlo. Primeramente se explica en que consiste la modificación y los cambios realizados en el algoritmo para implementarla y luego se le realizan pruebas, primero con conjuntos de datos artificiales y después con datos reales, para comprobar la eficacia del algoritmo modificado.

3.1. ¿Cómo mejorar MISMOTE?

El algoritmo de preprocesamiento de datos MISMOTE tiene como objetivo principal crear nuevos ejemplos interpolando pares de los ejemplos dados inicialmente hasta llegar a equilibrar el tamaño de las clases. Es por esto que si se quiere mejorar deben introducirse modificaciones en la forma que tiene de realizar su trabajo principal, la interpolación de bolsas.

En algunos conjuntos de datos el muestreo aleatorio con distribución uniforme de MISMOTE, puede provocar una falta de representatividad de las instancias positivas y por tanto podrían generarse bolsas negativas en la clase minoritaria. Este podría ser el caso de los conjuntos de datos donde subyace la hipótesis estándar porque como en este tipo de hipótesis basta unas pocas instancias positivas (tal vez solo una) para que la bolsa sea positiva es muy probable que cuando MISMOTE seleccione una instancia con probabilidad uniforme no escoja la instancia positiva y aunque el muestreo se aplica varias veces (tantas como instancias tenga la bolsa generada) podrían finalmente quedar excluidas de la nueva bolsa las instancias positivas.

Surge entonces la interrogante de qué pasaría si en lugar de muestrear con distribución uniforme, MISMOTE muestrea con distribución proporcional a la probabilidad de cada instancia de ser positiva. Se espera entonces que la bolsa generada tenga más probabilidad de pertenecer a la clase minoritaria (positiva),y así entonces se mejoraría la calidad de la clasificación en conjuntos de datos donde subyace la hipótesis estándar.

3.2. Probabilidad de una instancia de ser positiva

Para encontrar las instancias adecuadas para el proceso de interpolación de dos bolsas hay que tener en cuenta un conjunto de requisitos que estas deben cumplir si es que son positivas, como que no pueden estar presentes en ninguna de las bolsas negativas y sí en las positivas. La intersección de las bolsas positivas menos la unión de las bolsas negativas da como resultado instancias de las bolsas positivas que no están en las bolsas negativas, esto es lo esencial que se debe conocer cuando se desea conocer la probabilidad de una instancia de ser positiva.

3.2.1. Densidad Diversa

Sean $D = \{B_1^+, \cdots, B_p^+, B_1^-, \cdots, B_n^-\}$ el conjunto de bolsas de entrenamiento con p bolsas positivas y n negativas, la probabilidad de que una instancia x sea el punto t de máxima densidad diversa es

$$Pr(x = t|B_1^+, \cdots, B_p^+, B_1^-, \cdots, B_n^-)$$

Usando la regla de Bayes y asumiendo que las bolsas son condicionalmente independientes esto es equivalente a

$$\prod_{i=1}^{n} (Pr(x = t|B_i^+)) \prod_{i=1}^{m} (Pr(x = t|B_i^-)) \tag{3.2.1}$$

Esta es una definición general de densidad diversa, pero es necesario definir los términos del producto para instanciarla. En jjjj se proponen dos modelos para el cálculo de la densidad diversa, el modelo Noisy-Or y el modelo Most Likely Cause. En esta tesis se propone otro modelo basado en Most Likely Cause empleando operadores OWA. A continuación se definen los tres modelos usados para el cálculo de la densidad diversa. En la notación se usa $D = \{B_1, \cdots, B_k\}$ cuando no se tiene en cuenta la etiqueta de la bolsa, donde $k = n + p$, y_i es la etiqueta de clase de la bolsa B_i que toma los valores de 1 o -1 en dependencia de si la instancia en cuestión pertenece a una bolsa positiva o negativa respectivamente, B_{ij} representa la instancia j-ésima de la bolsa i-ésima y $P(B_{ij}^+|x^+)$ la probabilidad de que B_{ij}^+sea positiva si x lo es.

Modelo Noisy-Or

Este modelo asume que el evento solo puede ocurrir si al menos una de las causantes ocurre. También asume que la probabilidad de que cualquier causa que no ocasione el evento es independiente de cualquier otra causa. Por tanto el modelo Noisy-Or se define mediante las ecuaciones (3.2.2) y (3.2.3):

$$Pr(x = t|B_i^+) = \frac{1 - \prod_{j=1}^{p} (1 - P(B_{ij}^+|x^+))}{Z} \tag{3.2.2}$$

$$Pr(x = t|B_i^-) = \frac{1 - \prod_{j=1}^{p} (1 - (1 - P(B_{ij}^+|x^+)))}{Z} \tag{3.2.3}$$

Donde Z es una constante de normalización. Tomando $Z = 1$ y sustituyendo (3.2.2) y (3.2.3) en (3.2.1) se obtiene la probabilidad $P(x^+|D)$ de que x sea positiva dado un conjunto de entrenamiento D en la ecuación (3.2.4):

$$P(x^+|D) = \prod_{i=1}^{k} \left[\frac{1+y_i}{2} - y_i \prod_{j=1}^{B_i} (1 - P(B_{ij}^+|x^+)) \right] \tag{3.2.4}$$

Modelo Most Likely Cause

En este modelo la probabilidad $Pr(x = t|B_i)$ se estima buscando solamente la instancia de la bolsa que es más cercena al punto objetivo t de forma que requiere menos asunciones de independencia. El modelo Most Likely Cause se define mediante las ecuaciones (3.2.5) y (3.2.6):

$$Pr(x = t|B_i^+) = \frac{\max_j Pr(B_{ij}^+|x^+)}{Z} \tag{3.2.5}$$

$$Pr(x = t|B_i^-) = \frac{1 - \max_j (1 - Pr(B_{ij}^+|x^+))}{Z} \tag{3.2.6}$$

Tomando $Z = 1$ y sustituyendo (3.2.5) y (3.2.6) en (3.2.1) se obtiene la ecuación (3.2.7):

$$P(x^+|D) = \prod_{i=1}^{k} \left[\frac{1+y_i}{2} - y_i \max_{j=1}^{B_i} (P(B_{ij}^+|x^+) \right] \tag{3.2.7}$$

Modelo Most Likely Cause con operador OWA

El modelo Most Likely Cause utiliza el operador máximo que puede ser afectado por valores extremos. Por tanto en esta tesis proponemos el modelo Most Likely Cause con operador OWA.

Los operadores OWA modelan un proceso de agregación en el cual una secuencia A de n valores escalares son ordenados decrecientemente y pesados según el valor correspondiente a sus posiciones en el vector de peso $W = \{w_i\}$, donde $w_i \in [0, 1]$ y $\sum_i^n w_i = 1$.El operador OWA sobre una colección A utilizando un vector de pesos W se define como

$$OWA_W(A) = \sum_{i=1}^{n} w_i a_i$$

donde w_i y a_i son los elementos $i - ésimo$ del vector de peso y la colección respectivamente.

Una de las ventajas del uso de los operadores OWA es su flexibilidad dado que permite modelar disímiles estrategias de agregaciones. Por ejemplo los operadores máximo, mínimo y promedio pueden ser modelados a través de los operadores OWA utilizando los vectores de pesos W_{max}, W_{min} y W_{avg} respectivamente. Estos vectores de peso se definen de la siguiente manera

$$W_{max} = \{w_i\},\ donde\, w_1 = 1, w_i = 0, i \neq 1$$

$$W_{min} = \{w_i\},\ donde\, w_n = 1, w_i = 0, i \neq n$$

$$W_{avg} = \{w_i\},\ donde\, w_i = \frac{1}{n}, i = 1, \ldots, n$$

Definiendo un vector de peso W cuyos valores decrezcan monótona y suavemente obtenemos un operador máximo donde el efecto de los valores extremos es suavizado.

El modelo Most Likely Cause con operador OWA se define mediante las ecuaciones (3.2.8) y (3.2.9):

$$Pr(x = t|B_i^+) = \frac{OWA_W \underbrace{Pr(B_{ij}^+|x^+)}_{j}}{Z} \tag{3.2.8}$$

$$Pr(x = t|B_i^-) = \frac{1 - OWA_W \underbrace{(1 - Pr(B_{ij}^+|x^+))}_{j}}{Z} \tag{3.2.9}$$

Tomando $Z = 1$ y sustituyendo (3.2.8) y (3.2.9) en (3.2.1) se obtiene la ecuación (3.2.7):

En este modelo también se escoge como en el caso anterior la probabilidad mayor $P(B_{ij}^+|x^+))$, pero para ello se usan operadores OWA.

$$P(x^+|D) = \prod_{i=1}^{k} \left[\frac{1 + y_i}{2} - y_i OWA_W(\underbrace{P(B_{ij}^+|x^+))}_{j \in [1,\ldots,|B_i|]} \right] \tag{3.2.10}$$

El vector de peso W usado con el operador OWA es un vector lineal definido por la ecuación 3.2.11, donde p es el número de instancias de la bolsa B_i.

$$\text{Wmax-L} = \left\langle \frac{2}{p+1}, \frac{2(p-1)}{p(p+1)}, \ldots, \frac{4}{p(p+1)}, \frac{2}{p(p+1)} \right\rangle \tag{3.2.11}$$

En esta investigación se usa la ecuación (3.2.12)

$$P(B_{ij}^+|x^+) = e^{-[Dcos(x,B_{ij})]^2} \tag{3.2.12}$$

para calcular la probabilidad que tiene una instancia B_{ij}^+de ser positiva dada una instancia x^+que se asume positiva, donde $Dcos$ la distancia coseno entre dos instancias. Esta fórmula toma en cuenta la distancia que existe entre las dos instancias de forma que mientras más lejanas estén menor será la probabilidad.

3.2.2. Selección de una instancia teniendo en cuenta su probabilidad de ser positiva

Para esto se desarrolló un método que escoge aleatoriamente dentro del rango de los pesos(probabilidad) y luego devuelve el objeto al que pertenece ese peso.A continuación se explica mejor con un ejemplo:

1) Suponiendo que se tiene un conjunto de X con un peso de 4, un conjunto Y con peso 2 y un conjunto$Z$ con peso 14 . No es necesario que todos los promedios sumen 100, estos valores de peso pueden ser cualquier número tanto grande como pequeño.

xxxxyyzzzzzzzzzzzzzz

2) Ahora se escoge un número aleatorio entre 1 y la suma de todos los peso que en este caso es 20

3) por ejemplo si el número es 9, se busca a que letra pertenece ese peso en la sucesión anterior, en la posición 9 esta una Z, este es el objeto escogido finalmente, si es 5 entonces se escoge una Y.

Como se puede apreciar en el ejemplo hay más probabilidad de que se escoja una Z que una Y debido a que el peso que tiene la misma es mayor que el de Y.

En el la nueva versión del algoritmo MISMOTE se utiliza esta técnica para dado un vector de pesos, correspondientes a las probabilidades de cada una de las instancias de la bolsa de ser positivas, escoger aleatoriamente una instancia teniendo en cuenta su probabilidad correspondiente.

3.3. Nueva versión de MISMOTE

A continuación se muestran los algoritmos utilizados para la modificación del MISMOTE. El algoritmo 2 no se modifica ya que los cambios se van a realizar en el proceso de la generación de una bolsa, proceso realizado en el algoritmo 5 .

Las modificaciones son añadidas en las líneas 9 y 10 para que en lugar de seleccionar instancias aleatoriamente, se seleccionen teniendo en cuenta la probabilidad de ser positivas, para lo cual primeramente se calculan las probabilidades de todas las instancias de una bolsa de ser positivas con el método Probabilidad en las líneas 7 y 8, que devuelve un arreglo con las probabilidades de cada instancia de la bolsa de ser positiva utilizando el enfoque de Densidad Diversa. Luego con el método

descrito en la sección (3.2.2) se seleccionan las instancias teniendo en cuenta su probabilidad.

Input:
bolsa X de partida para la generación de una nueva bolsa
conjunto NN de los k vecinos más cercanos de X

Output:
bolsa sintética Z

```
1  NumAttrs ← número de atributos
2  minClass ← valor de la clase positiva o minimal
3  Z ← ∅ inicializar una nueva bolsa sintética
4  Y ← escoger aleatoriamente una bolsa de NN
5  Size ← calcular el tamaño promedio de las bolsas en NN
6  for r ← 1 to Size do
7      probX ← Probabilidad(X,minClass) arreglo con las probabilidades de ser positivas cada una
       de las instancias de X
8      probY ← Probabilidad(Y,minClass) arreglo con las probabilidades de ser positivas cada una
       de las instancias de Y
9      x ← Seleccion aleatoria con probabilidad(probX)
10     y ← Seleccion aleatoria con probabilidad(probY)
11     z ← inicializar instancia sintética
12     for a ← 1 to NumAttrs do
13         x_a ← valor del atributo a-ésimo de x
14         y_a ← valor del atributo a-ésimo de y
15         λ ← número aleatorio entre 0 y 1
16         z_a ← x_a + λ (y_a − x_a)
17     agregar z a Z
18 return Z
```

Algorithm 5: MISMOTE: Generación de la bolsa sintética

3.4. Experimentos

Los experimentos a realizar tendrán como objetivo demostrar cuál de las variantes de modificación de MISMOTE con los distintos modelos es la mejor y si alguna supera al MISMOTE original. Los experimentos se organizaron en un conjunto de pasos que se muestran en la figura (3.4.1). Primeramente se tomaron varios conjuntos de datos desbalanceados para luego aplicarles las cuatro variantes de MISMOTE, la variante original y las otras tres propuestas en esta tesis en la sección (3.2.1). Como resultado se obtienen cuatro variantes de preprocesamiento. En el paso siguiente se entrena un algoritmo de clasificación multinstancia con cada una de las cuatro variantes de preprocesamiento y se evalúa el clasificador con un conjunto de datos desbalanceado de prueba para conocer su desempeño. Este paso se repite con varios algoritmos de clasificación y los desempeños de los clasificadores resultantes se promedian para obtener un desempeño medio en cada variante de preprocesamiento. En la sección (3.4.2)se describen los algoritmos de clasificación empleados en la experimentación. El objetivo principal del experimento es determinar si existen diferencias significativas entre los desempeños medios de los clasificadores en las distintas variantes de preprocesamiento.

Para verificar las hipótesis planteadas se ejecutó primeramente el experimento en datos generados artificialmente que modelan la hipótesis estándar, luego se replicó el experimento con datos reales para

contrastar los resultados. En la sección (3.4.1) se describen los datos usados en la experimentación y la forma en que fueron generados los datos artificiales.

El experimento siguió un método riguroso de validación y se aplicaron pruebas estadísticas, tal como se describe en la sección .

Finalmente los resultados son presentados y discutidos en la secciónes (3.5) y (3.6).

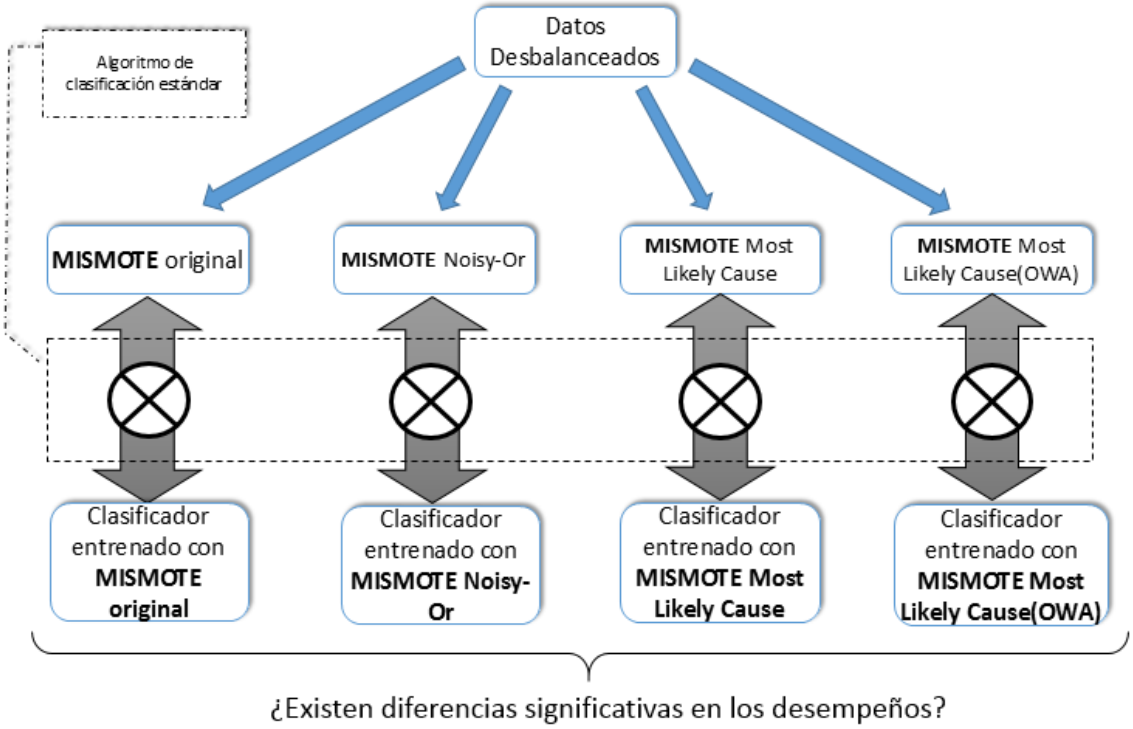

Figura 3.4.1: Diseño del experimento para determinar cuál es la mejor variante de MISMOTE

3.4.1. Conjuntos de Datos

Como se dijo anteriormente realizamos experimentos con datos generados artificialmente y con datos reales. Los datos reales fueron los mismos empleados en el capítulo 2 que se muestran en la tabla (2.4.1), a excepción del thioredoxin que no se pudo usar por su excesivo tiempo de computo. Los datos generados artificialmente se describen a continuación.

Es difícil garantizar que un conjunto de datos determinado cumple con rigor una hipótesis multinstancia. Como nos interesa conocer el comportamiento de los métodos que proponemos para mejorar la calidad de la clasificación en conjuntos de datos desbalanceados que cumplen efectivamente con la hipótesis multinstancia estándar fue necesario crear conjuntos de datos artificiales que garantizaran este requisito.

Es útil recordar que la hipótesis multinstancia estándar plantea que una bolsa es positiva si contiene al menos una instancia positiva y que una bolsa es negativa si todas sus instancias son negativas.

Para generar conjuntos de datos que satisfagan la hipótesis estándar, debemos tener la capacidad de generar instancias positivas a voluntad para poder insertarlas en bolsas positivas, así mismo debemos poder generar instancias negativas para insertarlas en las bolsas negativas que generemos. Para ello creamos un concepto de instancia positiva y otro concepto de instancia negativa. Cada concepto esta descrito por una distribución de probabilidades Gausiana con media μy desviación típica σ por cada dimensión del espacio de atributos. Para generar una instancia positiva se generan valores aleatorios para cada atributo siguiendo la distribución de probabilidades del concepto positivo. De igual manera para generar una instancia negativa se generan valores aleatorios para cada atributo siguiendo la distribución de probabilidades del concepto negativo. Cada bolsa negativa se generó con 10 instancias negativas mientras que cada bolsa positiva se generó con un número aleatorio entre 1 y 3 instancias positivas y el resto de instancias negativas hasta completar las 10 instancias por bolsa. El número de atributos por instancia fue entre 4 y 8.

Fueron generados conjuntos de datos artificiales con cuatro niveles de desbalance. Cada conjunto de datos está formado por 1000 bolsas. La distribución entre bolsas negativas y positivas se muestra en la tabla (3.4.1). El desbalance de las clases va desde un nivel pequeño de 3 hasta un desbalance muy alto de 99. Para cada nivel de desbalance se generaron 10 conjuntos de datos de entrenamiento y 10 de prueba.

Nivel de desbalance	Cantidad de bolsas positivas	Cantidad de bolsas negativas	IR
bajo	250	750	3,0
medio	100	900	9,0
alto	50	950	19,0
muy alto	10	990	99,0

Cuadro 3.4.1: Cantidad de bolsas de cada clase según el IR

3.4.2. Algoritmos de clasificación

En la Tabla 3.4.2 presentan los algoritmos usados en la experimentación, así como los parámetros con que fueron usados en el experimento. Se indica brevemente la justificación de su selección. En el algoritmo SimpleMI se usa C4.5 como clasificador base debido a que es un clasificador ampliamente reconocido y uno de los 10 algoritmos más importantes en la minería de datos.

Clasificador	Parámetros	Justificación de la selección
CitationKNN[44]	Citadores: 3 Referencias: 3	Encarna el enfoque de aprendizaje basado en instancias
MITI[6]		Encarna el enfoque de aprendizaje basado en árboles de decisión
SimpleMI	Clasificador base: C4.5	Convierte los datos multinstancia el monoinstancia para aplicar un clasificador base

Cuadro 3.4.2: Clasificadores multinstancia utilizados en las pruebas con las distintas variantes de MISMOTE

3.4.3. Método de validación y medidas de desempeño

La evaluación de la clasificación se llevo a cabo mediante un esquema de validación cruzada 5 CV en los conjuntos de datos reales. En los artificiales se usó hold out con 10 repeticiones. La medida de desempeño utilizada fue AUC, presentada en la Sección 1.3.1.

Se usó la prueba de rankings de Friedman para verificar la hipótesis nula de que no hay diferencias significativas entre el desempeño promedio de los clasificadores en cada una de las 6 variantes de MISMOTE. para determinar en cual de las variantes hay diferencias significativas se usó como post hoc la prueba de Shaffer. El procedimiento es similar al aplicado en el capítulo 2 y las descripciones de los métodos se encuentran en la sección (2.4.3).

3.5. Resultados con datos artificiales

A continuación se presentan los resultados de los experimentos con los conjuntos de datos artificiales diferenciando cada uno de los desniveles de clase, desde el nivel más bajo hasta el más alto. Las pruebas estarán enfocadas en demostrar cuál de las variantes de MISMOTE es la más adecuada para aplicar a datos desbalanceados.

Desbalance bajo (IR=3)

La tabla 3.5.1 muestra los valores promedios de la medida de desempeño AUC para los diez conjuntos de datos.

Datos	MISMOTE_ORIGINAL	MISMOTE_Noisi-Or	MISMOTE_Most Likely Cause	MISMOTE OWA
DA1	0,764496↓	0,792152889	0,802669333↑	0,791923556
DA2	0,821616↓	0,863810667	0,864288↑	0,864288↑
DA3	0,801577778↓	0,829549333↑	0,829549333↑	0,829549333↑
DA4	0,796472↓	0,824430222	0,830042667↑	0,824557333
DA5	0,774924444↓	0,811609778	0,821803556↑	0,797326222
DA6	0,796804444↓	0,851285333	0,867752889↑	0,837856
DA7	0,778662222	0,756972445↓	0,792033778↑	0,770901333
DA8	0,807652444↓	0,833659556	0,845677333↑	0,820452444
DA9	0,808153778↓	0,840809778↓	0,840825778↓	0,841295111↑
DA10	0,809059556↑	0,803376889	0,803328	0,796103111↓

Cuadro 3.5.1: Resultados de AUC promedio para los conjuntos de datos artificiales con IR=3

Como se puede apreciar la mayoria de los resultados más bajos se encuentran en la primera columna perteneciente al MISMOTE original. La figura 3.5.1 muestra los ranking al aplicar la prueba de Friedman. Se observa que la variante Most Likely Cause obtiene el mejor desempeño y que el MISMOTE original es superado por todas las variantes propuestas.

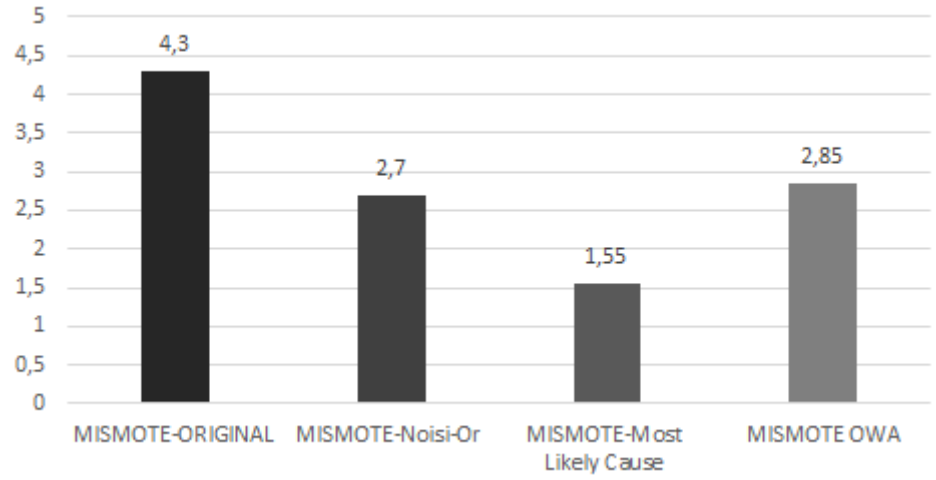

Figura 3.5.1: Prueba de Friedman para los resultados con los conjuntos de datos con IR=3

La prueba de Iman y Daveport devuelve una significación de $3{,}908149762033951 \times 10^{-4} < 0{,}05$, lo que indica que existen diferencias significativas entre al menos dos de las variantes comparadas. Entonces se pasa a determinar dónde están las diferencias a través de la tabla 3.5.2 con los resultados post hoc.

AUC

algorithms	p	Shaffer
MISMOTE-ORIGINAL vs. MISMOTE-Most Likely Cause	**0.000101**	**0.005**
MISMOTE-ORIGINAL vs. MISMOTE-Noisi-Or	0.023652	0.008333
MISMOTE-ORIGINAL vs. MISMOTE OWA	0.040305	0.008333
MISMOTE-Most Likely Cause vs. MISMOTE OWA	0.065992	0.008333
MISMOTE-Noisi-Or vs. MISMOTE-Most Likely Cause	0.103876	0.01
MISMOTE-Noisi-Or vs. MISMOTE OWA	0.832004	0.05

Cuadro 3.5.2: Post hoc con prueba de Shaffer para encontrar diferencias entre pares de variantes de MISMOTE en conjuntos de datos con IR=3

La prueba de Shaffer revela que existen diferencias significativas entre aplicar el MISMOTE-Most Likely Cause y aplicar el MISMOTE original, y del ranking de Friedman en la figura 3.5.1 se deduce que la variante de MISMOTE que implementa el modelo Most-Likely-Cause es significativamente mejor que el algoritmo original.

Desbalance medio (IR=9)

La tabla 3.5.3 muestra los valores promedios de la medida de desempeño AUC para los diez conjuntos de datos.

Datos	MISMOTE_ORIGINAL	MISMOTE_Noisi-Or	MISMOTE_Most Likely Cause	MISMOTE OWA
DA1	0,777218519↓	0,803842593↑	0,820192593↑	0,803842593↑
DA2	0,821868518↓	0,862590741↑	0,862590741↑	0,862590741↑
DA3	0,787837037↓	0,83855↑	0,83855↑	0,83855↑
DA4	0,777214815↓	0,809535185	0,813805555↑	0,809535185
DA5	0,812646296↓	0,824483333	0,846222222↑	0,828177778
DA6	0,787457408↓	0,857614815	0,859842593↑	0,856633333
DA7	0,79427963	0,758611111	0,812814815↑	0,75092963↓
DA8	0,810383333↓	0,862509259	0,84822037	0,867874074↑
DA9	0,784787037↓	0,824174074↑	0,824174074↑	0,824174074↑
DA10	0,812061111↑	0,779957407	0,797688889	0,773018519↓

Cuadro 3.5.3: Resultados de AUC promedio para los conjuntos de datos artificiales con IR=9

Observando estos resultados es también apreciable que los valores mas bajos se encuentran fundamentalmente en la primera columna. Al aplicarles la prueba de Friedman se obtienen los ranking representados en la figura 3.5.2. Igualmente se observa que la variante Most Likely Cause obtiene el mejor desempeño y que el MISMOTE original es superado por todas las variantes propuestas.

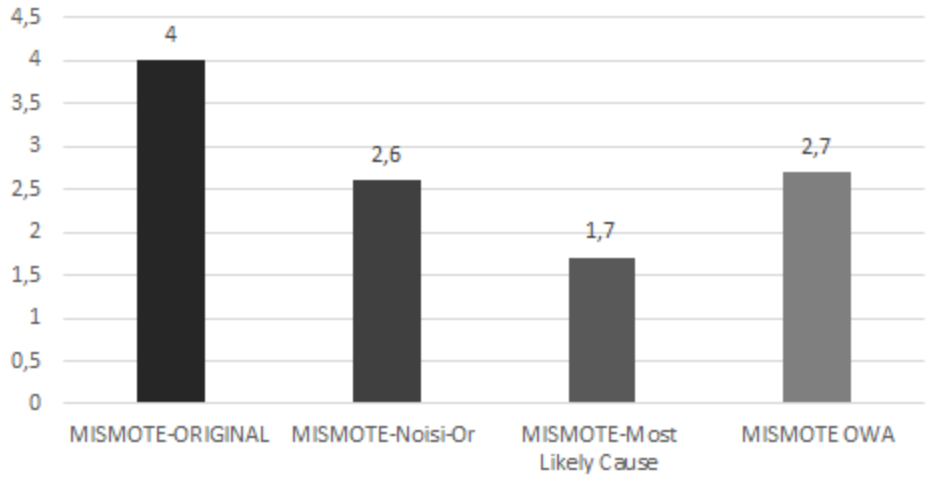

Figura 3.5.2: Prueba de Friedman para los resultados con los conjuntos de datos con IR=9

La prueba de Iman y Daveport devuelve una significación de $9{,}830217094917674 \times 10^{-4} < 0{,}05$ por lo que se dice que existen diferencias significativas y se pasa entonces a determinar donde están las diferencias a través de la tabla de Post hoc 3.5.4:

AUC

algorithms	p	Shaffer
MISMOTE-ORIGINAL vs. MISMOTE-Most Likely Cause	**0.001143**	**0.008333**
MISMOTE-ORIGINAL vs. MISMOTE-Noisi-Or	0.047715	0.008333
MISMOTE-ORIGINAL vs. MISMOTE OWA	0.065992	0.01
MISMOTE-Most Likely Cause vs. MISMOTE OWA	0.157299	0.0125
MISMOTE-Noisi-Or vs. MISMOTE-Most Likely Cause	0.203092	0.016667
MISMOTE-Noisi-Or vs. MISMOTE OWA	0.887537	0.025

Cuadro 3.5.4: Post hoc con prueba de Shaffer para encontrar diferencias entre pares de variantes de MISMOTE en conjuntos de datos con IR=9

En el caso de datos con una tasa de desbalance IR=9, igualmente como en los de IR=3, existen diferencias en la medida AUC entre aplicar el MISMOTE-Most Likely Cause y aplicar el MISMOTE original, por lo que la variante de MISMOTE que implementa el modelo Most-Likely-Cause mejora la calidad de la clasificación más que el algoritmo original.

Desbalance alto (IR=19)

La tabla 3.5.5 muestra los valores promedios de la medida de desempeño AUC para los diez conjuntos de datos.

Datos	MISMOTE_ORIGINAL	MISMOTE_Noisi-Or	MISMOTE_Most Likely Cause	MISMOTE OWA
DA1	0,752305263	0,787985965↑	0,748238596↓	0,787985965↑
DA2	0,817445614↓	0,891473684↑	0,890459649	0,891473684↑
DA3	0,813670175↓	0,855575438↑	0,855575438↑	0,855575438↑
DA4	0,746270176↓	0,755736842	0,756238597↑	0,755736842
DA5	0,794319298	0,783603509↓	0,798249123↑	0,785905263
DA6	0,805807017↓	0,837519298	0,909631579↑	0,837519298
DA7	0,798214035	0,769252632	0,804↑	0,73871579↓
DA8	0,785652632↓	0,849782456	0,854670175↑	0,850119298
DA9	0,812757895↓	0,846336842	0,846336842	0,847150877↑
DA10	0,804736842	0,80625614	0,835410526↑	0,793315789↓

Cuadro 3.5.5: Resultados de AUC promedio para los conjuntos de datos artificiales con IR=19

Aquí nuevamente los resultados más bajos se encuentran en el método MISMOTE original. Al aplicarles la prueba de Friedman se obtienen los rankings representados en la figura 3.5.3 . Igualmente se observa que la variante Most Likely Cause obtiene el mejor desempeño y que el MISMOTE original es superado por todas las variantes propuestas.

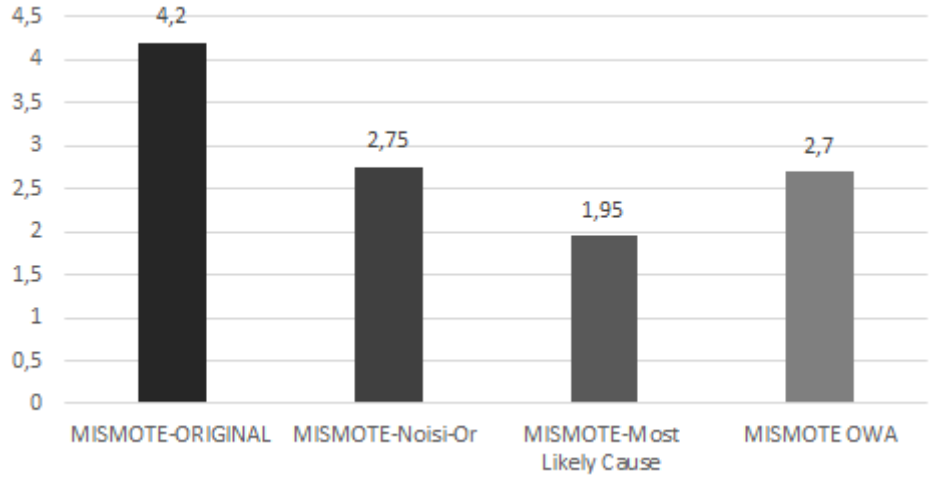

Figura 3.5.3: Prueba de Friedman para los resultados con los conjuntos de datos con IR=19

La prueba de Iman y Daveport devuelve una significación de 0,01445898574866181 < 0,05 por lo que se dice que existen diferencias significativas. A continuación se pasa a determinar donde están las diferencias a través de la tabla 3.5.6de post hoc.

AUC

algorithms	p	Shaffer
MISMOTE-ORIGINAL vs. MISMOTE-Most Likely Cause	**0.001463**	**0.005**
MISMOTE-ORIGINAL vs. MISMOTE OWA	0.033895	0.008333
MISMOTE-ORIGINAL vs. MISMOTE-Noisi-Or	0.040305	0.008333
MISMOTE-Noisi-Or vs. MISMOTE-Most Likely Cause	0.257899	0.01
MISMOTE-Most Likely Cause vs. MISMOTE OWA	0.288844	0.0125
MISMOTE-Noisi-Or vs. MISMOTE OWA	0.943628	0.05

Cuadro 3.5.6: Post hoc con prueba de Shaffer para encontrar diferencias entre pares de variantes de MISMOTE en conjuntos de datos con IR=19

En el caso de datos con una tasa de desbalance IR=19 también existen diferencias entre aplicar el MISMOTE-Most Likely Cause y aplicar el MISMOTE original. En este nivel de desbalance el MISMOTE-Most Likely Cause prueba ser mejor que el MISMOTE original para los datos del experimento.

Desbalance muy alto(IR=99)

La tabla 3.5.7 muestra los valores promedios de la medida de desempeño AUC para los diez conjuntos de datos.

Datos	MISMOTE_ORIGINAL	MISMOTE_Noisi-Or	MISMOTE_Most Likely Cause	MISMOTE OWA
DA1	0,755555556↑	0,752457912	0,731363636↓	0,752457912
DA2	0,816397306↓	0,862407407↑	0,862407407↑	0,862407407↑
DA3	0,756498317↓	0,814528619↑	0,814528619↑	0,814528619↑
DA4	0,787828283↓	0,802272727↑	0,802272727↑	0,802272727↑
DA5	0,87003367	0,881144781↑	0,810774411↓	0,872356902
DA6	0,786144781↑	0,756986532↓	0,777441077	0,756986532↓
DA7	0,764511785↑	0,737087542	0,732929293	0,643198653↓
DA8	0,845521886↓	0,910505051↑	0,910505051↑	0,906144781
DA9	0,799427609↓	0,900673401↑	0,900673401↑	0,900673401↑
DA10	0,81469697↑	0,766700337	0,762525253	0,723249158↓

Cuadro 3.5.7: Resultados de AUC promedios para los conjuntos de datos artificiales con IR=99

Ya con este desbalance los resultados más bajos aunque están también en gran medida en la primera columna , ya solo son 5 de los 10, quedando los restantes distribuidos fundamentalmente en la columna perteneciente al MISMOTE que implementa el modelo *Most Likely Cause(OWA)*.Al aplicarles la prueba de Friedman se obtienen los resultados expuestos en la figura 3.5.4 . En este nivel de desbalance se observa que los mejores resultados estan en la variente de MISMOTE que implementa el modelo *Noisy-Or*, mientras que la diferencia entre MISMOTE original y las variantes propuestas se ha reducido.

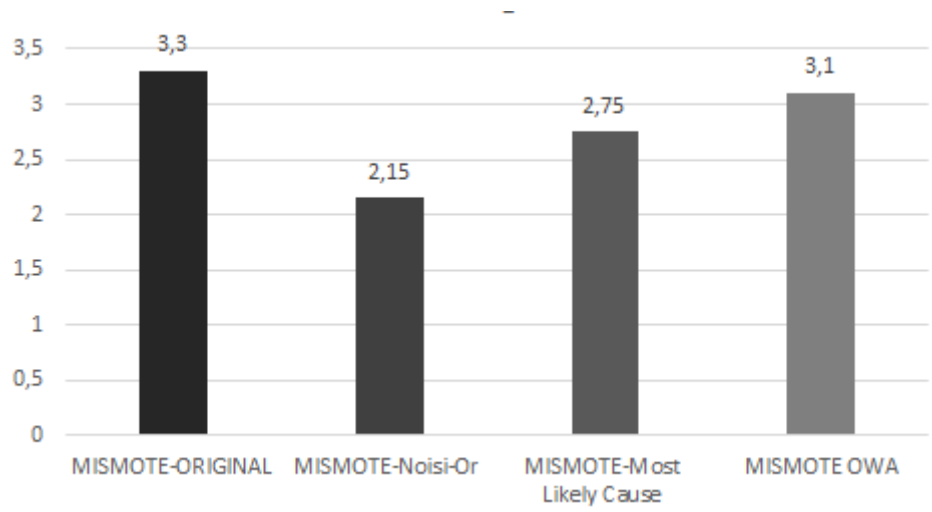

Figura 3.5.4: Prueba de Friedman para los resultados con los conjuntos de datos con IR=99

La prueba de Iman y Daveport devuelve una significación de $0{,}24244803933134454 > 0{,}05$ por lo que no existen diferencias significativas en aplicar cualquier variante de MISMOTE .

3.6. Resultados con datos reales

A continuación se realiza el experimento anterior sobre datos reales. En este caso solo se obtuvieron los resultados con el algoritmo de clasificación MITI, los cuales se presentan en la tabla 3.6.1.

DATOS	MISMOTE(Original)	MISMOTE(Noisy-Or)	MISMOTE(Most likely cause)	MISMOTE(OWA most likely cause)
WIRSEL1	0,776315789	0,669064327↓	0,727192982	0,804678363↑
WIRSEL2	0,769298246	0,722105263↑	0,649298246↓	0,830994152↑
WIRSEL3	0,807076023↑	0,734824561↓	0,781520468	0,806315789
WIRSEL4	0,687679739	0,727124183↑	0,671797386	0,647679739↓
WIRSEL5	0,637777778	0,612222222↓	0,697777778↑	0,688888889
WIRSEL6	0,62245098↓	0,630457516	0,678235294↑	0,642124183
Corel1	0,778947368↑	0,740526316	0,732368421↓	0,764736842
Corel2	0,696842105	0,712631579↑	0,705	0,686052632↓
Corel3	0,689473684↑	0,592368421	0,573421053↓	0,574210526
Corel4	0,880526316↑	0,780263158↓	0,783157895	0,781842105
Corel5	0,937894737	0,985263158↑	0,924736842↓	0,985263158↑
Corel6	0,692105263	0,653684211	0,730789474↑	0,635263158↓
Corel7	0,899473684↑	0,843421053↓	0,844210526	0,862631579
Corel8	0,892368421↓	0,909736842	0,915263158↑	0,915263158↑
Corel9	0,673157895	0,69↑	0,661578947	0,633684211↓
Corel10	0,82	0,804473684↓	0,832631579	0,834210526↑
Corel11	0,740789474↑	0,686842105↓	0,688684211	0,705263158
Corel12	0,625263158↓	0,635↑	0,632105263	0,626578947
Corel13	0,786052632	0,799210526	0,816052632↑	0,773684211↓
Corel14	0,815↑	0,787105263	0,802631579	0,772894737↓
Corel15	0,816842105↑	0,752894737	0,738421053↓	0,782368421
Corel16	0,709473684↑	0,614473684↓	0,691315789	0,638947368
Corel17	0,709210526	0,790526316	0,692631579↓	0,797368421↑
Corel18	0,818157895↑	0,735263158↓	0,735526316	0,777368421
Corel19	0,778684211↑	0,710526316	0,734210526	0,705526316↓
Corel20	0,598157895↑	0,556842105	0,560263158	0,55↓

Cuadro 3.6.1: Resultados de AUC con el clasificador MITI en los conjuntos de datos reales para las cuatro variantes de MISMOTE.

Al aplicar el test de Friedman para comparar los resultados anteriores los resultados devueltos son los reflejados en la figura 3.6.1. La figura revela un resultado completamente distinto a los obtenidos en los conjuntos de datos artificiales. Aquí el MISMOTE original es el que obtiene mejores resultados y las tres variantes propuestas quedan rezagadas con un desempeño similar.

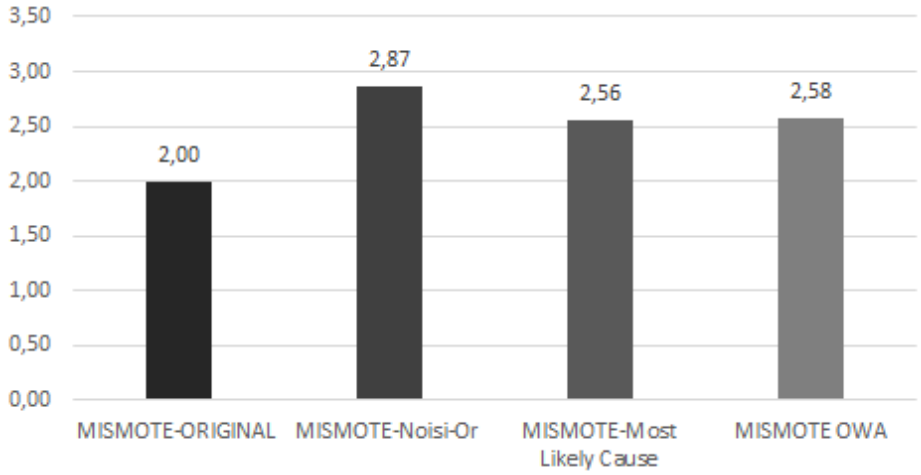

Figura 3.6.1: Prueba de Friedman para los resultados con los conjuntos de datos reales

La prueba de Iman y Daveport devuelve una significación de $0{,}10335759868657339 > 0{,}05$. Por tanto se rechaza la hipótesis nula, lo que indica que no existen diferencias significativas entre las cuatro variantes.

Este resultado contradictorio entre los datos reales y los datos generados artificialmente nos conduce a pensar que posiblemente los datos reales no se ajustan adecuadamente a la hipótesis estándar.

3.7. Conclusiones parciales

En conjuntos de datos que modelan la hipótesis estándar las variantes de MISMOTE que usan muestreo dirigido tienen mejor desempeño que la variante original de MISMOTE que usa muestreo aleatorio, lo cual sugiere la veracidad de la hipótesis propuesta: seleccionar instancias con más probabilidades de ser positivas en el proceso de interpolación de MISMOTE incrementa la probabilidad de obtener bolsas positivas. Otro resultado observado en los conjuntos de datos que modelan la hipótesis estándar es que a medida que aumenta el nivel de desbalance de clases disminuyen las diferencias en el desempeño de las distintas variantes de MISMOTE. Para menores niveles de desbalance los mejores resultados son obtenidos por el modelo Most Likely Cause. A medida que aumenta el desbalance el modelo Most Likely Cause disminuye su efectividad y aumenta el desempeño del modelo Noisy-Or.

Los resultados observados en los datos reales son distintos a los observados en los datos artificiales que modelan la hipótesis estándar. El MISMOTE original obtiene mejores resultados que los modelos que usan muestreo dirigido, lo cual probablemente esté relacionado con el hecho de que los datos reales no necesariamente se ajustan a la hipótesis estándar. El ajuste de la hipótesis estándar a un conjunto de datos real es si acaso muy difícil de probar y está fuera del alcance de esta tesis.

Conclusiones

Concretando, con la realización de este proyecto de diplome se arribó a las siguientes conclusiones:

- El submuestreo aleatorio de la clase mayoritaria mejora la clasificación de conjuntos de datos desbalanceados en algunos dominios de aplicación.
- De forma general no existe un balance óptimo entre sobremuestreo de la clase minoritaria y submuestreo de la clase mayoritaria en la clasificación multinstancia, pero para determinadas aplicaciones puede existir una combinación que sea la más eficaz.
- Las pruebas con datos artificiales que cumplen rigurosamente con la hipótesis estándar muestran que las variantes de MISMOTE que muestrean con distribución proporcional a la probabilidad de cada instancia de ser positiva son mejores que la variante original de MISMOTE con muestreo con probabilidad uniforme. Pero los datos reales no necesariamente cumplen de forma rigurosa con la hipótesis estándar.
- Como no es posible determinar el ajuste de la hipótesis estándar en un conjunto de datos, no es conveniente la mejora de MISMOTE con los modelos propuestos que basan su funcionamiento en la hipótesis estándar.

Recomendaciones

- Hacer un estudio profundo con conjuntos de datos multinstancia reales para determinar que características pueden relacionarse con la hipótesis multinstancia y que permita obtener mejoras en la clasificación al aplicar las variantes de MISMOTE desarrolladas en esta tesis.

Bibliografía

[1] R.A. Amar, D.R. Dooly, S.A. Goldman, and Q. Zhang. Multiple-instance learning of real-valued data. In *Machine Learning-International Workshop*, pages 3–10, 2001.

[2] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 561–568, 2002.

[3] Boris Babenko, Piotr Dollár, Zhuowen Tu, and Serge Belongie. Simultaneous learning and alignment: Multi-instance and multi-pose learning. In *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition*, 2008.

[4] M. Basu and T.K. Ho, editors. *Data Complexity in Pattern Recognition*. Advanced Information and Knowledge Processing. Springer, 2006.

[5] L. Bjerring and E. Frank. Beyond trees: Adopting MITI to learn rules and ensemble classifiers for multi-instance data. In *Proceedings of the 24th international conference on Advances in Artificial Intelligence*, pages 41–50. Springer, 2011.

[6] H. Blockeel, D. Page, and A. Srinivasan. Multi-instance tree learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 57–64, 2005.

[7] J. Bolton and P. Gader. Application of multiple-instance learning for hyperspectral image analysis. *IEEE Geoscience and Remote Sensing Letters*, 8(5):889–893, 2011.

[8] J. Bolton, P. Gader, H. Frigui, and P. Torrione. Random set framework for multiple instance learning. *Information Sciences*, 181(11):2061–2070, 2011.

[9] Y. Chen and J.Z. Wang. Image categorization by learning and reasoning with regions. *The Journal of Machine Learning Research*, 5:913–939, 2004.

[10] S. Das, B. Amoedo, F. De la Torre, and J. Hodgins. Detecting parkinsons' symptoms in uncontrolled home environments: a multiple instance learning approach. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, volume 2012, pages 3688–3691. IEEE, 2012.

[11] T. G. Dietterich, R. H. Lathrop, and T. LozanoPerez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[12] J. Ding, H. Cheng, J. Huang, J. Liu, and Y. Zhang. Breast ultrasound image classification based on multiple-instance learning. *Journal of Digital Imaging*, 25(5):620–627, 2012.

[13] S. Feng, W. Xiong, B. Li, C. Lang, and X. Huang. Hierarchical sparse representation based multi-instance semi-supervised learning with application to image categorization. *Signal Processing*, 94:595–607, 2014.

[14] G. Fu, X. Nan, H. Liu, R. Y. Patel, P. R. Daga, Y. Chen, D. E. Wilkins, and R. J. Doerksen. Implementation of multiple-instance learning in drug activity prediction. *BMC Bioinformatics*, 13(15):1–12, 2012.

[15] S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064, 2010.

[16] S. García and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[17] V. García, R. A. Mollineda, and J. S. Sánchez. On the k-NN performance in a challenging scenario of imbalance and overlapping. *Pattern Analysis and Applications*, 11(3-4):269–280, 2008.

[18] I. Gondra and T. Xu. A multiple instance learning based framework for semantic image segmentation. *Multimedia Tools and Applications*, 48(2):339–365, 2010.

[19] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[20] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449, 2002.

[21] A. Katsamanis and J. Gibson. Multiple instance learning for classification of human behavior observations. In *Proceedings of the 4th international conference on Affective computing and intelligent interaction-Volume Part I*, pages 145–154. Springer-Verlag, 2011.

[22] H. Kriegel, A. Pryakhin, and M. Schubert. An EM-approach for clustering multi-instance objects. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 139–148. Springer, 2006.

[23] C. Lee, A. Katsamanis, M. P. Black, B. R. Baucom, P. G. Georgiou, and S. S. Narayanan. Affective state recognition in married couples ' interactions using PCA-based vocal entrainment measures with multiple instance learning. In *Proceedings of the 4th international conference on Affective computing and intelligent interaction-Volume Part II*, pages 31–41. Springer-Verlag, 2011.

[24] B. Li, W. Xiong, and W. Hu. Web horror image recognition based on context-aware multi-instance learning. In *IEEE 11th International Conference on Data Mining*, pages 1158–1163. IEEE, 2011.

[25] D. X. Li, J. L. Fan, D. W. Wang, and Y. Liu. Latent topic based multi-instance learning method for localized content-based image retrieval. *Computers and Mathematics with Applications*, 64(4):500–510, 2012.

[26] H. Liu, C. Zhou, J. Shen, P. Li, and S. Zhang. Video caption detection algorithm based on multiple instance learning. In *2010 Fifth International Conference on Internet Computing for Science and Engineering*, pages 20–24. IEEE, 2010.

[27] Oded Maron and Aparna Lakshmi Ratan. Multiple-instance learning for natural scene classification. In *Proceedings of the 15th International Conference on Machine Learning*, pages 341–349, 1998.

[28] F. U. Minhas and A. Ben-Hur. Multiple instance learning of calmodulin binding sites. *Bioinformatics*, 28(18):i416–i422, 2012.

[29] T. M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, New York, NY, USA., 1997.

[30] J. Ning, W. Shi, S. Yang, and P. Yanne. Visual tracking based on distribution fields and online weighted multiple instance learning. *Image and Vision Computing*, 31(11):853–863, 2013.

[31] M. Popescu and A. Mahnot. Early illness recognition using in-home monitoring sensors and multiple instance learning. *Methods of information in medicine*, 51(4):359–367, 2012.

[32] R. C. Prati, G. E. Batista, and M. C. Monard. Class imbalances versus class overlapping: an analysis of a learning system behavior. In *MICAI 2004: Advances in Artificial Intelligence*, pages 312–321. Springer, 2004.

[33] Q. Qiao and P. Beling. Classroom video assessment and retrieval via multiple instance learning. In *Proceedings of the 15th international conference on Artificial Intelligence in Education*, pages 272–279. Springer, 2011.

[34] G. Quellec, M. Lamard, M. D. Abràmoff, E. Decencière, B. Lay, A. Erginay, B. Cochener, and G. Cazuguel. A multiple-instance learning framework for diabetic retinopathy screening. *Medical Image Analysis*, 16(6):1228–1240, 2012.

[35] S. Ray and D. Page. Multiple instance regression. In *Machine Learning-International Workshop*, pages 425–432, 2001.

[36] Dánel Sánchez Tarragó. *Algoritmos para la Clasificación Multinstancia*. Doctoral thesis, Universidad de Granada, Granada, España, 2014.

[37] P. Simard, Y. LeCun, and J. S. Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems 5 (NIPS Conference)*, pages 50–58, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[38] P. Simard, Y. Lecun, B. Victorri, and J. Denker. Tangent prop - a formalism for specifying selected invariances in an adaptive network. In *Advances in Neural Information Processing Systems 4 (NIPS Conference)*, volume 4, pages 895–903, 1992. Bw71y Times Cited:29 Cited References Count:0 Advances in Neural Information Processing Systems.

[39] X. Song, L. C. Jiao, S. Yang, X. Zhang, and F. Shang. Sparse coding and classifier ensemble based multi-instance learning for image categorization. *Signal Processing*, 93(1):1–11, 2013.

[40] L. Sun, Y. Lu, K. Yang, and S. Li. ECG analysis using multiple instance learning for myocardial infarction detection. *IEEE transactions on biomedical engineering*, 59(12):3348–3356, 2012.

[41] D.M.J. Tax, E. Hendriks, M.F. Valstar, and M. Pantic. The detection of concept frames using clustering multi-instance learning. In *20th International Conference on Pattern Recognition (ICPR)*, pages 2917–2920, 2010.

[42] R. Teramoto and H. Kashima. Prediction of protein-ligand binding affinities using multiple instance learning. *Journal of Molecular Graphics and Modelling*, 29(3):492–497, 2010.

[43] K. Wagstaff and T. Lane. Salience assignment for multiple-instance regression. In *Proceedings of the International Conference on Machine Learning 2007 Workshop on Constrained Optimization and Structured Output Spaces.*, 2007.

[44] J. Wang and J-D. Zucker. Solving the multiple-instance problem: A lazy learning approach. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 1119–1126, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[45] S. Wang, M. T. McKenna, T. B. Nguyen, J. E. Burns, N. Petrick, B. Sahiner, and R. M. Summers. Seeing is believing: Video classification for computed tomographic colonography using multiple-instance learning. *IEEE transactions on medical imaging*, 31(5):1141–1153, 2012.

[46] X. Wang, S. Matwin, N. Japkowicz, and X. Liu. Cost-sensitive boosting algorithms for imbalanced multi-instance datasets. In *Advances in Artificial Intelligence*, pages 174–186. Springer, 2013.

[47] Z. Wang, V. Radosavljevic, B. Han, and Z. Obradovic. Aerosol optical depth prediction from satellite observations by multiple instance regression. In *Proceedings of the SIAM International Conference on Data Mining*, pages 165–176. SIAM, 2008.

[48] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, and others. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[49] X. Xu and E. Frank. Logistic regression and boosting for labeled bags of instances. In *Proc. of the PacificAsia Conf. on Knowledge Discovery and Data Mining*, pages 272–281. Springer-Verlag, 2004.

[50] W. Yang, Y. Gao, and L. Cao. TRASMIL: A local anomaly detection framework based on trajectory segmentation and multi-instance learning. *Computer Vision and Image Understanding*, 117(10):1273–1286, 2013.

[51] K. Zhang and H. Song. Real-time visual tracking via online weighted multiple instance learning. *Pattern Recognition*, 46(1):397–411, 2013.

[52] M.L. Zhang and Z.H. Zhou. Multi-instance clustering with applications to multi-instance prediction. *Applied Intelligence*, 31(1):47–68, 2009.

[53] T. Zhang, S. Liu, C. Xu, and H. Lu. M4l: Maximum margin multi-instance multi-cluster learning for scene modeling. *Pattern Recognition*, 46(10):2711–2723, 2013.

[54] Z.H. Zhou and M.L. Zhang. Multi-instance multi-label learning with application to scene classification. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems*, pages 1609–1616. MIT Press, 2006.

[55] S. Zhu and X. Tan. A novel automatic image annotation method based on multi-instance learning. *Procedia Engineering*, 15:3439–3444, 2011.