Konputazio Ingeniaritza eta
Sistema Adimentsuak Unibertsitate Masterra

Máster Universitario en Ingeniería Computacional
y Sistemas Inteligentes

Konputazio Zientziak eta Adimen Artifiziala Saila

Departamento de Ciencias de la Computación e Inteligencia Artificial

Master Tesia

Tesis de Máster

Mejora de la efectividad de la clasificación en la plataforma WEKA en base al uso de métodos de remuestreo sobre la distribución de clases óptima

Asier Arostegui Carrasco

Zuzendaritza
Dirección

Jesús María Pérez de la Fuente

Konputagailuen Arkitektura eta Teknologia

Arquitectura y Tecnología de Computadores

Universidad del País Vasco Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Konputazio Ingeniaritza eta Sistema Adimentsuak Unibertsitate Masterra

Máster Universitario en Ingeniería Computacional y Sistemas Inteligentes

Resumen

Los problemas de clasificación de bases de datos desbalanceadas constituyen un paradigma específico dentro del Machine Learning que ha sido ampliamente estudiado en los últimos años y que puede ser abordado desde distintos puntos de vista. Hay distintos tipos de aproximaciones a este problema: las más comúnmente encontradas en literatura son aquellas de tipo algorítmico, las de selección de atributos (features) y las de tratamiento de datos o remuestreo. La ventaja de las aproximaciones de tratamiento de datos es que son versátiles y pueden aplicarse a cualquier tipo de algoritmo y se basan en la aplicación de un remuestreo de las instancias disponibles para tratar de balancear la clase minoritaria, con una aproximación clásica de rebalanceo al 50%.

Este trabajo plantea un doble objetivo: por un lado, extender análisis empíricos previos que muestran que la distribución óptima no tiene por qué ser la del 50% que plantearon las primeras hipótesis y que dependerá del tipo de problema o conjunto de datos a estudiar, del tipo de clasificador elegido, del tipo de algoritmo de remuestreo aplicado y de la métrica que se defina para la evaluación del clasificador, lo que se refiere típicamente como dependencia del contexto. Del análisis de resultados, se comprobará si algunos valores concretos de las variables de contexto obtienen un mejor desempeño que otros sobre el universo de análisis considerado.

Por otro lado, se pretende poner a disposición de la comunidad un módulo sobre la plataforma WEKA que permita, no sólo automatizar el procedimiento para replicar el estudio, sino poder aplicar esta implementación de manera general para encontrar una distribución óptima basada en el uso de distintas técnicas de remuestreo adecuadas para un contexto determinado, es decir, para un problema de clasificación, un método de remuestreo, un algoritmo de clasificación y un criterio de bondad concreto que pueda definir el usuario.

Palabras clave: *Clasificadores – WEKA – Distribución óptima – Problemas de clase desbalanceada – Métodos de remuestreo – Dependencia de contexto*

Aurreak erakusten du atzea nola dantzatu

Agradecimientos

En primer lugar, me gustaría agradecer a mi director, Txus, por haberme ayudado, especialmente en el arranque del viaje, pero también a lo largo de todo el desarrollo del trabajo y por su disponibilidad ante las dudas y obstáculos que han ido surgiendo en todas las etapas del proyecto.

Por otro lado, a mi familia por permitirme evadirme, aun estando en la misma casa, para hacer aquello que me gusta, especialmente durante un periodo en el que el mundo se ha hecho más pequeño y las ausencias se han notado más que nunca.

Índice

Lista de Figuras

Lista de Tablas

1 Introducción

1.1 El problema de la Clasificación

El problema de la clasificación, también referida en literatura como Reconocimiento de Patrones, es uno de los paradigmas del Machine Learning, que puede definirse como un problema que parte de un conjunto de *n* registros $w_1,w_2,\ldots,w_n$ cada uno de los cuales queda caracterizado por *i* atributos $a_1,a_2,\ldots,a_i$ que pueden ser de tipo numérico, booleano o nominal y *m* clases[1] $c_1,c_2,\ldots,c_m$ y cuyo objetivo es la creación de un modelo que automáticamente asigna a cada uno de los registros w_n su correspondiente clase c_n.

Matemáticamente, esto puede expresarse como

$$\mathbb{A}^i \rightarrow \{c_1, c_2, \ldots, c_m\}$$

En el concepto del Machine Learning, el aprendizaje (Learning) se refiere al procedimiento de generación del clasificador, en el cual se analizan los atributos de cada muestra disponible en un proceso que se denomina "entrenamiento", de cara a crear un modelo preciso que sea capaz de definir la clase propia de un nuevo registro, del que no conocemos su clase a priori, en base a sus atributos. El "Aprendizaje Supervisado" (Supervised Learning) se aplica cuando el conjunto de registros de entrenamiento dispone de, además de sus atributos correspondientes, una etiqueta de clase verificada previamente que será utilizada por el algoritmo de aprendizaje para construir el modelo referido.

Para generar estos modelos de aprendizaje se han desarrollado varias aproximaciones [1] como, entre otras, las basadas en árboles de decisión [2], K-Nearest Neighbor (kNN) [3], Support Vector Machine (SVM) [4] o clasificadores de análisis de probabilidad, como el Naive Bayes [5]

1.2 Aspectos específicos de las distribuciones desbalanceadas

El problema de los conjuntos de datos desbalanceados [6], [7], [8] aparece cuando una de las clases, usualmente aquella que es clave para el objetivo de la clasificación y a la que se la denomina *clase positiva*, tiene muy pocas ocurrencias en el conjunto de datos. Gestionar el problema del desbalanceo de datos es uno de los principales problemas que afronta la minería de datos con un impacto notable en multitud de casos prácticos [9] y, más allá del paradigma de la diagnosis médica, en el que típicamente los diagnosticados de ciertas enfermedades son minoría sobre el total del universo analizado, se extiende en multitud de campos, como los de detección

[1] Nótese que la clase a predecir en esta definición es discreta, lo que separa a los problemas de clasificación, a los que nos vamos a referir en este trabajo, de los problemas de regresión, en los que la clase se sustituye por una variable de salida continua, que constituiría la variable a predecir por el modelo.

de fraude, clasificación de textos, reconocimiento de caras o localización de problemas de calidad en redes de telecomunicaciones.

Bajo este paradigma, hay muchas ocasiones en las que ya el problema original se caracteriza por un problema biclásico, es decir, el problema genérico de *m* clases $c_1,c_2,\ldots,c_m$ se concreta en un problema con $m=2$, como el ya comentado ejemplo de la diagnosis de una enfermedad específica, donde hay dos únicos valores posibles de clase que típicamente se referirán como negativo (sano, como clase dominante) y positivo (enfermo, como clase clave cuya ocurrencia es rara). Por otro lado, partiendo de un problema general multiclásico, se puede llegar a uno (o varios) biclásicos al distinguir una clase de especial interés (que será la *clase positiva*) frente a la agregación de todas las demás que constituirán la *clase negativa*. Esto sucede cuando, de entre todas las clases inicialmente especificadas, hay una clase clave que centra el interés de una determinada estrategia y, en el ejercicio de esta reducción del número original de clases *m* a 2, puede emerger un problema de conjunto de datos desbalanceado.

Por otro lado, esta técnica de la reducción al problema biclásico también es usada habitualmente para transformar un problema multiclásico desbalanceado con varias clases minoritarias en varios problemas biclásicos desbalanceados y aplicar técnicas específicas de los problemas biclásicos desbalanceados, aunque como alternativa se puede encontrar trabajos sobre algoritmos específicos para conjuntos de datos desbalanceados multiclase [10].

La clasificación de conjuntos de datos desbalanceados presenta dos problemas respecto a la aproximación general usada en los problemas de clasificación:

Por un lado, las métricas más habitualmente usadas en problemas de clasificación balanceada, asociada al desempeño del "porcentaje de acierto del clasificador" no funcionan correctamente en un problema desbalanceado con una clase clave, según el planteamiento presentado para este tipo de problemas. Extenderemos las consideraciones sobre las métricas de evaluación en la clasificación desbalanceada a continuación.

Por otro lado, podemos encontrar trabajos que muestran que los métodos clasificadores generales ofrecen un pobre resultado, cuando se enfrentan a un conjunto de datos desbalanceado [11].

Hay distintas aproximaciones que engloban las distintas técnicas usadas para enfrentarse a la clasificación de conjuntos de datos desbalanceados que suelen referirse como técnicas de remuestreo o de tratamiento de datos, técnicas basadas en algoritmos específicos y técnicas basadas en selección de características (feature selection). Exploraremos los detalles de las variantes y evolución de estas técnicas en el apartado de Estado del Arte.

1.3 Criterios de evaluación de clasificadores

A la hora de trabajar con los clasificadores, un aspecto fundamental es la definición de un criterio que permita comparar el desempeño de cada uno para establecer cuál de ellos es mejor que otro al enfrentarse a un determinado problema. Para ello, es necesario definir unas métricas apropiadas

La mayor parte de las métricas que se utilizan para la evaluación de clasificadores biclásicos se definen a partir de los elementos de la matriz de confusión, que se ilustra en la **Tabla 1**.

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Tabla 1: *Matriz de Confusion*

La matriz de confusión maneja los siguientes conceptos básicos a partir de la combinación de los casos predichos por el clasificador al enfrentarlos a un conjunto de datos de test, como positivos o negativos, y enfrentarlos con su verdadera caracterización, como positivos o negativos. De esta combinación surgen cuatro posibilidades:

- **VP** es la cantidad de *positivos* que fueron *clasificados correctamente* como positivos por el modelo.
- **VN** es la cantidad de *negativos* que fueron *clasificados correctamente* como negativos por el modelo.
- **FN** es la cantidad de *positivos* que fueron *clasificados incorrectamente* como negativos.
- **FP** es la cantidad de *negativos* que fueron *clasificados incorrectamente* como positivos.

A partir de estos conceptos se definen las métricas más básicas y habituales, recogidas en la **Tabla 2**, donde se indican los indicadores más habituales con sus acepciones en español e inglés, cuyos términos están más estandarizados, así como las fórmulas basadas en los términos de la matriz de confusión previamente definidos y la descripción de las propias métricas.

Metrica	Formula	Descripción
Exactitud *(Accuracy)*	$\frac{VP + VN}{VP + FP + VN + FN}$	Relación entre las predicciones correctas sobre el total de instancias
Tasa de error *(Error Rate)*	$\frac{FP + FN}{VP + FP + VN + FN}$	Relación entre las predicciones erróneas sobre el total de instancias
Sensibilidad *(Sensitivity)*	$\frac{VP}{VP + FN}$	Fracción de positivos reales que son correctamente predichos
Especificidad *(Specificity)*	$\frac{VN}{VN + FP}$	Fracción de negativos reales que son correctamente predichos
Precision *(Precision)*	$\frac{VP}{VP + FP}$	Relación entre los positivos correctamente predichos entre todos los casos predichos como positivos
Exhaustividad *(Recall)*	$\frac{VP}{VP + VN}$	Relación entre los positivos correctamente predichos entre todos los casos correctamente predichos

Tabla 2: *Métricas de evaluación básica basadas en la matriz de confusión*

Uno de los aspectos a tener en cuenta en los procedimientos aplicados a los conjuntos de datos desbalanceados es el de la evaluación del modelo.

Cuando aplicamos las distintas métricas de evaluación a los clasificadores, debemos tener en cuenta que la mayoría de los algoritmos de aprendizaje obtienen una precisión elevada sobre la clase mayoritaria, pero entregan una predicción pobre de la clase minoritaria, que es la clase clave del análisis [12]. Como típico ejemplo extremo de este caso, el modelo podría tratar la clase minoritaria como ruido e ignorarla totalmente para predecir siempre la clase mayoritaria, obteniendo un modelo sencillo con una alta exactitud (*accuracy*), que suele ser una métrica muy común para evaluar un clasificador multiclásico balanceado, pero una baja precisión (*precision*) o exhaustividad (*recall*).

Se puede observar, por tanto, que los algoritmos clásicos tienden típicamente a generar una menor precisión o exhaustividad sobre la clase minoritaria, que es típicamente nuestra clase clave, que sobre la clase mayoritaria. Del mismo modo, tenemos que tener en cuenta que determinadas medidas (precisión o sensibilidad) pueden ser confiables, además de intuitivas, para evaluar la clasificación de datos balanceados, pero pierden eficiencia al enfrentarse a conjuntos de datos desbalanceados.

De esta manera, como ya se ha mencionado, una vez seleccionada una métrica adecuada para un problema desbalanceado, que detecte y pese específicamente los fallos de predicción del algoritmo en base a las consecuencias de tales fallos sobre el problema concreto, se puede comprobar que los métodos clasificadores habituales no alcanzan valores satisfactorios para estas métricas, ya adaptadas para un conjunto de datos desbalanceado [11].

Esto es debido a la posible falta de datos caracterizadores de la clase minoritaria en el conjunto de entrenamiento, al enfrentarnos a una distribución de datos muy desbalanceada, de modo que los datos de la clase minoritaria son tan escasos que los patrones caracterizadores no emergen en

el clasificador y de esta manera la frontera de decisión que establece el clasificador puede estar muy lejos de la verdadera frontera de decisión del problema [13].

Otra dificultad proviene de los problemas relacionados con la complejidad o el solapamiento referidos a la "fronteras" de discriminación entre las clases [14], que son más difíciles de establecer en una distribución desbalanceada porque muestras "ruidosas" de la clase mayoritaria pueden tener un peso similar a las ocurrencias de la clase minoritaria.

Por ello, los procedimientos de evaluación a utilizar al enfrentarse a un problema de clasificación de un conjunto de datos desbalanceados deben basarse en unas métricas adecuadas que tengan en cuenta estos problemas y así, dependiendo del problema concreto y de las consecuencias de los desajustes de predicción (falsos positivos, falsos negativos, …) el usuario podrá dar preferencia a una métrica u otra, pero siempre teniendo en cuenta las posibilidades que son adecuadas para este tipo de problemas.

A la hora de elegir una métrica adecuada, se tendrán en cuenta los siguientes aspectos:

- **Complejidad y coste computacional el cálculo**. De hecho, una de las razones del éxito de las métricas básicas presentadas es precisamente su facilidad de cálculo.
- **Poder de discriminación**. Algunas métricas muestran unos valores muy parecidos bajo ciertas condiciones, aunque el comportamiento de los clasificadores evaluados sea distinto.
- **Comportamiento en conjuntos de datos desbalanceados**. Como el objetivo es trabajar con conjuntos de datos desbalanceados, esta característica será importante. Una buena métrica deberá ser capaz de realizar una buena evaluación sobre la clase minoritaria, que suele ser la clase clave bajo estudio, evitando que el peso del número de instancias de la clase mayoritaria oculte un mar comportamiento de la clasificación de la clase clave, como en el ejemplo mencionado para la exactitud.

Dentro del análisis del Estado del Arte, revisaremos cuales son las métricas alternativas más habituales para la evaluación de clasificadores aplicados a conjuntos de datos desbalanceados a partir de los distintos trabajos publicados.

1.4 WEKA, un entorno de trabajo para el Machine Learning

Para realizar las experimentaciones sobre los aspectos referidos hasta el momento, nos apoyaremos en una herramienta que nos permita trabajar de manera cómoda con las implementaciones de los distintos algoritmos de clasificación y las métricas asociadas a los mismos. Para ello, utilizaremos WEKA.

WEKA es un entorno de trabajo que incluye un set de herramientas que facilita la implementación de distintas técnicas de Machine Learning de modo que puedan ser aplicadas a distintos problemas concretos. Dentro de este entorno de trabajo, WEKA pone a disposición del usuario, además de la implementación de los algoritmos de Machine Learning, otras utilidades, como herramientas de manipulación y procesado de datos, visualización de resultados y de

esquemas en función de las técnicas seleccionadas, definición de reglas de validación y comparación de resultados, aplicación de métricas de evaluación, etc.

WEKA[2] es el acrónimo de *Waikato Environment for Knowledge Analysis* (Entorno para el análisis de conocimiento Waikato), desarrollado desde el departamento de Ciencias de la Computación de la universidad de Waikato en Nueva Zelanda. El proyecto WEKA comienza a andar en 1993, desarrollándose inicialmente en TCL y C y en 1997 se da un hito fundamental al redefinirse sobre Java, que es el lenguaje sobre el que continúa desarrollándose actualmente.

WEKA es reconocido como un punto de referencia por la comunidad del Data Mining y del Machine Learning, tanto en ámbitos de tipo académico como aplicados al mundo empresarial y varios artículos recogen sus características principales y sus aplicaciones a lo largo de los años a campos concretos [15], [16], [17], [18]. Una de las razones por las que WEKA ha tenido tanto éxito dentro de la comunidad del Machine Learning es que está licenciado como GNU (*General Public License*) [19] de modo que cualquier usuario puede modificar el código del mismo o extenderlo con nuevos módulos.

Este reconocimiento que disfruta WEKA en la comunidad y sus características de GNU han propiciado un desarrollo amplio de la herramienta, tanto en la disponibilidad de módulos como en una abundante literatura que explora y evalúa los módulos implementados en WEKA, analizando los resultados aplicados a sus campos de estudio [20], [21], [22].

Del mismo modo, WEKA participa en los análisis comparativos que realiza la comunidad sobre las distintas herramientas de minería de datos disponibles [22].

Inicialmente, WEKA ofrece cinco grandes aplicaciones, además de otras opciones en la barra de menú superior que facilitan la gestión del propio programa y sus recursos, la ayuda del mismo y la visualización y gestión de las fuentes de datos en sus distintas posibilidades de formatos de entrada.

La ventaja que aporta la arquitectura de WEKA es que, una vez que un módulo para la implementación de un algoritmo es desarrollado, éste queda disponible para todas las aplicaciones de WEKA, facilitando la integración y la utilización del mismo en distintas tareas. Esta es una característica fundamental que utilizaremos como palanca para sacar partido a la implementación en la que se basará nuestra propuesta de investigación.

[2] Weka es también un ave nativa de Nueva Zelanda, del tamaño de un pollo, que destaca por una actitud de curiosidad frente a su entorno.

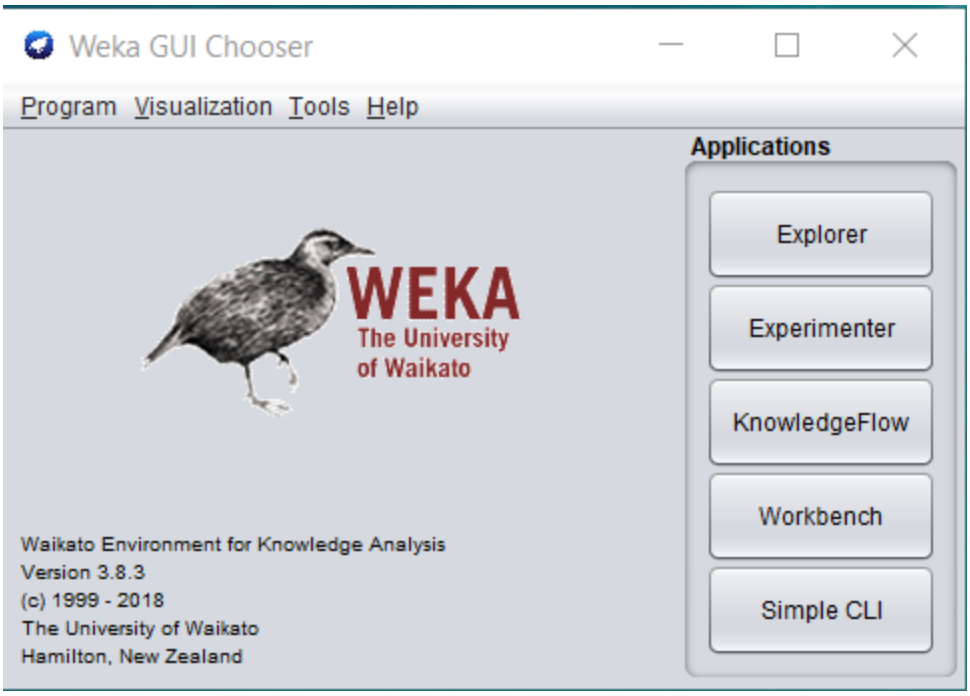

***Fig. 1**: Menú de inicio con las aplicaciones de la plataforma WEKA*

Las cinco grandes aplicaciones que ofrece WEKA, que pueden visualizarse en el menú inicial de la herramienta que presenta la **Fig.1**, son:

- Explorer

Es la aplicación básica junto con el CLI (*Simple CLI*) y la más comúnmente utilizada por los usuarios, sobre todo en una aproximación inicial. Permite operar sobre una única fuente de datos. La aplicación presenta una interfaz gráfica que pone a disposición del usuario toda la colección de algoritmos de análisis implementados por WEKA con sus valores por defecto y sus ventanas desplegables para permitir reconfiguraciones ad-hoc de los algoritmos.

Dentro de esta aplicación se incluyen distintas tareas, cada una asociada a una pestaña del interfaz gráfico: Preprocesado de los datos y aplicación de filtros (*Preprocess*), clasificación (*Classify*), *Clustering (Cluster)*, búsqueda de asociaciones (*Associate*), selección de atributos (*Select Attributes*) y visualización de datos (*Visualize*).

- Experimenter

Esta aplicación está especialmente diseñada para aplicar varios métodos de clasificación sobre varios conjuntos de archivos de entrada, permitiendo un análisis estadístico posterior entre todos ellos. De esta manera, podremos automatizar esas tareas para extraer conclusiones. Este módulo será clave en la fase de experimentación de este trabajo, ya que esta aplicación puede acceder a

cualquier módulo que se incluya en el programa y que pueda estar disponible también en el Explorer.

Las tareas de esta aplicación se concretan en tres sencillas fases asignadas cada una a una pestaña del interfaz gráfico y que siguen el flujo normal de un experimento: la definición en sí del experimento a través de la selección de fuentes y clasificadores, así como del número de repeticiones (*Setup)*, la fase propiamente de ejecución del experimento, donde simplemente se controla la ejecución y los posibles errores (*Run*) y la fase de análisis y evaluación donde se especifican las métricas estadísticas con las que se quiere evaluar el experimento (*Analyse*)

- Knowledge Flow

Si la aplicación Experimenter permite definir un experimento que compare para varias fuentes de datos varios clasificadores, la aplicación Knowledge Flow ofrece la posibilidad de crear un flujo de operaciones secuencial, donde el usuario puede definir una serie de tareas como la aplicación de tratamientos de preprocesado, filtros, detección de clusters, aplicación de clasificadores, evaluación de métricas, rutinas de visualización, etc.

- Workbench

Está es la última aplicación añadida a WEKA y básicamente permite combinar todas las interfaces comentadas hasta ahora en un solo lugar, permitiendo combinar cada una de ellas sin necesidad de abrir nuevas aplicaciones.

- Simple CLI

Presente desde las primeras versiones de la herramienta, es una abreviación de *Simple Command Line Interface*, permitiendo enviar comandos directamente a la aplicación. Esto permite utilizar todos los argumentos y parámetros que están disponibles en cada algoritmo y a su vez, combinarlos en un script para personalizar la tarea al máximo, aunque requiere conocer previamente las opciones de cada módulo o algoritmo implementado, lo que puede ser testado previamente desde el Explorer.

2 Estado del Arte

Como se ha mencionado en la introducción, el problema de la clasificación aplicada a los conjuntos de datos desbalanceados ha sido un área de estudio con mucha actividad en la comunidad del Machine Learning y de la Minería de Datos, una vez que se detectó las particularidades de este tipo de problema y los problemas que planteaban los algoritmos de clasificación que se usaban típicamente en problemas "clásicos", esto es, con un desbalanceo moderado.

Como ya adelantábamos, ha habido muchas aproximaciones distintas a este problema, aplicando multitud de algoritmos y variantes. Entre todas estas aproximaciones podemos destacar tres que agrupan a su vez varias técnicas diferentes:

- Algoritmos específicos para conjuntos de datos desbalanceados
- Selección de atributos (*Feature Selection*)
- Técnicas de remuestreo o tratamiento de datos

A continuación, se profundizará en las características principales de estas tres aproximaciones, como han evolucionado, las razones de su coexistencia y el estado actual de las mismas, para finalmente resumir en un cuadro comparativo las principales características de estas tres aproximaciones.

Seguidamente, se revisarán las métricas de evaluación utilizadas específicamente para la evaluación de clasificadores aplicados sobre conjuntos de datos desbalanceados, teniendo en cuenta los problemas asociados a los criterios de evaluación de los clasificadores cuando se aplican a este tipo de distribuciones. Se describirán las distintas alternativas posibles con un foco especial en aquellas que serán utilizadas en la fase de experimentación posterior del trabajo.

2.1 Algoritmos específicos para conjuntos de datos desbalanceados

Una vez demostrado que los algoritmos clásicos, que se aplicaban a problemas comunes con un desbalanceo moderado, no eran adecuados para enfrentarse a los problemas de conjuntos de datos muy desbalanceados, se comienzan a proponer varios algoritmos nuevos específicos para este tipo de problemas.

Los métodos de **aprendizaje de una clase**, también conocidos como métodos basados en el **reconocimiento**, se basan en la especialización en identificar la clase clave (clase minoritaria) y rechazar las demás basándose en la modelización de los ejemplos de la clase clave para así reconocerla como "objetivo de identificación". La definición del umbral de decisión es la clave para evitar falsos positivos, con un umbral bajo, o falsos negativos, con un umbral agresivo. Esta técnica parece funcionar bien en problemas que combinan un desbalanceo alto y un ruido dimensional alto [24].

Otros clasificadores se basan en tratar de maximizar una **función objetivo** asociada al conjunto de datos objetivo, la clave de este tipo de clasificadores es que se ha observado que la mayor

parte de las aplicaciones de problemas reales tienen unos costes de error de clasificación distintos, no uniformes y no homogéneos. De hecho, en muchas ocasiones los costes asociados a cada tipo de error no se conocen y estos clasificadores deben generar su matriz de costes para caracterizarlos, proceso que tiende al sobreajuste. Para los conjuntos de datos desbalanceados, se aplica un desplazamiento en el coste de la función para romper la imparcialidad a favor de la clase minoritaria [24].

Una propuesta específica de **red neural modular** (MNN) basada en la técnica de divide-and-conquer también ha sido propuesta [25]. En este caso, transforma un problema multiclásico desbalanceado en un problema simétrico biclásico, mostrando un desempeño mejor para el tipo de problemas desbalanceado que la utilización de una red neural clásica.

A partir de los algoritmos de **SVM** (*Support Vector Machines*) se han propuesto variantes para mejorar la clasificación de datos desbalanceados, como la que los combina con la técnica de función objetivo, que muestran buenos resultados cuando no hay ruido presente [24].

La **combinación de métodos** o **clasificadores múltiples (MCS)** se basan en la utilización de varios clasificadores para aprovechar las características de cada uno. Hay muchas variantes analizadas para aprovechar de manera óptima las características de los clasificadores constituyentes en función del objetivo perseguido, no restringido a la gestión de distribuciones desbalanceadas. Dos de esas variantes de combinación de clasificadores que se han utilizado en el ámbito referido en este trabajo, pero que no se aplican exclusivamente al mismo, son el *boosting*, que viene a ser una especie de aplicación en serie, donde cada clasificador depende del resultado del anterior y se centra en los errores del mismo y el *bagging*, que podríamos asimilar a una aplicación en paralelo, en el que cada clasificador realiza su función a partir del conjunto de datos original y da su diagnosis, de modo que la decisión final se calcula a partir del resultado de cada uno que, en principio, tendrán el mismo peso. Precisamente los métodos de combinación de los algoritmos SVM han sido ampliamente estudiados [24].

2.2 Selección de atributos (*Feature Selection*)

Esta aproximación se basa en seleccionar, dentro de un conjunto de datos, los atributos o características fundamentales que mejoran la métrica que se desea optimizar en el clasificador. Es importante tener en cuenta esa correlación, ya que la métrica elegida a priori, ligada a las características específicas del problema, puede condicionar la propia selección de atributos, de modo que una métrica alternativa genera una selección de atributos distinta.

La historia de la selección de atributos se inicia como un campo de análisis de la estadística y en los últimos años, con la explosión de la minería de datos, estas técnicas han tenido un gran desarrollo y se han publicado numerosas investigaciones al respecto con amplios análisis experimentales, como las que recoge Huang en su trabajo [26].

Los conjuntos de datos de alta dimensionalidad, esto es, con un gran número de atributos pueden reducir el desempeño del clasificador al ser más sensibles al ruido que generan ciertos atributos y este problema se hace más acuciante al enfrentarnos a conjuntos de datos desbalanceados [27].

Hay que tener en cuenta que las técnicas de selección de atributos surgieron precisamente para reducir la dimensionalidad como se ha comentado y, por tanto, no son exclusivas de la solución a los problemas de los conjuntos de datos desbalanceados. Como hemos mencionado, los trabajos de la selección de atributos se iniciaron en estadística hace varias décadas y así, estas técnicas se usan también para evitar el riesgo de sobreajuste de los clasificadores, para generar modelos más eficientes y rápidos y para obtener un mayor conocimiento del modelo de clasificación, por el hecho de haber identificado las características o atributos clave del conjunto de datos.

Los métodos de selección de atributos se basan en los conceptos de la redundancia de características y relevancia de características que son revisados por Huang en su trabajo [26], donde también presenta una taxonomía de los distintos métodos, que se incluye en la **Fig.2**.

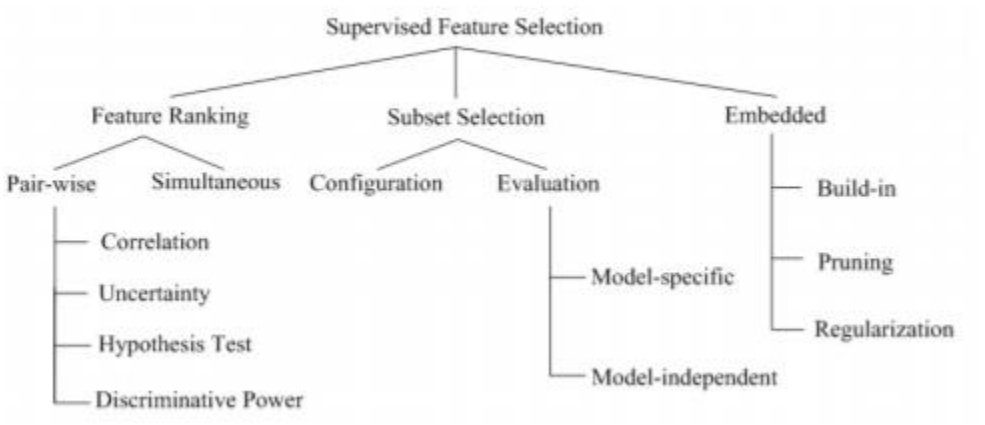

Fig. 2: *Clasificación de algoritmos de Selección de Atributos, según Huan, S. (2015)*

En primer lugar, distinguimos una aproximación que se basa en el ranqueo de los atributos y. dentro de esa clase, se pueden distinguir, por un lado, aquellos que usan distintas técnicas (correlación, incertidumbre, test de hipótesis y potencia discriminante) que comparan por separado cada pareja de atributos y por otro, los que realizan un ranqueo simultáneo de todos los atributos. El inconveniente de estos métodos es que, por un lado, para el ranqueo por parejas de atributos, pueden pasar desapercibidos aquellos atributos que interactúan en grupo con bajas dependencias individuales y, por otro lado, para el ranqueo simultáneo, el proceso de detección del set de atributos que interactúan en el contexto bajo estudio, en nuestro caso la clasificación de una distribución desbalanceada, puede ser muy complejo.

Una segunda aproximación es la selección de un subconjunto de atributos, que también puede a su vez dividirse en dos categorías; aquellas que analizan las correlaciones entre atributos, entre otros métodos cómo la manta de Markov, los filtros de mínima-redundancia-máxima-relevancia

y los filtros de correlación. Esta técnica, sin embargo, no garantiza una selección que ayude a eliminar los problemas asociados a las distribuciones desbalanceadas.

La segunda categoría dentro de esta aproximación de selección de un subconjunto de atributos, cubre los métodos de evaluación que determinan lo bueno que es la selección llevada a cabo, basándose en la métrica que predefinamos y consideremos adecuada y así, eligiendo convenientemente la métrica adaptada a nuestro problema de clases desbalanceadas es como adaptamos el método al problema concreto de optimizar el comportamiento para una distribución desbalanceada con una métrica de evaluación específica.

Finalmente, los métodos embebidos hacen referencia a algoritmos que ya están basados en una selección de atributos (como los árboles de decisión); en métodos de poda, que van eliminando atributos tras comprobar que no se degrada el desempeño del clasificador y en los métodos de regularización, que se basan en funciones objetivo que minimizan errores de ajuste.

Al igual que sucede con otras aproximaciones del campo del Machine Learning, los trabajos publicados apuntan a que no hay un método de selección de atributos que sea superior a otros de manera universal. No solo eso, en algunos estudios, se ha concluido que las evaluaciones basadas en validación cruzada, que han sido las que se han utilizado de manera clásica para otros métodos, pueden no ser la mejor aproximación específicamente para una evaluación correcta de las técnicas de selección de atributos, al tender al sobreajuste [28], proponiéndose realizar la evaluación con conjuntos de datos totalmente independientes.

2.3 Técnicas de remuestreo o tratamiento de datos

Las técnicas de remuestreo, incluidas dentro de aquellas conocidas como de tratamiento o de preprocesamiento de datos, se basan en la transformación del conjunto de datos inicial para generar un conjunto de datos alternativo, siendo el objetivo principal de esa transformación, lograr una distribución de las clases más balanceada que en la base de datos de partida, de modo que, al aplicar un clasificador clásico conocido, desaparezcan los problemas asociados al desbalanceo.

La gran ventaja de esta aproximación se basa en que, una vez realizado el tratamiento de datos para obtener el conjunto de datos alternativo, puede aplicarse cualquier algoritmo de clasificación clásico, lo que permite elegir la aproximación más conveniente al tipo de problema, teniendo en cuenta las características ya ampliamente estudiadas de los algoritmos de clasificación disponibles.

En la evolución de las técnicas de remuestreo podemos distinguir históricamente la aparición de los métodos de remuestreo básicos y los métodos de remuestreo avanzados, también referidos como “inteligentes” que aparecieron posteriormente.

Si nos centramos en los problemas biclásicos, estás técnicas nos permitirán, a partir de la base de datos original, generar una distribución alternativa donde aumentaremos la proporción de muestras de la clase minoritaria y/o reduciremos la proporción de muestras de la clase

mayoritaria modificando de esta manera la proporción de muestras de cada clase y por tanto rebalanceando la distribución respecto a la del punto de partida.

Los métodos de remuestreo básico son el submuestreo y el sobremuestreo. El submuestreo se basa en eliminar elementos de la clase mayoritaria (o alternativamente escoger una muestra subconjunto de la clase mayoritaria que formará parte de la nueva distribución) y preservar todos los casos de la clase minoritaria en el nuevo conjunto de datos. La forma más básica de hacer submuestreo es eliminar aleatoriamente parte de los casos de la clase dominante, técnica conocida como *Random Subsampling* (RANSUB). Esta técnica es atractiva por su rapidez y bajo coste computacional en la aplicación a conjuntos de datos muy grandes, aunque, como contrapartida, el descarte aleatorio de elementos de la distribución original puede generar pérdida de información, si eliminamos los casos cerca de la frontera de decisión, que pueden comprometer el desempeño a posteriori del clasificador.

El sobremuestreo, por el contrario, se basa en replicar aquellos elementos de la clase minoritaria hasta alcanzar un número comparable al de la clase mayoritaria. El método más básico de hacer sobremuestreo es elegir de manera aleatoria aquellos elementos a replicar, técnica conocida como *Random Oversampling* (RANOVER). En este caso no se pierde información, pero esta técnica tiene otros inconvenientes, como el mayor coste computacional, el hecho de replicar casos en lugar de introducir nuevos datos, de modo que el problema de fondo de la falta de información persiste o el riesgo de replicar patrones mal clasificados, por lo que es importante asegurar que partimos de una correcta clasificación inicial. El riesgo de sobreajuste al replicar elementos de la clase minoritaria es otro de los inconvenientes de estas técnicas.

Los métodos de remuestreo avanzados proponen añadir o eliminar las muestras de manera adaptativa, incluyendo la generación de nuevos elementos de la clase minoritaria de manera sintética a partir de las muestras reales disponibles.

El método SMOTE (*Synthetic Minority Oversampling TEchnique)*[29] es un método de sobremuestreo avanzado que genera nuevos patrones de la clase minoritaria localizados en el espacio i-dimensional de sus atributos entre una muestra de la clase minoritaria y sus k vecinos más cercanos, a partir de una suma de vectores que unen el elemento original con cada uno de sus vecinos, multiplicado cada uno por un factor aleatorio entre 0 y 1. Esta técnica está ampliamente extendida como método de enfrentarse al aprendizaje de los sistemas desbalanceados, con una amplia historia de aplicaciones en los últimos años y está implementado en la mayoría de las herramientas de machine learning y todavía es fruto de análisis en distintas publicaciones [30].

Como alternativa para reforzar las fronteras de decisión varios autores propusieron el SMOTE Frontera (*Borderline-SMOTE*) [31], en el que los elementos creados sintéticamente se localizan en la frontera, considerando tales casos objetivo del algoritmo, aquellos elementos de la clase minoritaria que tienen más de la mitad de sus k vecinos más cercanos en la clase mayoritaria, con dos variantes, en una usando sólo los vecinos de la clase minoritaria y en otra usando también los vecinos de clase mayoritaria, utilizando un factor “reducido” (entre 0 y 0.5) para los vectores que conectan con esos vecinos de la clase mayoritaria. De esta manera, esta técnica trata de realizar

el sobremuestreo y a la vez definir con más precisión la frontera de decisión a partir de los datos disponibles.

En cuanto a técnicas de submuestreo avanzado, varias técnicas han sido propuestas como el submuestreo basado en clusters [32] o el CPM (*Class Purity Maximization)* con buenos resultados en problemas sin desbalanceo extremo [33] que, después de dividir la distribución en clusters, filtra aquellos clusters que sólo contienen muestras de la clase mayoritaria. La técnica ENN (*Edited Nearest Neighbour)* [34] se basa por el contrario en eliminar una muestra que podría considerarse ruidosa, eliminando aquellas muestras en las que dos de los tres vecinos más cercanos pertenecen a la clase contraria a la de la muestra, ya que esta técnica puede aplicarse tanto a la clase mayoritaria como a la minoritaria. Esta técnica tiende a reducir el ruido y puede eliminar patrones mal clasificados en el conjunto original, pero, si estamos seguros de que la clasificación original era correcta, puede eliminar información importante en la frontera de decisión.

Finalmente podemos encontrar métodos híbridos, que combinan varias técnicas, siendo uno de los más comunes referenciados el SMOTE-ENN [35]. Este método contempla un sobremuestreo inicial y una limpieza con ENN, habiéndose también analizado la variante del ENN-SMOTE [37], consiguiendo de esta manera, una aproximación de menor coste computacional al realizar primero la limpieza, técnica que ha demostrado unos resultados similares al SMOTE-ENN en algunos escenarios.

La cuestión de la determinación de la **distribución óptima** que genera el mejor desempeño del clasificador que se usará a posteriori es la clave de estas técnicas. Ya inicialmente Weiss & Provost [36] demostraron empíricamente que, efectivamente, se puede encontrar una diferencia significativa entre distintos porcentajes de desbalanceo obtenido a partir de distintos patrones de rebalanceo y que los rangos óptimos son contiguos para el porcentaje de error de clasificación apareciendo los máximos en distintos porcentajes de distribución dependiendo del problema (dependencia del contexto).

El propio trabajo ya señalaba las limitaciones asociadas a la utilización de un árbol de decisión como algoritmo de aprendizaje y que los análisis realizados podrían tener una alta dependencia del clasificador a utilizar, del mismo modo que había quedado demostrado que existía una dependencia con la métrica. En cualquier caso, los resultados muestran que la aproximación clásica de una distribución natural perfectamente balanceada al 50% no es la distribución óptima para todos los casos.

2.4 Resumen comparativo de las aproximaciones para clasificar distribuciones de datos desbalanceadas

Como resumen de las aproximaciones comentadas y principales familias de algoritmos utilizados para la clasificación de distribuciones desbalanceadas, se presenta la **Tabla 3** que recopila las principales características asociados a cada uno de ellos, en función de la literatura publicada al respecto.La tabla presentada resume las distintas alternativas comentadas como aproximaciones al problema del tratamiento de la clasificación desbalanceada, destacando como ejemplos concretos los algoritmos comentados, con las ventajas y desventajas más significativas de cada método.

Aproximación	Generalidades	Algoritmo	Ventajas	Inconvenientes
Algoritmo Específico	Obtienen buenos resultados en problemas específicos. Pueden requerir una adaptación al problema concreto compleja	BoostingSVM	Mejora la predicción de la clase minoritaria respecto al SVM	Ignora la distribución de la clase desbalanceada. Sensibles al ruido
		Método de reconocimiento	Buen rendimiento en desbalanceo extremo y ruido	No puede aplicarse a algoritmos de árboles de decisión
		Función de coste objetivo	Buenos resultados tras ajustar la función al problema	Complejidad para definir la función de coste. Tienden al sobreajuste
Selección de atributos	Ayudan a aprender y simplificar la dimensionalidad de los problemas. Afronta el problema de escalabilidad en conjuntos de datos grandes con alta dimensionalidad.	Ranqueo por pares de atributos	Bajo coste computacional, permite técnicas de preprocesamiento	Dificultad para detectar atributos que interactúan en grupo sin dependencias individuales significativas
		Ranqueo simultaneo (Relief)	Detecta dependencias condicionales (mejora sobre el ranqueo por pares)	La capacidad de identificar atributos que interactúan en grupo bajo estudio puede ser muy costosa
		Selección de subconjunto de atributos por correlación	Coste computacional efectivo	No es capaz de estimar la precisión del modelo, requiere una evaluación posterior. Puede persistir el problema del desbalanceo
		Selección de subconjunto de atributos por evaluación	La evaluación va incorporada.	Alto coste computacional, no aplicable a problemas con un gran número de atributos.
Remuestreo	Versátiles para usar otros clasificadores, introducen el problema de la distribución óptima. Básicos: fáciles de implementar, pero riesgos de pérdida de información o sobreajuste Avanzados: evitan los problemas de los básicos, pero aumenta el coste computacional	Submuestreo aleatorio (básico)	Coste computacional bajo. Independiente del clasificador posterior a utilizar	Puede eliminar información útil
		Sobremuestreo aleatorio (básico)	Facilmente implementable. Independiente del clasificador posterior a utilizar	Introduce coste computacional al algoritmo siguiente, puede tender al sobreajuste
		Remuestro inteligente (SMOTE/ENN)	Independiente del clasificador posterior a utilizar. Implementación sencilla	Introduce coste computacional.

***Tabla 3**: Tabla comparativa de aproximaciones y principales algoritmos utilizados para la clasificación de distribuciones desbalanceadas*

2.5 Métricas de evaluación utilizadas para la evaluación de clasificadores aplicados sobre conjuntos de datos desbalanceados

Las métricas básicas basadas en los términos de la matriz de confusión comentadas en la introducción (*1.3 Criterios de evaluación de clasificadores*) tienen la ventaja de que son fáciles de calcular con un bajo coste computacional, y los resultados obtenidos a partir de la aplicación de estas métricas, son fáciles ordenar e intuitivos, por lo que típicamente han tenido mucha aplicabilidad.

Sin embargo, como también se presentaba en la introducción, algunas de estas métricas, como la exactitud tiene problemas para discriminar y caracterizar correctamente cuál es el mejor clasificador [38] y puede llevar asociado un sesgo negativo hacia las clases minoritarias en las distribuciones desbalanceadas [39].

Las métricas recogidas en la **Tabla 2** no son adecuadas para discriminar convenientemente el óptimo comportamiento de un clasificador usando tan sólo una de ellas. Además, en las distribuciones desbalanceadas hay que tener en cuenta las consideraciones ya comentadas en cuanto a los porcentajes de errores de clasificación cometidos sobre la clase mayoritaria y la minoritaria sobre el total de instancias.

A continuación, se presentan métricas alternativas, analizadas en distintos estudios, que puedan plantear mejoras sobre las métricas básicas presentadas:

- **Área bajo la curva ROC**

  El acrónimo ROC se refiere a *Receiver Operator Characteristic* y proviene de la teoría de detección de señales desarrollada por los ingenieros y operadores de radares para medir la eficacia en la detección de objetos enemigos y es una representación gráfica de la **sensibilidad** frente a la **especificidad** para un sistema de clasificación binario o biclásico. La ventaja de la curva ROC es que es independiente de la distribución de las clases.

  El área bajo la curva ROC, que en literatura se denomina tambien AUC (*Area Under the Curve)*, donde la curva en este acrónimo se refiere precisamente a la curva ROC, es el indicador cuantificable asociado a una curva ROC que más se utiliza para caracterizar el rendimiento del clasificador asociado y este índice se puede interpretar como la probabilidad que tiene el clasificador de etiquetar un elemento elegido aleatoriamente del conjunto de datos como de manera correcta (o con una probabilidad mayor asociada a la clase correcta), [40] y [41].

  También se ha demostrado que el área bajo la curva ROC puede ser interpretada como una aplicación del análisis estadístico de Wilcoxon [42].

  De esta manera un valor de AUC=1, significaría que el clasificador siempre acertaría la clase correcta, un valor de AUC=0 siempre daría el resultado opuesto al correcto (lo que tendría un

gran poder discriminante, pero inverso al real) y AUC =0.5 significaría que el clasificador no es capaz de distinguir casos positivos y negativos, siendo similar a una selección al azar. Por ello, lo esperable es encontrar valores de AUC en el intervalo 0.5<AUC<1, siendo el clasificador tanto mejor cuanto más cerca de 1 se encuentre.

Computacionalmente, el resultado de AUC requiere el cálculo de una integral en un caso de variable continua, pero para un caso discreto biclásico como el que nos ocupa, siendo un cálculo más complejo que el de las métricas básicas, puede simplificarse al cálculo de la siguiente fórmula:

$$AUC = \frac{\sum_{i=1}^{n+} \sum_{j=1}^{n-} 1_{f(x_i^+) > f(x_j^-)}}{n^+ n^-}$$

Donde la función *f(x)* se refiere a la probabilidad (función de decisión) de que un caso sea definido como clase positiva o negativa; el subíndice aparece aplicado al 1 en la fórmula, es decir, $f(x_i^+) > f(x_j^-)$, significa que, si se cumple tal, la función vale 1 y en caso de no cumplirse la condición, la función vale 0; *n+* y *n-* sería el número de casos positivos y negativos de la distribución original. Por tanto, si un clasificador *f(x)* es capaz de otorgar una probabilidad de clase positiva mayor que la probabilidad de clase negativa a una muestra positiva maximiza el valor de AUC, que se normaliza con el denominador a 1 [41].

- **Estadístico Kappa**

El estadístico Kappa o índice Kappa es un índice de concordancia propuesto por Cohen que, aplicado a los clasificadores, mide la concordancia existente entre las frecuencias de ocurrencia asociadas a cada clase y las frecuencias obtenidas al usar el clasificador, de modo que las proporciones entre las clases forman parte de la construcción del propio índice.

De esta manera si denominamos Co al porcentaje de "aciertos" del clasificador bajo estudio y Ca al porcentaje de "aciertos" que se hubiera obtenido al azar, teniendo en cuenta la proporción de casos positivos y negativos que hay en el conjunto de datos bajo estudio y, por tanto, su desbalanceo, el índice K sería igual a:

$$K = \frac{Co - Ca}{1 - Ca}$$

Si K=0, entonces no hay diferencia entre los resultados del clasificador y aquellos obtenidos por puro azar. Si K alcanza el valor de 1 entonces el clasificador acierta todos los casos (*Co=1*) y el clasificador es perfecto. Un valor negativo es posible e indica discordancia, es decir el clasificador predice la clase opuesta a la real. La ventaja del estadístico Kappa, como vemos es que el término *Ca* que incluye, ya tiene en cuenta la razón entre las clases en el conjunto de datos bajo estudio [43]

- **Otras métricas**

  Además de las comentadas arriba, existen otras métricas que se utilizan para la evaluación de clasificadores aplicados a distribuciones de datos desbalanceadas. Entre ellas se pueden mencionar:

  - Error Cuadrático Medio (MSE), que calcula el valor medio de la diferencia al cuadrado entre el valor real y el predicho para todos los puntos de datos, usando funciones de discretización como la comentada en el cálculo del AUC para las aproximaciones discretas biclásica [44]. Al elevar al cuadrado los valores, los errores positivos no se compensan con los negativos. Pero, además, los errores elevan el índice más rápidamente que su métrica "hermana", el MAE. (Error Medio Absoluto) donde se usa simplemente el valor absoluto para evitar la compensación entre errores "negativos" y "positivos".

  - La medida F (F-Measure) es una media armónica entre Precisión y Exhaustividad, es decir:

    $$F - measure = 2\frac{Precisión * Exhaustividad}{Precisión + Exhaustividad}$$

    F-measure se mueve en el intervalo [1,0], siendo 1 el valor asociado a una perfecta precisión y exhaustividad (resultados de clasificación perfectos) y 0 un valor en el que o bien la precisión o la exhaustividad es cero. Este índice tambien es conocido como el DSC (*Dice Similarity Coefficient*).

  - Área bajo la curva PRC (Precison Recall Curve, que viene a ser una variante de la ROC) y a su vez está relacionada con la F-Measure al tener en cuenta las mismas variables. Típicamente suele utilizarse menos frecuentemente que la curva ROC. Los problemas asociados a la dependencia de la F-Measure y la PRC sigue siendo objeto de estudio actualmente. [44]

  - Métricas basadas en gráficas, que son capaces de asistir evaluaciones desde distintos puntos de vista, donde podrían englobarse las comentadas ROC y PRC, pero extendidas a otras combinaciones como B-ROC o curva de coste entre otros. [45]

  - Además, existen métricas específicamente diseñadas para determinados clasificadores, como la ganancia de información o entropía, usadas en clasificadores de árboles de decisión. El problema de este tipo de métricas es que, cuanto más específicas sean, más se alejan de la posibilidad de ser usadas para una comparación del desempeño de los distintos clasificadores, por motivos obvios.

En un primer análisis con el clasificador implementado en este trabajo[3], se revisó una preselección de cuatro métricas implementadas en WEKA, donde se comprobaron las evoluciones de esas métricas en función de los distintos porcentajes de distribuciones tras remuestrear la base de datos original, realizando el análisis en una pequeña muestra de bases de datos extraídas del conjunto que forma parte posteriormente del set experimental. Los resultados de dos de esas bases de datos, a modo de ejemplo, pueden comprobarse en los **Apéndices**, en el apartado *7.1 Análisis del comportamiento de métricas de bondad al variar la distribución de la clase minoritaria aplicando remuestreo*.

En el análisis de este estudio nos basaremos en la aplicación de las métricas del Área bajo la curva ROC y Estadístico Kappa, aunque el algoritmo que desarrollaremos en WEKA estará preparado para optimizar otras métricas que hemos mencionado aquí y ya implementadas en el entorno de trabajo.

Seleccionamos estas dos métricas ponderando los siguientes motivos:

- Implementación de la métrica disponible previamente en el entorno de trabajo WEKA (al igual que otras métricas que podrá manejar el módulo, aunque no formarán parte del análisis de resultados)
- Observación del comportamiento de las métricas al variar la distribución de la clase minoritaria aplicando remuestreo.
- Posibilidad de aplicación de la métrica a distintos tipos de clasificadores, al no tratarse de métricas específicas.
- Amplia aplicación en distintos estudios previos sobre los tipos de clasificadores y, sobre todo, de distribuciones de datos objeto de análisis.
- Orientación al análisis de conjuntos de datos desbalanceados, frente a otras métricas básicas como la precisión o el porcentaje de errores que, en principio, son de más amplia aplicación en otros tipos de distribuciones de datos, pero presentan inconvenientes ante distribuciones desbalanceadas, como se ha comentado.

[3] Ver detalles en el capítulo *4 Implementación del algoritmo en WEKA*

3 Propuesta de Investigación

3.1 Punto de partida

A partir de la revisión del estado del arte, se puede concluir que la cuestión de la dependencia del contexto en el problema de la clasificación de conjuntos de datos desbalanceados es una característica que aparece de manera constante en los distintos artículos revisados.

Teniendo en cuenta esta característica, se plantea la automatización de un método que pudiera, a partir de la definición propia de dicho contexto para un problema de distribución inicialmente biclásico, encontrar un proceso de optimización de la distribución de clases mediante la técnica de remuestreo más apropiada, con el objetivo de obtener el mejor resultado posible por el algoritmo de clasificación, evaluado sobre una métrica o criterio de bondad que pueda ser seleccionada a priori como parte del propio contexto del problema.

Para ello y teniendo en cuenta la versatilidad que aportan los métodos de remuestreo, al poder apoyarse en las características que aportan los distintos clasificadores en los que se basan, se utilizará un método basado en la búsqueda, para un algoritmo de remuestreo concreto y predefinido, del porcentaje de remuestreo óptimo para un problema dado, permitiendo la comparación del resultados de distintos algoritmos de remuestreo bajo una cierta métrica, con el objetivo de que dicha técnica pueda aplicarse a la multitud de problemas de clasificación desbalanceada aplicados del mundo real que están emergiendo día a día.

Dentro de las variantes de técnicas de remuestro, aquellas denominadas como de remuestreo inteligente son las más prometedoras respecto a los algoritmos básicos ya que, pese a requerir una cierta complejidad computacional adicional, puede ser capaz de solventar ciertos problemas que en ocasiones aparecen en ciertos conjuntos de datos respecto a la eliminación de la información útil o el sobreajuste por la propia distribución de la clase minoritaria en el espacio dimensional de atributos. Puesto que perseguimos analizar la versatilidad del algoritmo, trataremos de comparar el comportamiento de los distintos algoritmos sobre un conjunto amplio de bases de datos que nos permita determinar una generalización a la hora de comprobar los resultados aplicados a todo el conjunto.

A partir del comentado análisis de Weiss & Provost [36] respecto al análisis de la distribución óptima, en los trabajos de I.Albisua et al. [37] y [46] se plantea por primera vez una metodología estructurada para encontrar la distribución óptima sobre los algoritmos de remuestreo principales.

En estos trabajos se describe un proceso basado en dos fases con el objetivo de encontrar un remuestreo de la base de datos original con el porcentaje de distribución entre la clase mayoritaria y minoritaria más adecuado para la aplicación posterior de un clasificador concreto que se evaluará con una métrica específica.

Así, en [37] se presenta el análisis aplicado a 26 bases de datos obtenidas de la UCI para los clasificadores C4.5 y CTC, que es un algoritmo de construcción de árboles consolidado. Este

trabajo se utilizará como punto de partida para la investigación que propone este trabajo como un complemento al mismo.

En este trabajo generaremos para el entorno de trabajo WEKA, un código para la implementación del procedimiento propuesto en [37], de una manera automatizada y flexible, basada en las características de modularidad de WEKA.

Basados en ese procedimiento, trabajaremos con un conjunto de bases de datos alternativo de un tamaño algo mayor que presentaremos a continuación y, apoyados en la flexibilidad que aporta el entorno de trabajo, analizaremos, para un conjunto de algoritmos de clasificación ya implementados en WEKA, los resultados de las distintas alternativas de remuestreo que hemos implementado, con el objetivo de extender las conclusiones del estudio usado como punto de partida.

3.2 Procedimiento de investigación propuesto

En la primera fase del procedimiento propuesto en estudio de partida, tal y como se refleja en la **Fig.3** se realiza, sobre la distribución original de datos, una validación cruzada partiendo cinco veces en 10 subconjuntos los datos, de los cuales, en cada caso, se utilizan 9 porciones para el entrenamiento y se reserva la décima parte para la evaluación, manteniendo en cada porción un número proporcional de cada clase respecto a la distribución original. De cada subconjunto de entrenamiento se generan 100 submuestras para cada una de las 13 proporciones de la clase minoritaria definidas en la siguiente lista: (2%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 95%, 98%), lista a la que se añade el porcentaje de la distribución original. De esta manera se generan 5x10x14x100 muestras por cada conjunto de datos y todas las submuestras tienen un tamaño idéntico: el número de casos de la clase minoritaria en el conjunto de entrenamiento: de esta forma aseguramos que es posible generar cualquier clase de distribución determinada por la lista sin necesidad de replicar ningún caso, ya que aplicaremos en esta fase únicamente la técnica del submuestreo aleatorio (RANSUB) por su bajo coste computacional.

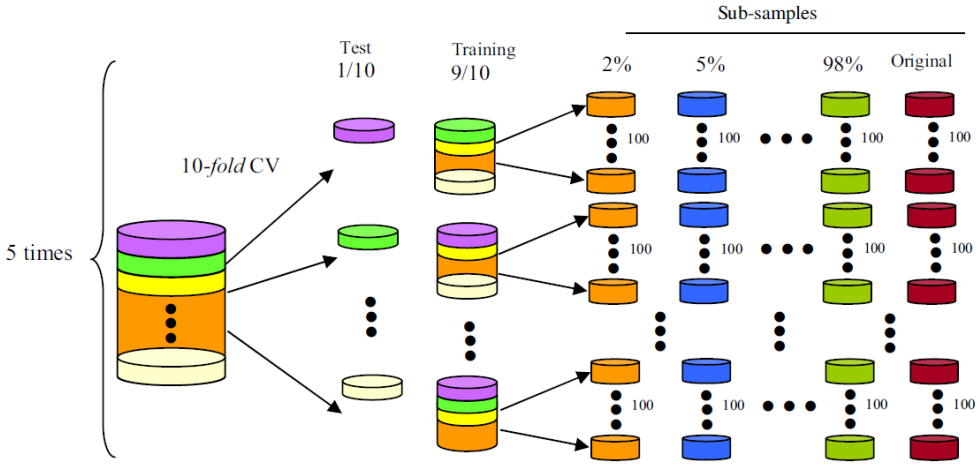

Fig. 3: *Esquema de propuesta para la fase 1 de la investigación de un método automatizado apoyado en WEKA para una búsqueda de la distribución óptima para un remuestreo, propuesto por Albisua et al. (2010), implementado en WEKA*

En nuestro caso, respecto a la implementación de esta primera fase, reutilizaremos todas las propuestas para esta primera fase del trabajo de partida [37], reduciendo el número de particiones (*folds*) a 5 subconjuntos de datos, reutilizando la lista de barrido propuesta y el tamaño de muestras, por lo que generaremos un total de 5x5x14x100 muestras.

Tras esta primera fase, seremos capaces de estimar para cada conjunto de datos la distribución óptima a través de la evaluación de las distribuciones obtenidas por un submuestreo aleatorio básico, en concreto el de tamaño mínimo (RANSUB Size Min).

En la segunda fase, ilustrada en la **Fig.4**, se incluyen los métodos de remuestreo alternativos y evaluaremos cuál es la mejor técnica de remuestreo para el conjunto de datos específico, pero en lugar de probarlo en todas las posibles distribuciones, nos basaremos en los resultados de la primera fase. Utilizaremos, del mismo modo, una validación cruzada partiendo cinco veces en cinco subconjuntos los datos y generando 50 muestras en cada subconjunto en esta segunda fase.

Ahora bien, no podemos asegurar que el valor óptimo sea el mismo para todas las técnicas, si bien los resultados de Weiss y Provost y de los análisis posteriores muestran una posición contigua de los valores óptimos y, por ello, se propone realizar la prueba con la distribución óptima hallada en la fase 1, una distribución con un desbalanceo un 5% y un 10% superior y otra con un desbalanceo un 5% y un 10% inferior a esa distribución óptima de fase 1 y además, incluir la evaluación de una distribución perfectamente balanceada (50%) como referencia fija, en el caso de que esta no esté ya incluida en el conjunto de prueba descrito para la fase dos, asociada aquella aproximación clásica que se propuso inicialmente en cuanto a objetivo de rebalanceo. Cada método de remuestreo que probemos así, es evaluará para estos cinco (o seis en el caso de que la muestra balanceada se incluya adicionalmente) valores de distribución, por lo que en el caso más amplio generaremos 5x5x6x50 muestras para esta segunda fase.

Como en la segunda fase podemos elegir el tipo de remuestreo a aplicar, repetiremos este análisis para cada una de las cuatro técnicas de remuestreo que vamos a comparar:

- Submuestreo aleatorio de tamaño mínimo (RANSUB Size Min), siendo el tamaño mínimo el del número de instancias de la clase minoritaria.
- Submuestreo aleatorio de tamaño máximo (RANSUB Size Max), siendo el tamaño máximo el número de instancias que incluye todas las instancias de la clase minoritaria del conjunto de datos original.
- Sobremuestreo aleatorio (RANOVER)
- Método SMOTE

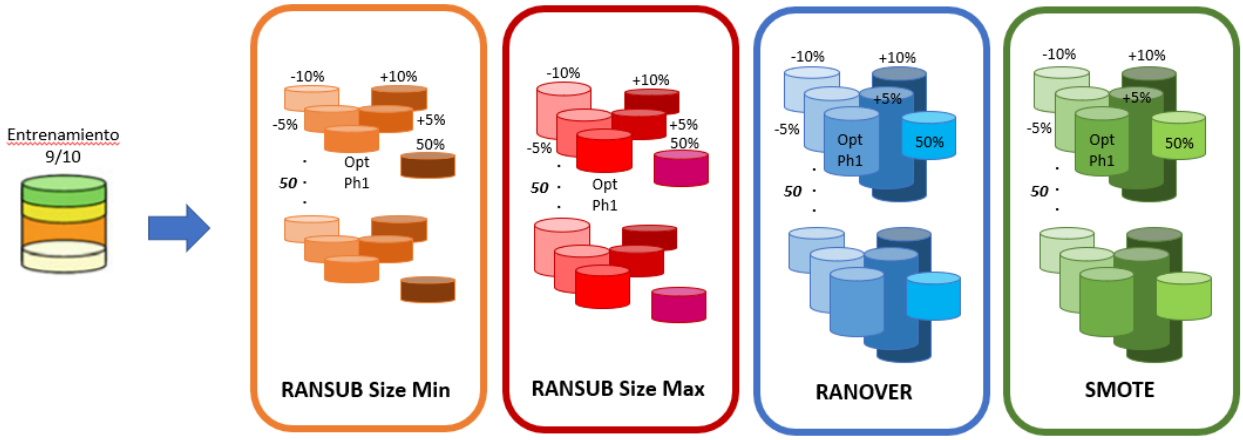

Fig. 4: *Esquema de propuesta para la fase 2 de la investigación de un método automatizado apoyado en WEKA para una búsqueda de la distribución óptima para un remuestreo*

El procedimiento que se propone para llevar a cabo esta investigación se basa en una implementación flexible en WEKA basada en el punto de partida descrito que permita realizar una fase de investigación posterior que extienda el horizonte de los trabajos preliminares.

De esta manera, se plantea que el algoritmo implementado en WEKA permitirá adaptar por el usuario, además de la obvia aplicación a distintas bases de datos, las siguientes variables de entorno:

- La lista de proporciones de la clase minoritaria (barrido) en la fase 1.
- Los intervalos de las proporciones de la clase minoritaria sobre el óptimo calculado en la fase 1 que se aplican en la fase 2.
- El método de remuestreo de la fase 2.
- La métrica de evaluación a aplicar, para la que se persigue el óptimo de la distribución.
- El tipo de clasificador a aplicar.
- El número de veces que repartimos los datos (*runs*) y particiones que hacemos (*folds*) de la base de datos para aplicar el procedimiento.

Como **métodos de remuestreo** se plantea implementar el código de los de submuestreo aleatorio (RANSUB) con sus distintas variantes, el de sobremuestreo aleatorio (RANOVER) y el SMOTE.

De esta manera, la propuesta de investigación pretende utilizar las características de la aplicación de WEKA Explorer y Experimenter, incluida en el propio entorno de trabajo, para facilitar los análisis que permitan una extensión de los incluidos en el trabajo utilizado como punto de partida, de cada a extender el conjunto experimental y generalizar las conclusiones sobre las distribuciones óptimas bajo distintos criterios de bondad para su evaluación.

Para ello, utilizaremos como base de estudio un conjunto de 33 **bases de datos**, cuyo listado aparece en la **Tabla 4**, obtenidas desde el repositorio de KEEL (*Knowledge Extraction based on Evolutionary Learning*) [47] dentro de la categoría de "conjunto de datos desbalanceados" (*Supervised Classification / Imbalanced data sets for* classification) al que se puede acceder a través de su página web seleccionando la opción *KEEL-dataset (Data Set Repository)* en su menú, tal y como se muestra en la **Fig.5**.

KEEL, además de proponer una herramienta open-source que permite trabajar con distintos algoritmos de machine learning, similar a WEKA, aunque con una especial orientación hacia los algoritmos evolutivos, pone a disposición de la comunidad investigadora un repositorio de conjuntos de datos de referencia y resultados de experimentos, gestionado por la Universidad de Granada y que ha sido utilizado en otros trabajos publicados, de modo que permite una fácil comparativa.

Fig. 5: *Menú de la página de KEEL, con el acceso al repositorio de conjuntos de datos resaltado*

Data set	#Atts.	#Examples	Imbalance	Size Of Min. Class	Size of Maj. Class
Abalone19	8	4174	0.77%	32	4142
Yeast6	8	1484	2.49%	37	1447
Yeast5	8	1484	2.96%	44	1440
Yeast4	8	1484	3.43%	51	1433
Yeast2vs8	8	482	4.15%	20	462
Glass5	9	214	4.2%	9	205
Abalone9vs18	8	731	5.65%	41	690
Glass4	9	214	6.07%	13	201
Ecoli4	7	336	6.74%	23	313
Glass2	9	214	8.78%	19	195
Vowel0	13	988	9.01%	89	899
Page-blocks0	10	5472	10.23%	560	4912
Ecoli3	7	336	10.88%	37	299
Yeast3	8	1484	10.98%	163	1321
Glass6	9	214	13.55%	29	185
Segment0	19	2308	14.26%	329	1979
Ecoli2	7	336	15.48%	52	284
New-thyroid1	5	215	16.28%	35	180
New-thyroid2	5	215	16.89%	36	179
Ecoli1	7	336	22.92%	77	259
Vehicle0	18	846	23.64%	200	646
Glass0123vs456	9	214	23.83%	51	163
Haberman	3	306	27.42%	84	222
Vehicle1	18	846	28.37%	240	606
Vehicle2	18	846	28.37%	240	606
Vehicle3	18	846	28.37%	240	606
Yeast1	8	1484	28.91%	429	1055
Glass0	9	214	32.71%	70	144
Iris0	4	150	33.33%	50	100
Pima	8	768	34.84%	268	500
Ecoli0vs1	7	220	35%	77	143
Wisconsin	9	683	35%	239	444
Glass1	9	214	35.51%	76	138
Mean	9.39	919.94	17.61%	120	799.94
Median	8	482	15.48%	52	444

Tabla 4: *Listado y descripción general del conjunto de datos del universo experimental propuesto, disponible en el repositorio KEEL*

Finalmente, una vez definidas las técnicas de remuestreo a probar y el conjunto de bases datos a utilizar, probaremos tales métodos de remuestreo en los conjuntos de datos propuestos para aplicar sobre los siguientes **algoritmos de clasificación**, que ya están implementados en WEKA, partiendo del J48, como implementación del C4.5 que era usado en el trabajo de referencia [37]:

- J48 (Implementación de C4.5 en WEKA)
- Naive-Bayes
- IBk con k=1 (Implementación de KNN, con k=1)
- IBk con k=3 (Implementación de KNN, con k=3)
- PART
- JRIP (Implementación del RIPPER)

Finalmente, utilizaremos como **criterio de bondad** las métricas Kappa y área bajo la curva ROC.

3.3 Hipótesis propuestas

Las preguntas principales que se intentarán responder a partir de la experimentación son las siguientes:

- ¿Merece la pena aplicar el método de remuestreo en vez de utilizar el conjunto de datos original sin ningún tipo de remuestreo? La hipótesis es que, pese a existir una dependencia de contexto, en general la aplicación del método de remuestreo es capaz de mejorar los resultados del clasificador.

- ¿Hay algún método de remuestreo que muestre un mejor desempeño que los demás? La hipótesis es que sí, que los algoritmos de remuestreo más complejos y en concreto, aquellos denominados “inteligentes”, generadores de muestras sintéticas, son más prometedores.

- ¿Está sujeta la elección del método de remuestreo a la dependencia de contexto? La hipótesis es que sí, que, a pesar de que pueda haber una tendencia general que permita caracterizar el desempeño estadístico de algunos métodos de remuestreo por encima de otros, cada contexto específico definido por el conjunto de datos, la métrica de evaluación elegida y el algoritmo de clasificación establecerá un método de remuestreo óptimo no generalizable asociado a tal contexto.

Hay una serie de preguntas secundarias respecto a la caracterización de los resultados de los valores de distribución óptima encontrados en las distintas técnicas de remuestreo:

- ¿Existe alguna relación entre la distribución correspondiente al conjunto de datos original y a la distribución óptima? La hipótesis es que no hay ninguna relación y el valor de la distribución óptima tiene una alta dependencia del contexto.

- ¿Cuántas veces el barrido de fase 2 modifica la distribución óptima calculada en el barrido de fase 1? La hipótesis es que, además de afinar sobre el intervalo, la aplicación de técnicas de remuestreo más complejas, deberían obtener distribuciones óptimas distintas, ya que el método de remuestreo se puede considerar una variable más de contexto.

- ¿Cuántas veces la distribución óptima resulta ser la distribución balanceada (50%)? Los análisis de los trabajos del punto de partida ya mostraban óptimos distintos al de la distribución balanceada, esperamos que las pruebas sobre nuestro universo experimental muestren que, de manera significativa, el balanceo al 50% no es la mejor opción para obtener los mejores resultados.

3.4 Otros objetivos

Por otro lado, más allá del análisis de estos resultados, el objetivo que persigue este trabajo es que el proceso experimental permita la validación de la propuesta desarrollada en WEKA para ponerlo a disposición pública, al tratarse de un entorno *Open Source* y, de esta forma, permitir replicar o extender de manera sencilla los resultados experimentales a la comunidad investigadora y, a su vez, permitir una aplicación directa utilizando la plataforma WEKA sobre problemas prácticos del mundo real.

4 Implementación del algoritmo en WEKA

4.1 Estructura de nuevas clases a incluir en WEKA

Sobre la estructura de funciones definidas en WEKA, vamos a necesitar 6 nuevos elementos de código que se organizan de la siguiente manera:

- Weka
  - src/main/java
    - weka.classifiers.meta
      - *NewDistribution2PhasesClassifier.java*
    - weka.filters.supervised.instance
      - *NewDistSubsample.java*
      - *NewDistOversample.java*
      - *NewDistSMOTE.java*
      - *NewDistSpecifiable.java*
    - weka.filters.supervised.instance.newdistsubsample
      - *InstancesByClass.java*

De todas estas clases, *NewDistribution2PhasesClassifier* contiene el algoritmo del clasificador que estamos implementando, que será descrito más adelante y parte de una implementación anterior de una versión limitada al barrido de la primera clase, sin posibilidad de elegir el método de remuestreo.

Para implementar la propuesta de dos fases y permitir la posibilidad de selección del método de remuestreo, este código llama al resto de clases que necesitamos añadir, por lo que vamos a describirlas brevemente:

InstancesByClass es una clase que extiende la clase *Instances*, localizada en weka.core, para añadir ciertas funciones que resultan de utilidad en nuestro procesado como es la obtención de las clases de un dataset; la obtención de los índices donde empieza cada clase, cuando el dataset está ordenado por su atributo de clase o la adición (concatenación) de nuevos casos a un dataset.

NewDistSpecifiable es un interfaz que ayuda a identificar los filtros seleccionables por nuestro clasificador, ya que vamos a permitir que el usuario seleccione el tipo de remuestreo (submuestro aleatorio, sobremuestreo aleatorio o SMOTE) a partir de los filtros disponibles en WEKA pero sólo aquellas clases específicamente diseñadas para este propósito, por lo que cualquier otro filtro disponible no puede ser utilizado y este interfaz nos ayudará a identificar aquellos filtros permitidos. La condición que deben cumplir los filtros permitidos es que permitan indicar a partir del porcentaje de casos de la clase minoritaria la nueva distribución de clases en el data set, que se traduce en que deberán implementar dos funciones concretas que permiten ese proceso para poder ser seleccionables.

NewDistSubsample es la clase que implementa el subsampleo aleatorio, permitiendo generar una nueva muestra a partir del dataset original, seleccionando elementos sin reemplazo y atendiendo a las condiciones en cuanto a porcentaje de elementos de la clase minoritaria y número de elementos totales que se le indiquen como parámetros. Esta clase cumple con los requisitos de *NewDistSpecifiable*.

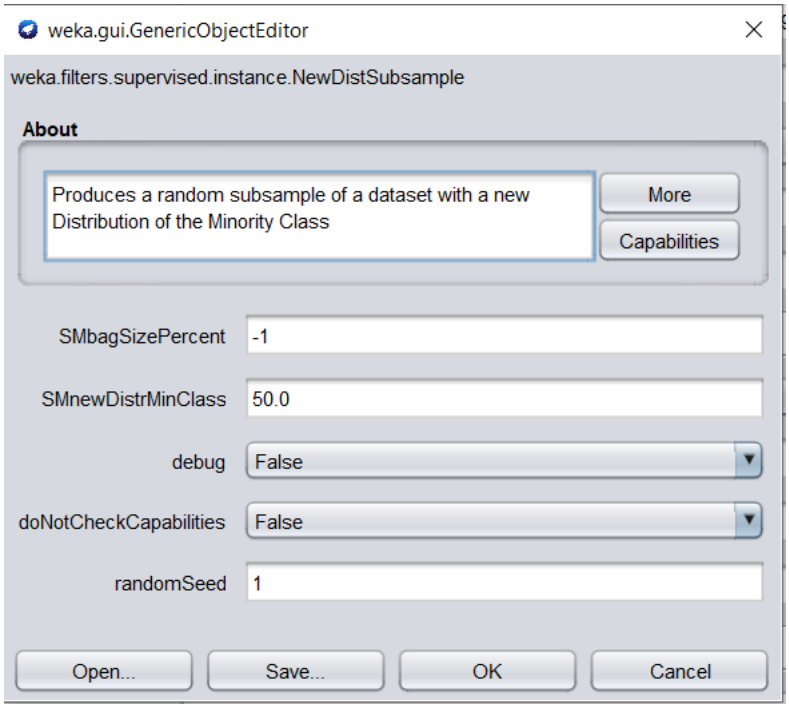

Fig. 6: *Ventana de configuración del filtro NewDistSubsample en WEKA*

Los parámetros configurables de la clase *NewDistSubsample* pueden observarse en la **Fig.6** que refleja la ventana de configuración en el GUI de Weka para este filtro. Estos parámetros son el objetivo de distribución nueva del porcentaje de la clase mínima, el tamaño de la muestra como porcentaje del set de entrenamiento y con dos valores especiales que son los que vamos a utilizar: -1, para el tamaño mínimo, que iguala el tamaño de la clase minoritaria, de modo que siempre se puede asegurar cualquier porcentaje de distribución prefijado y -2, para el tamaño máximo, que utilizará el máximo de muestras posibles a partir del porcentaje de distribución deseada y el número de muestras de clase mínima disponible. Además, podemos controlar la información disponible en consola, el chequeo previo de características (recomendado mantener a False) y el control de la semilla de la función aleatoria.

NewDistOversample es una clase que implementa el sobremuestreo aleatorio, permitiendo generar una nueva muestra a partir del dataset original, añadiendo nuevos casos repetidos de la muestra original, bien de la clase minoritaria o de la mayoritaria en función del porcentaje de

clase minoritaria deseado para la nueva muestra. Esta clase se define como una extensión de la *NewDistSubsample* y por supuesto, cumple con los requisitos de *NewDistSpecifiable*.

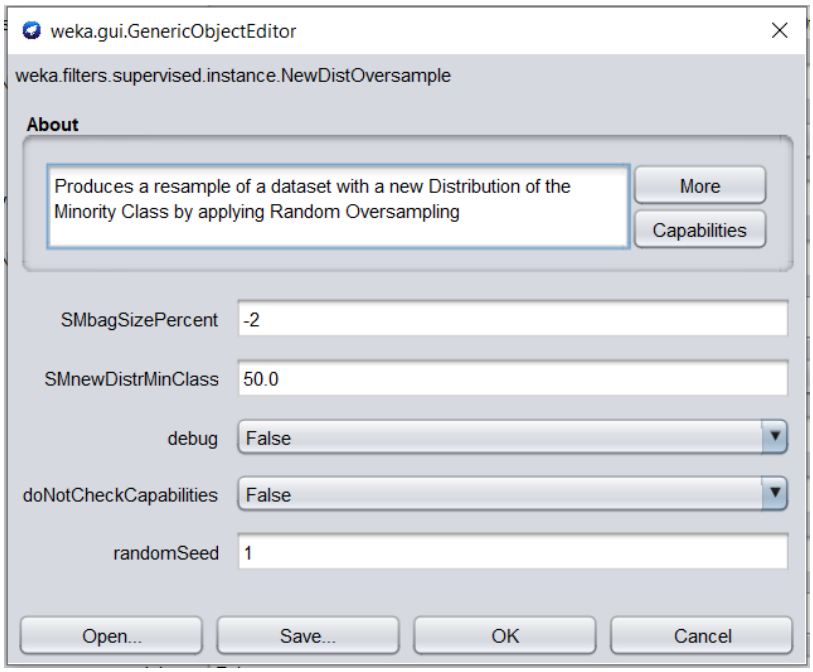

***Fig. 7**: Ventana de configuración del filtro NewDistOversample en WEKA*

Los parámetros configurables de esta clase, que pueden observarse en la ventana de GUI de WEKA para este filtro que ilustra la **Fig.7**, son los mismos que implementa *NewDistSubsample*, comentados previamente.

NewDistSMOTE es la clase que implementa el método de remuestreo basado en SMOTE que, su vez, extiende la clase del filtro SMOTE que ya existía en WEKA, la cual, si bien no forma parte de la instalación por defecto de WEKA, forma parte del package SMOTE que se puede instalar con ayuda del Package Manager de la propia aplicación. *NewDistSMOTE* implementa las funciones necesarias para hacerlo compatible con la inferfaz *NewDistSpecifiable*, introduciendo la funcionalidad que permite seleccionar los porcentajes de elementos de la clase minoritaria sobre la nueva distribución.

Los parámetros configurables de *NewDistSMOTE* pueden observarse en la **Fig.8** que refleja la ventana de configuración en el GUI de Weka para este filtro y básicamente son todos aquellos heredados de la clase extendida SMOTE, al que se añade el parámetro que define el porcentaje deseado de elementos de la clase minoritaria en la distribución remuestreada resultante, que es el parámetro clave en esta implementación. A destacar también que es configurable el número de vecinos más cercanos que usará SMOTE para crear el nuevo elemento sintético y que por defecto se sitúa en 5.

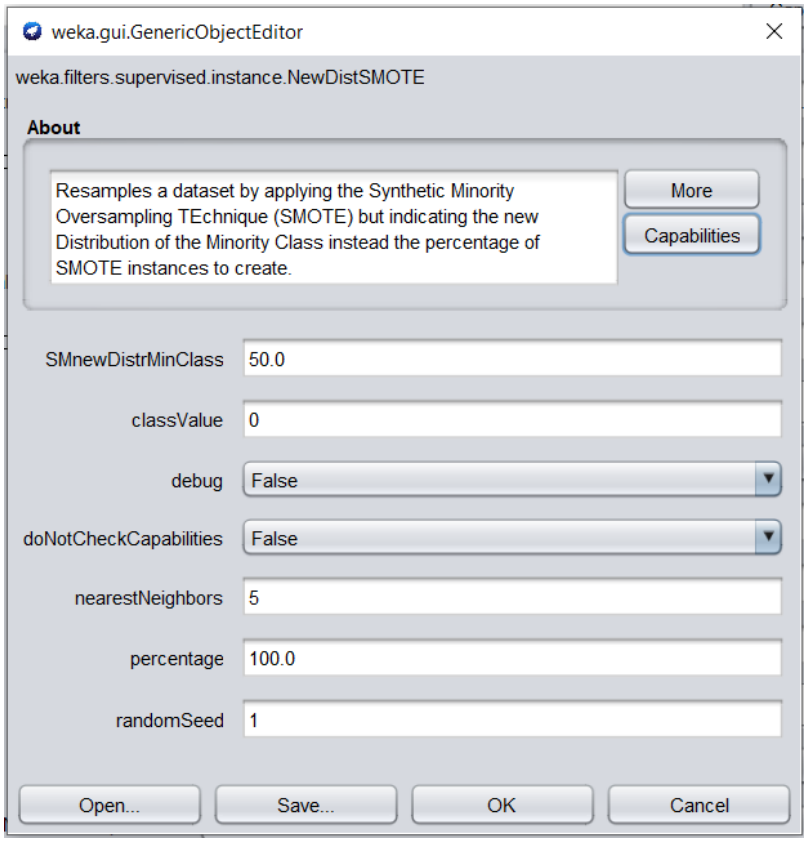

Fig. 8: *Ventana de configuración del filtro NewDistSMOTE en WEKA*

4.2 Definición del nuevo clasificador

NewDistribution2PhasesClassifier es la clase que contiene el código del clasificador que utilizaremos para encontrar la distribución óptima de la clase minoritaria en un remuestreo en función de la base de datos utilizada, el método de remuestreo que queramos utilizar, el clasificador que vayamos a utilizar y la métrica de evaluación que consideremos adecuada, entre otras variables.

De esta manera, las variables principales pueden configurarse en la ventana de configuración del GUI de WEKA para este clasificador, que se refleja en la **Fig.9**.

NewDistribution2PhasesClassifier es un clasificador de tipo "*meta*" (*metaclasificador*), ya que incluye combinaciones de otros algoritmos y también pertenece a la categoría de "*randomizable*", puesto que introduce la modelización de un comportamiento aleatorio a través del uso de funciones "*seed*", ya presentes en los filtros a los que llama como hemos revisado anteriormente.

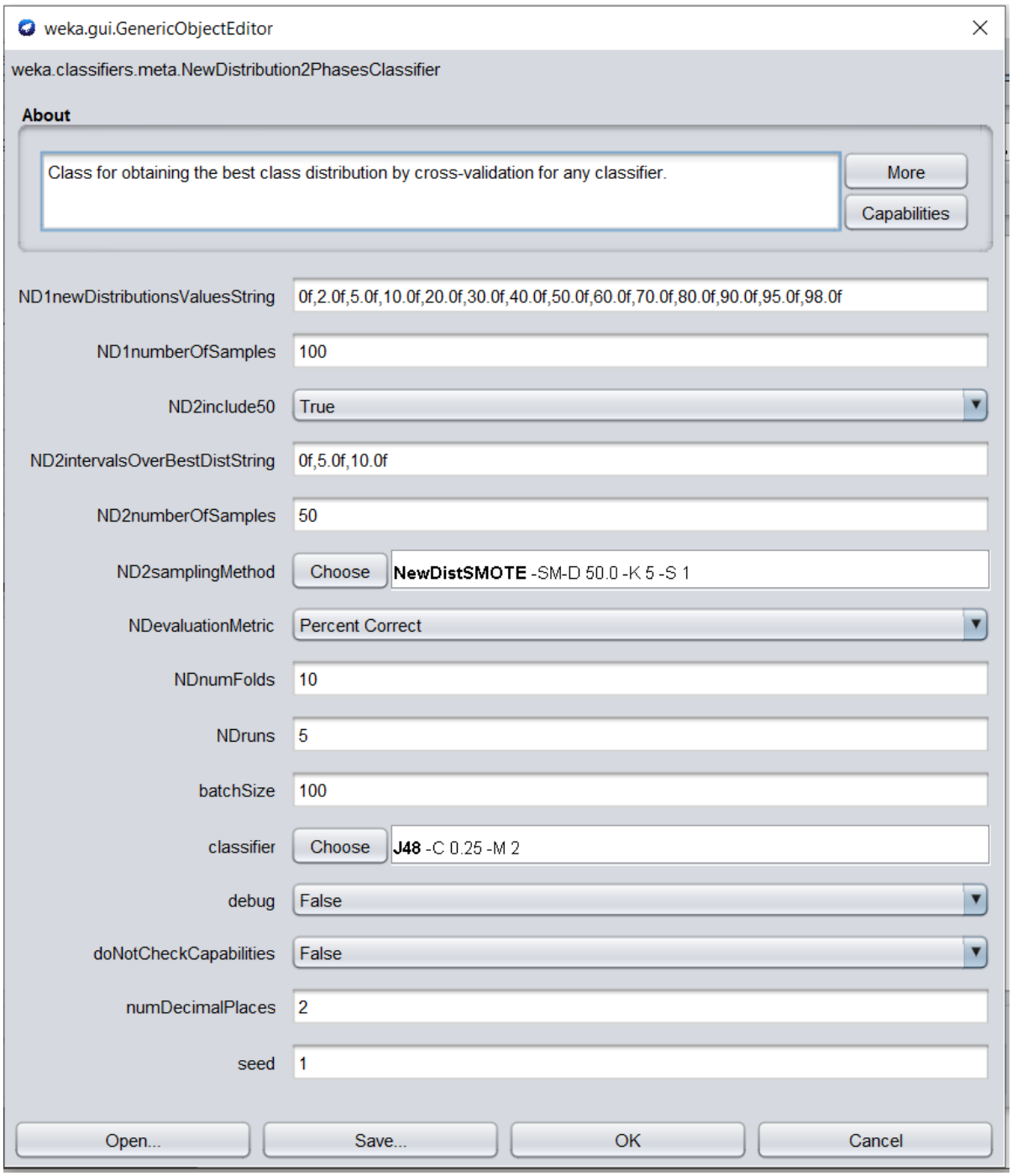

***Fig. 9**: Ventana de configuración del clasificador NewDistribution2PhasesClassifier en WEKA*

Como regla mnemotécnica general sobre las variables configurables en este clasificador podemos mencionar:

Si aparece *ND* en el nombre, hacen referencia a variables utilizadas en funciones exclusivamente desarrolladas para este clasificador. Si aparece *ND1*, estas variables afectarán a la fase 1 del algoritmo[4], es decir, en la determinación de un óptimo inicial a partir del barrido con un método de submuestreo aleatorio de tamaño mínimo. Si aparece *ND2*, las variables afectarán a la fase 2 del algoritmo, es decir, en la determinación de la distribución óptima a partir de un barrido local sobre el óptimo inicial de la fase 1 aplicando el método de remuestreo deseado. Finalmente, aquellas variables que únicamente contienen la referencia *ND* se corresponden con parámetros que afectan globalmente al código del clasificador, tanto en la fase 1 como en la fase 2 del mismo.

Teniendo en cuenta estas reglas de nomenclatura, podemos identificar mejor las variables configurables más importantes del clasificador.

Dentro de las variables que afectan a la fase 1 del algoritmo, tenemos:

- *ND1newDistributionsValuesString* hace referencia a la lista de los porcentajes de clase minoritaria objetivos para obtener las distribuciones submuestreadas que se compararán. Cabe destacar que la opción 0f hace referencia a la distribución original de la base de datos. Por defecto, aparece el barrido completo que vamos a usar en nuestra propuesta de investigación, incluyendo la distribución original.

- *ND1numberOfSamples* indica el número de submuestras que el algoritmo va a generar para cada uno de los porcentajes de distribución de clase minoritaria definidos en *ND1newDistributionsValuesString.* Por defecto, toma el valor 100 que es el que utilizaremos en nuestra propuesta de investigación.

Dentro de las variables que afectan a la fase 2 del algoritmo, tenemos:

- *ND2intervalsOverBestDistString* hace referencia al intervalo porcentual relativo al óptimo encontrado en la fase 1 que definirán los porcentajes de clase minoritaria objetivos para obtener las distribuciones remuestreadas que se compararán en la fase 2. El valor por defecto es una lista (0f, 5.0f, 10.0f) que incluirá el análisis de las distribuciones del óptimo de fase 1 y las distribuciones con porcentaje de clase minoritaria de +10%, +5%, -5% y -10% respecto al porcentaje óptimo de fase 1, siempre que estas queden en el rango (0,100), que son los intervalos que vamos a usar en nuestra propuesta de investigación. Hay que resaltar que, incluso aunque no se explicitara el valor 0f en la lista, el óptimo de la fase 1 siempre se va a incluir en la fase 2 del algoritmo.

[4] Para más detalles de las dos fases del algoritmo, comprobar la descripción del mismo en el apartado relativo a la propuesta de investigación.

- *ND2include50* permite introducir una distribución balanceada al 50% de la clase minoritaria incluso si está fuera del intervalo de *ND2intervalsOverBestDistString*, lo que permite comparar la distribución perfectamente balanceada con el proceso basado en el barrido y búsqueda refinada local.

- *ND2numberOfSamples* indica el número de submuestras que el algoritmo va a generar en la fase 2 para cada uno de los porcentajes de distribución de clase minoritaria definidos a partir de *ND2intervalsOverBestDistString*. Por defecto, toma el valor 50 que es el que utilizaremos en nuestra propuesta de investigación.

- *ND2samplingMethod* permite seleccionar el método de remuestreo, permitiendo elegir dentro de los filtros disponibles en WEKA. Sin embargo, hay que recordar que sólo están disponibles tres métodos de remuestreo que se corresponden con aquellos filtros que siguen la interfaz *NewSpecifiable*. Estos filtros se pueden encontrar bajo la carpeta correspondiente a *supervised/instance* y desde el propio GUI del clasificador permite abrir la ventana de configuración del filtro para configurarlo según la ventana presentada en el apartado anterior. Los filtros correspondientes a los tres métodos de remuestreo disponibles son:

    - *NewDistSubsample*
    - *NewDistOversample*
    - *NewDistSMOTE*

  NewDistSMOTE es la opción por defecto, aunque probaremos cuatro métodos de remuestreo, incluyendo opciones de las tres variantes en la propuesta de investigación.

Las principales variables que afectan globalmente al algoritmo son:

- *NDevaluationMetric* permite seleccionar la métrica bajo la que se evaluarán las distintas métricas, es decir, el criterio de evaluación sobre el que se busca el óptimo de la distribución. Las métricas disponibles son:

    - Percent Correct
    - Percent Incorrect
    - Kappa Statistic
    - TP Rate
    - FP Rate
    - Precision
    - Recall
    - F-Measure
    - ROC Area

  El valor por defecto es Percent Correct, aunque en la propuesta de investigación se usarán Kappa Statistic y ROC Area.

- *NDnumFolds* y *NDruns* son el número de particiones que se usarán para la validación cruzada y el número de veces que ejecutaremos esa validación cruzada. Por defecto tienen los valores de 10 y 5, que son los que usaremos en la propuesta de investigación.

- *Classifier* es la opción que permite elegir el clasificador sobre el que se aplican las distribuciones remuestreadas que calcula el algoritmo. Están disponibles todos los clasificadores implementados en WEKA y por defecto toma el valor de J48. Además, los clasificadores, al igual que sucede con los métodos de remuestreo, pueden ser configurados a través de sus propios parámetros. En la propuesta de investigación, los clasificadores que van a usarse son:

    - J48
    - NaiveBayes
    - IBk (con k=1 y k=3)
    - PART
    - JRIP

Teniendo en cuenta estos parámetros del clasificador, para los que nos hemos apoyado en la ventana GUI de configuración, vamos a revisar ahora el algoritmo implementado. El pseudocódigo asociado a este algoritmo se presenta en el **Algoritmo 1** que se describe a continuación, donde pueden mapearse fácilmente los inputs que requiere el algoritmo con los parámetros configurables que acabamos de repasar a partir de la ventana de configuración del clasificador en WEKA.

Algoritmo 1 NewDistribution2PhasesClassifier

Requiere:
S: data Set biclásico
E: métrica de Evaluación
M: Método de remuestreo
L: Lista de proporciones de remuestreo en fase 1
I: lista de Intervalos de remuestreo sobre el óptimo calculado de fase 1 para fase 2
R: número de Runs (veces que hacemos la validación cruzada)
F: número de Folds (particiones que se usan para la validación cruzada)
C: Clasificador base que se va a usar
B: Evaluación de la distribución balanceada a 50%
N1: Número de muestras en fase 1
N2: Número de muestras en fase 2

Fase 1
For #R
Reordenamos **S** aleatoriamente y estratificamos en #**F** folds
For #F
Define set de Entrenamiento
Define set de Test
For #L (cada elemento de la lista de distribución L)
Define la nueva distribución de la clase minoritaria según el valor del elemento
For #N1
Genera una nueva submuestra con la nueva distribución asignada
Construye un clasificador **C** con la nueva submuestra generada
Evalúa el clasificador construido según la métrica **E** y almacena el resultado
Calcula el valor medio asociado a la distribución
Calcula un vector con los valores asociados cada distribución de L
Encuentra la distribución óptima de Fase 1

Calcula la lista de proporciones de remuestreo de fase 2 (L2)
L2 -> distribución óptima de Fase 1 && +/- Intervalos definidos en **I**
If (B) then
L2 -> L2 && 50%

Fase 2
For #R
Reordenamos **S** aleatoriamente y estratificamos en #**F** folds
For #F
Define set de Entrenamiento
Define set de Test
For #L2 (cada elemento de la lista de distribución L2 creada después de Fase_1)
Comprobar que el **M** (método de remuestreo) predefinido es válido
Define la nueva distribución de la clase minoritaria según el valor del elemento y **M**
For #N2
Genera una nueva submuestra con la nueva distribución asignada y **M**
Construye un clasificador **C** con la nueva submuestra generada
Evalúa el clasificador construido según la métrica **E** y almacena el resultado
Calcula el valor medio asociado a la distribución
Calcula un vector con los valores asociados cada distribución de L2
Encuentra la distribución óptima de Fase 2 (distribución óptima final)

Generar una muestra con **M** para la distribución óptima calculada
Devolver el clasificador **C** aplicado a la muestra generada con el resultado evaluado según la métrica **E**

4.3 Información devuelta por el clasificador de dos fases implementado en WEKA

El algoritmo implementado sigue el proceso descrito en el **Algoritmo 1**, donde se especifican todas las entradas (*inputs*) bajo el epígrafe de "Requiere" y que se mencionan también en la descripción de las distintas ventanas de configuración del clasificador WEKA.

Además, al ejecutar el clasificador *NewDistribution2PhasesClassifier* en el *Explorer* de WEKA, éste ofrece una información de salida (*ouputs*). El código implementado incluye una sección para que incluya datos relevantes específicos de las pruebas que se han realizado en las dos fases del clasificador, además de la información que, por defecto, WEKA incluye como parte de un ejercicio del clasificador.

Así, la información que WEKA incluye por en el *informe de salida* es:

- El *esquema (scheme)*, que es la línea que incluye todas las opciones seleccionadas a través de las ventanas de configuración para el clasificador elegido y que sigue el formato que puede usarse para definir una programación de acciones en el *Experimenter* de WEKA, tal y como se puede comprobar en **Apéndices**, en el apartado *7.2*.

- El nombre de la base de datos que estamos utilizando en la prueba (*Relation*).

- Una información básica en cuanto a la base de datos que estamos utilizando en la prueba, que incluye el número de instancias que incluye la base de datos utilizada (*Instances*) y el número y listado de atributos incluidos en la base de datos (*Attributes*).

- El modo de test que se elija en el explorer (Validación cruzada, usar el conjunto de datos como muestra de entrenamiento, ...)

A continuación, aparecerá una información que dependerá del tipo de clasificador seleccionado dentro del *NewDistribution2PhasesClassifier*, en el ejemplo adjunto al haber elegido un clasificador de tipo JRIP, se devuelven las reglas declaradas y el número de reglas.

El siguiente bloque se corresponde con la **información específica de los ejercicios de remuestreo** programados en *NewDistribution2PhasesClassifier*, que se ha codificado específicamente para que forme parte del *informe de salida* y que se resalta en negrita en el ejemplo de salida adjunto en la **Tabla 5**. Esta información incluye:

- La distribución de clases de la base de datos original

- El criterio de bondad elegido para buscar la distribución óptima a través del método de remuestreo elegido

- Información de los métodos de remuestreo utilizados en fase 1 y fase 2, teniendo en cuenta que en fase 1 siempre se elegirá el submuestreo aleatorio de tamaño mínimo, la información más relevante es el tipo de remuestreo que seleccionamos para la fase 2

- Información detallada de los resultados obtenidos en la métrica elegida para cada uno de los porcentajes de distribución de definidos en la fase 1, al aplicar el submuestreo aleatorio de tamaño mínimo, donde puede observarse el valor del máximo y su distribución correspondiente.

- Información detallada de los resultados obtenidos en la métrica elegida para cada uno de los porcentajes de distribución de calculados en la fase 2, a partir del óptimo de la fase 1 y de los intervalos definidos para la fase 2. Se observa la lista testada y los resultados correspondientes al aplicar el remuestreo con su opción preseleccionada para la fase 2.

- Un resumen final que detalla el porcentaje de distribución óptimo encontrado en la fase 1 y el encontrado para la fase 2.

Esta información resulta útil cuando se quiere analizar en detalle el comportamiento de la localización de máximos y el comportamiento en general de las métricas en función del porcentaje de distribución de remuestreo, como se puede comprobar en los **Apéndices**, en el apartado *7.1 Análisis del comportamiento de métricas de bondad al variar la distribución de la clase minoritaria aplicando remuestreo.*

Finalmente se da información sobre el tiempo que ha llevado la construcción del modelo y un resumen estandarizado con los datos de evaluación de la prueba.

```
=== Run information ===

Scheme:     weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -
SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -
ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.JRip -num-decimal-places 4 -- -F 3 -N 2.0 -O 2 -S 1
Relation:    abalone9-18
Instances:   731
Attributes:  9
             Sex
             Length
             Diameter
             Height
             Whole_weight
             Shucked_weight
             Viscera_weight
             Shell_weight
             Class
Test mode:   evaluate on training data

=== Classifier model (full training set) ===

FilteredClassifier using weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S 1
on data filtered through weka.filters.supervised.instance.NewDistSMOTE -SM-D 35.0 -K 5 -S 1250

Classifier Model
JRIP rules:
===========

(Shell_weight >= 0.30552) and (Height >= 0.175984) and (Shucked_weight <= 0.564295) => Class=positive (163.0/8.0)
(Shell_weight >= 0.306933) and (Whole_weight >= 1.598992) and (Shucked_weight <= 0.807162) => Class=positive (45.0/0.0)
(Shell_weight >= 0.29) and (Length <= 0.58) and (Viscera_weight <= 0.197627) => Class=positive (64.0/3.0)
(Shell_weight >= 0.340059) and (Height <= 0.169324) and (Viscera_weight >= 0.3035) => Class=positive (23.0/3.0)
```

```
(Length <= 0.495777) and (Height >= 0.125) and (Length >= 0.481987) and (Height <= 0.14937) => Class=positive (30.0/2.0)
(Height >= 0.170642) and (Height <= 0.174883) => Class=positive (7.0/0.0)
(Height >= 0.125) and (Diameter <= 0.359983) and (Viscera_weight <= 0.107643) and (Length >= 0.434469) and (Length <= 0.471106)
=> Class=positive (13.0/1.0)
(Shell_weight >= 0.240264) and (Diameter <= 0.421963) and (Height <= 0.144125) => Class=positive (9.0/0.0)
(Diameter >= 0.57029) => Class=positive (6.0/2.0)
 => Class=negative (699.0/29.0)

Number of Rules : 10

The Original Distribution of minority class:  5.7456 %
Evaluation metric: Kappa Statistic
(Phase 1) weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0 filter used with 5 runs x 5 CV x 100 samples
(Phase 2) weka.filters.supervised.instance.NewDistSMOTE -SM-D 35.0 -K 5 -S 1250 filter used with 5 runs x 5 CV x 50 samples
(Phase 1) Distributions evaluated and related results:
Orig       2        5        10       20       30       40       50       60       70       80       90       95
           98
0          0        0        9.3662   16.1086  19.2114  19.1341  16.2279  12.239   7.3791   3.1448   1.2128   0.5914
0
(Phase 2) Distributions evaluated and related results:
20       25       30       35       40       50
42.4801  42.4574  42.4745  42.4969  42.4217  42.408
(Phase 1) The best New Distribution of minority class: 30    %
(Phase 2) The best Distribution of minority class for the selected resample method: 35    %

Time taken to build model: 91.13 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances       707          96.7168 %
Incorrectly Classified Instances      24           3.2832 %
Kappa statistic                  0.7379
Mean absolute error              0.0709
Root mean squared error          0.1723
Relative absolute error         64.8538 %
Root relative squared error     74.0213 %
Total Number of Instances          731

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.881    0.028    0.661      0.881   0.755      0.747    0.931     0.701     positive
               0.972    0.119    0.993      0.972   0.982      0.747    0.931     0.992     negative
Weighted Avg.  0.967    0.114    0.974      0.967   0.969      0.747    0.931     0.975

=== Confusion Matrix ===

  a   b   <-- classified as
 37   5 |   a = positive
 19 670 |   b = negative
```

***Tabla 5**: Ejemplo de informe de salida en WEKA Explorer de la implementación de NewDistribution2PhasesClassifier*

4.4 Diseño del experimento propuesto sobre la implementación del algoritmo en WEKA

Como se describe en el apartado *3.2 Procedimiento de investigación propuesto*, el diseño del experimento se basa en comparar el comportamiento de los distintos clasificadores al enfrentarse a las distintas bases de datos del set de experimentación, cuando tratamos de buscar la distribución óptima a través de distintas técnicas de remuestreo.

Se pretende, por tanto, comparar el comportamiento de un algoritmo clasificador bajo un mismo criterio de evaluación entre la muestra original, esto es, sin ningún tipo de remuestreo, y el conjunto de datos remuestreado con distintas técnicas.

Para ello utilizaremos un método de validación cruzada con 5 particiones que repetiremos 5 veces, buscando un compromiso entre el número de evaluaciones y el tiempo de computación, especialmente para aquellos conjuntos de datos más grandes y, como ya hemos adelantado, utilizaremos dos criterios de evaluación distintos para comprobar también que la propia decisión de este criterio va a condicionar los resultados, siendo éstas las métricas de bondad basadas en el estadístico Kappa y en el área bajo la curva ROC.

Para definir esta secuencia hay que partir de los clasificadores que queremos utilizar, de modo que inicialmente se lanza el clasificador directamente sobre la muestra original y después se llama al clasificador *NewDistribution2PhasesClassifier* desarrollado, tantas veces como alternativas de remuestreo se desean testar. En nuestro caso testaremos las opciones de submuestreo aleatorio con tamaño mínimo, submuestreo aleatorio con tamaño máximo, sobremuestreo aleatorio y SMOTE. Además, como vamos a probar dos métricas de evaluación, Kappa y área ROC, necesitaremos lanzar cada una de las alternativas de remuestreo dos veces, orientando cada una a maximizar cada una de las métricas de evaluación que queremos testar. Esto nos daría un total de 9 pruebas por cada clasificador, el original y 4 métodos de remuestreo orientados a cada una de las métricas de evaluación, que podemos resumir en la **Tabla 6.**

Algoritmos	**Método de Remuestreo**	**Métrica de Evaluación**
J48	Sin Remuestreo	Estadístico Kappa
NaiveBayes	Submuestreo aleatorio de tamaño mínimo	Area ROC
IBk (k=1)	Submuestreo aleatorio de tamaño máximo	
IBk (k=3)	Sobremuestreo aleatorio	
PART	SMOTE	
JRIP		

Tabla 6: *Alternativas testadas en la propuesta de investigación*

Este experimento se puede definir sobre el propio Experimenter de WEKA, una vez programado y testado el código del *NewDistribution2PhasesClassifier*. De esta manera, podemos definir

sobre la ventana de *Algorithms* la secuencia de algoritmos de clasificación que queremos comparar para automatizar el proceso. La programación completa con los distintos parámetros especificados para cada clasificador se incluye en los **Apéndices,** *7.2 Programación del experimento propuesto en la ventana Algorithms del Experimenter de WEKA.*

De esta manera, el experimento lanzará 54 clasificadores (6 originales y 4 remuestreos por cada una de las 2 métricas de evaluación por cada uno de esos 6 clasificadores originales) para cada una de las 33 bases de datos que forman parte del set de experimentación, lo que hace un total de 1782 clasificadores evaluados sobre el total de las bases de datos disponibles.

Por otro lado, desde el punto de vista de cada una de las bases de datos analizada, se lanzarán los comentados 54 clasificadores que serán evaluados 5 veces en validación cruzada de 5 particiones, por lo que se obtendrán 1350 evaluaciones para cada base de datos.

5 Resultados de la experimentación

Como parte fundamental de la experimentación, revisaremos inicialmente los resultados obtenidos, para cada uno de los clasificadores, al aplicar las distintas técnicas de remuestreo que hemos seleccionado en el diseño del experimento, de cada a intentar responder a las tres primeras preguntas que daban lugar a las hipótesis presentadas en el apartado *3.3 Hipótesis propuestas*:

- ¿Merece la pena aplicar el método de remuestreo en vez de utilizar el conjunto de datos original sin ningún tipo de remuestreo?
- ¿Hay algún método de remuestreo que muestre un mejor desempeño que los demás?
- ¿Está sujeta la elección del método de remuestreo a la dependencia de contexto?

Además, analizaremos si existen diferencias estadísticamente significativas en los datos de experimentación, más allá de la posible dependencia de contexto que podamos detectar.

En una segunda parte, como análisis adicional, trataremos de ver si hay alguna tendencia en cuanto al comportamiento de las distribuciones óptimas encontradas con las distintas alternativas de remuestreo, para tratar de responder a las preguntas que definían las hipótesis secundarias presentadas en el mismo apartado:

- ¿Existe alguna relación entre la distribución correspondiente al conjunto de datos original y a la distribución óptima?
- ¿Cuántas veces la distribución óptima resulta ser la distribución balanceada (50%)?
- ¿Cuántas veces el barrido de fase 2 modifica la distribución óptima calculada en el barrido de fase 1?

5.1 Análisis de la evaluación de resultados finales para cada clasificador tipo

Después de calcular los resultados de cada una de nuestras 33 bases de datos, con los resultados correspondientes a los 54 evaluaciones[5] contempladas en cada una, el siguiente paso es ejecutar un test estadístico para evaluar los resultados. Para ello utilizaremos la propia opción de análisis (pestaña *Analyse*) que implementa el módulo *Experimenter* de WEKA recargando los resultados que obtuvimos y salvamos en el experimento lanzado anteriormente.

Con este análisis vamos a establecer, para cada base de datos analizada y para cada uno de los clasificadores que estamos evaluando, supuesto que esta pudiera ser una variable prefijada por las características que queremos imponer a nuestro clasificador o porque queremos dar continuidad a un experimento donde ya fue prefijada esa variable, cuál es el método de remuestreo que mejor resultado obtuvo, incluyendo como referencia el caso de la distribución sin remuestreo, para identificar aquellos casos cuando ningún método de remuestreo logró superar el resultado aplicado sobre el conjunto de datos original.

En este caso, para la presentación de resultados, se abstraen las consideraciones respecto al valor del porcentaje de la distribución óptima que se obtiene en cada caso de remuestreo.

Presentaremos los resultados por separado para cada métrica utilizada, lo que nos permitirá comprobar de nuevo que, efectivamente la métrica condiciona para cada clasificador no sólo el resultado sino la técnica de remuestreo óptimo.

5.1.1 Métrica basada en el estadístico KAPPA

En la **Tabla 7** podemos observar los valores de las medias, aplicadas sobre los resultados obtenidos para cada una de las 33 bases de datos que constituyen el universo del experimento para el criterio de bondad Kappa sobre cada una de las alternativas de remuestreo testadas, incluyendo como referencia los resultados aplicados sobre el conjunto de datos original sin remuestreo alguno.

Los detalles de los resultados obtenidos para cada una de las bases de datos al aplicar un algoritmo de clasificación sobre las distintas alternativas de remuestreo pueden revisarse en los **Apéndices**, en el apartado *7.3 Resultados bajo el criterio de bondad Kappa de las 33 bases de datos bajo estudio para cada clasificador al aplicar cada variante de remuestreo.* En estas tablas de detalle, podemos observar de manera más clara el efecto de la dependencia de contexto al ver las diferencias que refleja cada una de las bases de datos.

[5] 6 algoritmos de clasificación, cada uno evaluado para la distribución sin remuestreo y 4 tipos de remuestreo (RANSUB Size Min, RANSUB Size Max, RANOVER, SMOTE) orientados hacia los criterios de bondad Kappa y Area ROC hacen un total de 6*(1+4*2) = 54 evaluaciones.

Algo Id.	Algoritmo de clasificación	Sin Remuestreo	RANSUB Size Min	RANSUB Size Max	RANOVER	SMOTE
1	J48	0.6096	0.5025	0.5942	0.6343	**0.6390**
2	NaiveBayes	0.5348	0.4918	0.5237	0.5362	**0.5438**
3	IBk1	0.6278	0.5300	0.6087	0.6279	**0.6325**
4	IBk3	0.6226	0.5396	0.6044	0.6181	**0.6353**
5	PART	0.5985	0.5007	0.5840	0.6281	**0.6308**
6	JRIP	0.6132	0.4849	0.5741	0.6296	**0.6328**

Tabla 7: *Media sobre las 33 bases de datos bajo estudio de los resultados obtenidos para cada clasificador al aplicar las distintas técnicas de remuestreo para la métrica Kappa*

En la **Tabla 7** podemos observar, resaltados en negrita, los mejores valores obtenidos para cada clasificador, analizando los valores medios obtenidos sobre todas las bases de datos testadas, donde se puede comprobar que los resultados recogidos para el remuestreo realizado con la técnica SMOTE son los que mejor desempeño demostraron evaluados bajo el criterio de bondad Kappa para todos los algoritmos de clasificación contemplados. La segunda opción en cuanto a desempeño es el remuestreo con RANOVER, salvo para el algoritmo IBk3. A resaltar tambien que el algoritmo de submuestreo aleatorio de tamaño mínimo, en valor medio, no muestra un desempeño mejor que el conjunto de datos original sin remuestreo para ninguno de los algoritmos de clasificación contemplados.

Si, alternativamente, tomamos de las tablas detalle recogidas en los **Apéndices**, en el apartado *7.3 Resultados bajo el criterio de bondad Kappa de las 33 bases de datos bajo estudio para cada clasificador al aplicar cada variante de remuestreo*, la selección para cada una de las bases de datos y para cada uno de los algoritmos de clasificación contemplados, el mejor resultado y el método de remuestreo asociado al mismo, obtenemos la **Tabla 8.** En esta tabla, en caso de empate con la muestra correspondiente al conjunto de datos sin remuestreo (que aparece etiquetado como Remuestreo "NO"), se elige ésta como preferente para mostrar que la aplicación de la técnica de remuestreo, que supone un coste adicional, no consigue mejora alguna. En esta tabla, para mejorar la visualización, etiquetamos a la opción de submuestreo aleatorio de tamaño mínimo como *RANSUB_1* y a la de submuestreo aleatorio de tamaño máximo como *RANSUB_2*.

En la tabla podemos comprobar que algunas bases de datos específicas muestran valores extremos que hacen que la muestra original ya alcance valores insuperables por cualquier técnica de remuestreo bajo la métrica analizada.

Clasificador	J48		NaiveBayes		IBk1		IBk3		PART		JRIP	
Nombre BBDD	***Valor***	***Remuestreo***	***Valor***	***Remuestreo***	***Valor***	***Remuestreo***	***Valor***	***Remuestreo***	***Valor***	***Remuestreo***	***Valor***	***Remuestreo***
Abalone19	0.0356	RANOVER	0.0123	SMOTE	0.0282	RANSUB_2	0.0302	RANSUB_2	0.0563	RANOVER	0.0486	RANOVER
Abalone9vs18	0.43	SMOTE	0.2187	NO	0.4666	NO	0.3387	SMOTE	0.4633	RANOVER	0.4138	RANOVER
Ecoli0vs1	0.9679	NO	0.9474	SMOTE	0.938	SMOTE	0.9637	SMOTE	0.9679	SMOTE	0.9696	NO
Ecoli1	0.7161	RANOVER	0.6635	RANOVER	0.6601	SMOTE	0.7225	NO	0.7059	SMOTE	0.7347	RANOVER
Ecoli2	0.7584	RANOVER	0.806	SMOTE	0.7766	NO	0.8412	NO	0.7649	RANOVER	0.7525	RANOVER
Ecoli3	0.5876	SMOTE	0.5452	NO	0.5608	SMOTE	0.5539	SMOTE	0.553	SMOTE	0.612	SMOTE
Ecoli4	0.6856	SMOTE	0.7922	NO	0.8121	SMOTE	0.8389	NO	0.72	SMOTE	0.7138	SMOTE
Glass0	0.5651	SMOTE	0.345	RANOVER	0.6291	NO	0.5863	NO	0.5858	RANOVER	0.6051	SMOTE
Glass0123vs456	0.7978	NO	0.7347	RANOVER	0.8454	SMOTE	0.8519	SMOTE	0.7945	SMOTE	0.7817	SMOTE
Glass1	0.4261	RANOVER	0.2539	NO	0.561	SMOTE	0.5147	NO	0.4242	SMOTE	0.4165	RANOVER
Glass2	0.3458	SMOTE	0.082	RANSUB_1	0.2564	SMOTE	0.2372	SMOTE	0.2882	SMOTE	0.3709	SMOTE
Glass4	0.6418	RANOVER	0.3632	SMOTE	0.7101	SMOTE	0.6633	SMOTE	0.6745	SMOTE	0.578	SMOTE
Glass5	0.8863	RANOVER	0.6915	SMOTE	0.7196	NO	0.6726	RANOVER	0.7961	NO	0.6495	RANOVER
Glass6	0.8109	RANSUB_2	0.7913	RANSUB_2	0.8088	NO	0.7988	SMOTE	0.7997	RANSUB_2	0.7787	NO
Haberman	0.2665	SMOTE	0.2597	SMOTE	0.1009	SMOTE	0.1314	SMOTE	0.2322	SMOTE	0.2673	RANSUB_2
Iris0	0.9749	NO	1	NO	1	NO	1	NO	0.9749	NO	0.9937	RANSUB_2
New-thyroid1	0.8974	SMOTE	0.9345	RANSUB_2	0.9416	NO	0.9458	RANOVER	0.9003	SMOTE	0.9117	RANOVER
New-thyroid2	0.9019	SMOTE	0.9429	RANSUB_2	0.9398	NO	0.942	SMOTE	0.8872	NO	0.8695	SMOTE
Page-blocks0	0.8465	RANOVER	0.7034	SMOTE	0.7398	SMOTE	0.7435	NO	0.8344	RANSUB_2	0.8473	RANSUB_2
Pima	0.4124	SMOTE	0.4499	SMOTE	0.3509	RANSUB_2	0.4222	SMOTE	0.4152	SMOTE	0.4382	SMOTE
Segment0	0.9737	RANOVER	0.62	RANSUB_1	0.9848	NO	0.9801	SMOTE	0.9746	SMOTE	0.9791	SMOTE
Vehicle0	0.8489	NO	0.348	RANOVER	0.8332	NO	0.8203	NO	0.852	SMOTE	0.8397	RANOVER
Vehicle1	0.4059	SMOTE	0.3445	RANSUB_1	0.3495	SMOTE	0.3668	SMOTE	0.4035	SMOTE	0.3977	SMOTE
Vehicle2	0.8975	RANOVER	0.4914	SMOTE	0.885	NO	0.8659	NO	0.9005	RANOVER	0.9013	SMOTE
Vehicle3	0.3942	RANSUB_2	0.3404	RANSUB_2	0.3155	SMOTE	0.3647	SMOTE	0.4101	SMOTE	0.4014	RANOVER
Vowel0	0.9369	NO	0.6906	NO	1	NO	1	SMOTE	0.9116	NO	0.9083	NO
Wisconsin	0.8986	SMOTE	0.919	SMOTE	0.9326	RANSUB_2	0.9441	SMOTE	0.9007	RANOVER	0.9136	RANSUB_2
Yeast1	0.3989	SMOTE	0.378	RANOVER	0.3071	SMOTE	0.3303	SMOTE	0.3657	SMOTE	0.3739	SMOTE
Yeast2vs8	0.5554	RANOVER	0.587	NO	0.5349	NO	0.6473	NO	0.542	RANOVER	0.6189	NO
Yeast3	0.7578	RANSUB_2	0.7447	RANOVER	0.6465	SMOTE	0.7171	RANSUB_2	0.7446	SMOTE	0.752	RANSUB_2
Yeast4	0.3268	RANOVER	0.3111	RANOVER	0.3358	SMOTE	0.3817	RANSUB_2	0.3502	RANOVER	0.3663	NO
Yeast5	0.763	NO	0.6215	NO	0.6972	SMOTE	0.7232	NO	0.717	NO	0.7322	NO
Yeast6	0.4979	NO	0.4296	NO	0.4552	NO	0.549	NO	0.5077	NO	0.5325	NO

Tabla 8: *Resultado de la mejor técnica de remuestreo específica para cada clasificador en cada una de las bases de datos de estudio bajo el criterio de evaluación de la métrica KAPPA*

Revisando el número de veces que cada técnica de remuestreo alcanzó el óptimo en cada base de datos según el clasificador previamente determinado (**Fig. 10**), observamos que el submuestreo aleatorio de tamaño mínimo (etiquetado como RANSUB_1 en la figura referida) es el que menos éxitos obtuvo, tan solo 3 específicamente en el algoritmo Naive-Bayes y ninguna vez fue la mejor opción para el resto.

Por otro lado, el remuestreo basado en SMOTE es el que más veces logró dar con la distribución óptima dentro de las comparadas.

Llama la atención que, para los algoritmos de k vecinos, la distribución sin remuestreo fue para muchas bases de datos la mejor opción, sólo superado por SMOTE, mientras que los algoritmos de submuestreo o sobremuestreo aleatorio tuvieron muy pocos éxitos.

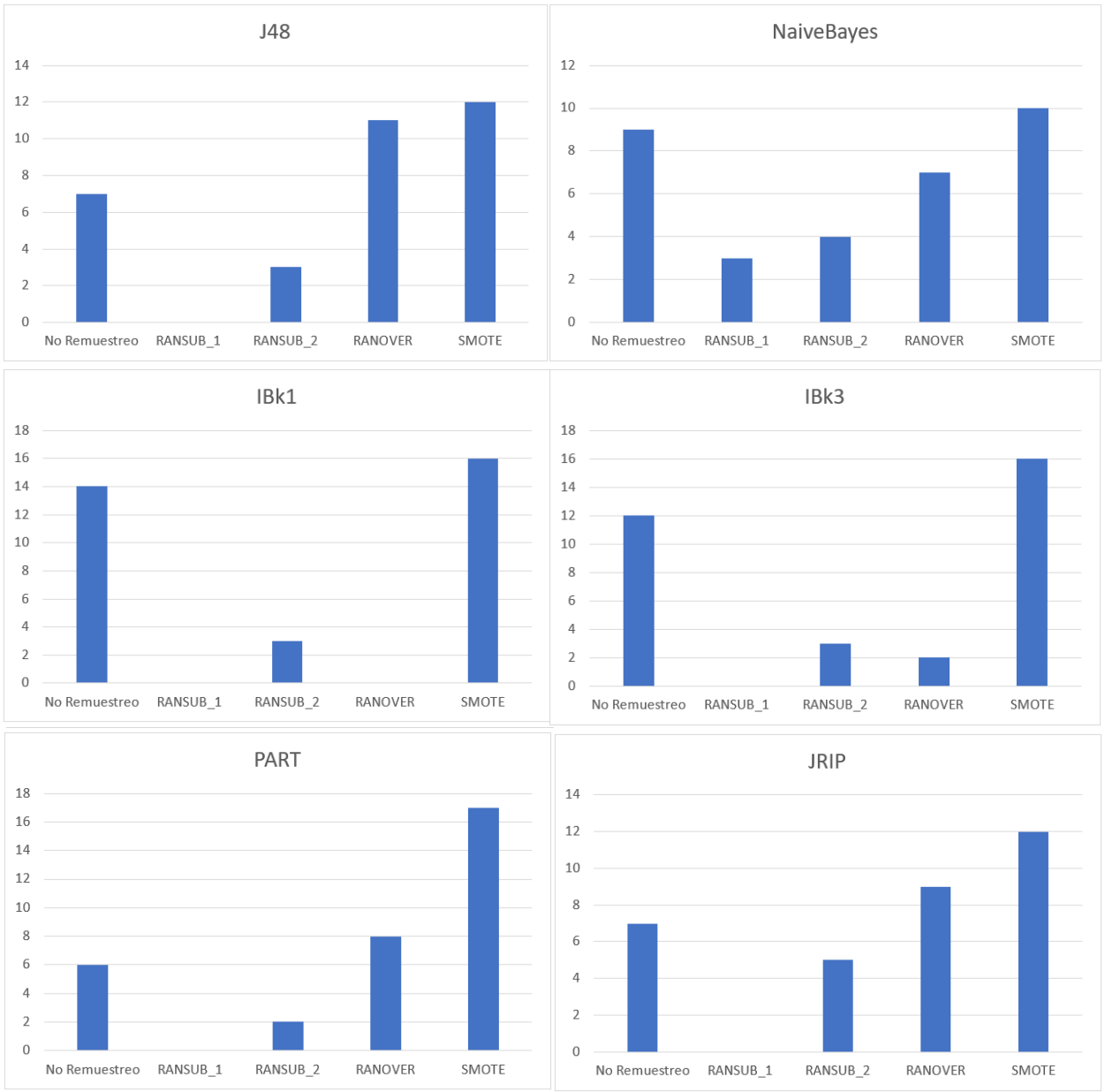

Fig. 10: *Número de bases de datos que obtuvieron su óptimo de distribución basado en cada una de las técnicas de remuestreo evaluadas, para cada uno de los clasificadores previamente determinados, bajo métrica KAPPA*

En definitiva, para la métrica KAPPA, las tablas y figuras confirman que no hay una técnica universal de remuestreo que sea en todos los casos la óptima, dependiendo de la estructura de la base de datos y del algoritmo clasificador usado. Si bien, el remuestreo SMOTE parece una opción con numerosos éxitos con cualquier variante de clasificador y la que mejor resultado medio muestra.

Además, se puede observar que hay casos, como en el clasificador Naive Bayes, las dos variantes del IBk o el PART en los que, cuando revisamos en cuántas bases de datos obtiene el mejor resultado cada clasificador, la opción de usar el clasificador sobre el conjunto de datos original sin remuestro es la segunda mejor, sin embargo, cuando revisamos la media de los resultados sobre todas las bases de datos, RANOVER es la segunda mejor opción[6]. Esto indicaría que, pese a que es habitualmente superado por SMOTE, los remuestreos realizados con RANOVER tambien presentan mejoras respecto a la aplicación sobre el conjunto de datos sin remuestreo, sobre aquellas bases de datos donde esta opción no se presenta como la mejor.

5.1.2 Métrica basada en al área bajo la curva ROC

En la **Tabla 9** podemos observar los valores de las medias, aplicadas sobre los resultados obtenidos para cada una de las 33 bases de datos que constituyen el universo del experimento, esta vez para el criterio de bondad del área sobre la curva ROC, sobre cada una de las alternativas de remuestreo testadas, incluyendo como referencia los resultados aplicados sobre el conjunto de datos original sin remuestreo alguno.

Del mismo modo que en el caso de la métrica Kappa, los detalles de los resultados obtenidos para cada una de las bases de datos al aplicar un algoritmo de clasificación sobre las distintas alternativas de remuestreo pueden revisarse en los **Apéndices**, en el apartado 7.4 *Resultados bajo el criterio de bondad del área bajo la curva ROC de las 33 bases de datos bajo estudio para cada clasificador al aplicar cada variante de remuestreo.* De nuevo, en esas tablas de detalle, podemos observar de manera exhaustiva el comportamiento y las variaciones en cada una de las bases de datos del universo experimental.

Algo Id.	Algoritmo de clasificación	Sin Remuestreo	RANSUB Size Min	RANSUB Size Max	RANOVER	SMOTE
1	J48	0.8229	0.8263	0.8493	0.8303	**0.8497**
2	NaiveBayes	**0.8738**	0.8627	0.8712	0.8737	0.8715
3	IBk1	0.8166	0.8255	0.8459	0.8162	**0.8490**
4	IBk3	0.8646	0.8638	0.8835	0.8636	**0.8859**
5	PART	0.8426	0.8306	0.8534	0.8353	**0.8639**
6	JRIP	0.8081	0.8072	0.8464	0.8355	**0.8546**

***Tabla 9**: Media sobre las 33 bases de datos bajo estudio de los resultados obtenidos para cada clasificador al aplicar las distintas técnicas de remuestreo para la métrica del área bajo la curva ROC*

[6] SMOTE es la mejor opción para estos algoritmos de clasificación, tanto a la hora de comparar los resultados de las medias de las 33 bases de datos, como al revisar el número de veces entre todas las bases de datos en los que la técnica de remuestreo correspondiente obtuvo el mejor valor.

Revisando la tabla, podemos observar un comportamiento particular para el algoritmo de clasificación de Naive Bayes, donde la mejor opción, si atendemos al comportamiento medio, parece ser la aplicación directa sobre el conjunto de datos original sin remuestreo, siendo la segunda opción el remuestreo con RANOVER.

Sin embargo, para el resto de los algoritmos de clasificación, vemos que el resultado medio sobre el conjunto de las bases de datos que mejor desempeño presenta es la opción de remuestreo SMOTE, siendo la segunda mejor opción el submuestreo aleatorio de tamaño máximo (RANSUB Size Max), lo cuál es, a su vez, una diferencia sustancial con la tabla observada para la métrica Kappa, donde la segunda opción era el remuestreo con RANOVER.

Otra diferencia respecto a la tabla observada cuando usábamos Kappa como criterio de bondad, es que, en este caso, hay algunos algoritmos de clasificación que, en la media agregada, llegan a presentar mejor desempeño aplicando la técnica de submuestreo aleatorio de tamaño mínimo (RANSUB Size Min), lo que no ocurría nunca con la otra métrica.

Esto confirma que el criterio de bondad es una variable que influye en el contexto a la hora de la selección de las mejores opciones para optimizar el resultado de un clasificador.

En la **Tabla 10** se presentan los resultados de cada base de datos del conjunto de experimentación para el método de remuestreo óptimo utilizado en cada uno de los clasificadores evaluados bajo el criterio del área bajo la curva ROC.

Se puede observar, comparando la **Tabla 8** y la **Tabla 10** que, más allá de los resultados, la opción óptima de remuestreo para cada base de datos varía al elegir una métrica distinta, lo que se confirma también revisando la tabla de los resultados medios. Así, para cada criterio de bondad, hay una técnica de remuestreo óptima distinta, dependiendo también de las características del propio set de datos, reforzando la idea de la dependencia del contexto.

Clasificador	J48		NaiveBayes		IBk1		IBk3		PART		JRIP	
Nombre BBDD	***Valor***	***Remuestreo***	***Valor***	***Remuestreo***	***Valor***	***Remuestreo***	***Valor***	***Remuestreo***	***Valor***	***Remuestreo***	***Valor***	***Remuestreo***
Abalone19	0.6561	RANSUB_1	0.6952	RANSUB_2	0.6423	RANSUB_2	0.6742	RANSUB_2	0.6554	RANSUB_2	0.7088	RANSUB_2
Abalone9vs18	0.7257	RANSUB_2	0.7476	SMOTE	0.7512	SMOTE	0.7797	SMOTE	0.7705	SMOTE	0.7527	RANOVER
Ecoli0vs1	0.9829	NO	0.9965	NO	0.9707	RANOVER	0.9867	RANSUB_1	0.9829	RANSUB_2	0.9807	NO
Ecoli1	0.9152	NO	0.9246	RANSUB_2	0.8658	SMOTE	0.9318	RANSUB_2	0.9201	NO	0.8995	RANOVER
Ecoli2	0.8795	RANOVER	0.9499	SMOTE	0.8967	RANOVER	0.9488	NO	0.891	RANSUB_2	0.8711	NO
Ecoli3	0.8569	RANSUB_2	0.9354	SMOTE	0.8536	SMOTE	0.9118	RANSUB_1	0.8826	SMOTE	0.8489	RANSUB_2
Ecoli4	0.8882	RANSUB_2	0.9935	RANOVER	0.9394	RANSUB_1	0.9903	RANSUB_2	0.9014	RANSUB_2	0.8986	NO
Glass0	0.8064	SMOTE	0.7644	SMOTE	0.825	SMOTE	0.8669	SMOTE	0.8314	NO	0.8237	RANOVER
Glass0123vs456	0.8955	SMOTE	0.9606	SMOTE	0.9334	SMOTE	0.9642	SMOTE	0.9221	SMOTE	0.9177	SMOTE
Glass1	0.7321	RANOVER	0.68	NO	0.7856	SMOTE	0.8331	RANSUB_2	0.7234	RANOVER	0.7347	SMOTE
Glass2	0.7326	SMOTE	0.7419	SMOTE	0.6781	SMOTE	0.7657	SMOTE	0.6933	SMOTE	0.7437	SMOTE
Glass4	0.8741	SMOTE	0.7848	RANSUB_2	0.9282	SMOTE	0.9579	SMOTE	0.8838	SMOTE	0.8724	SMOTE
Glass5	0.9846	NO	0.9663	NO	0.9183	SMOTE	0.9841	SMOTE	0.9846	NO	0.9168	RANOVER
Glass6	0.921	RANSUB_2	0.931	RANSUB_2	0.9145	SMOTE	0.9285	SMOTE	0.9156	RANSUB_2	0.9316	SMOTE
Haberman	0.6302	RANSUB_2	0.6444	RANSUB_1	0.5701	SMOTE	0.6043	NO	0.6476	SMOTE	0.6654	RANSUB_2
Iris0	0.984	NO	1	NO	1	NO	1	NO	0.984	NO	0.996	RANSUB_2
New-thyroid1	0.9595	SMOTE	0.9998	NO	0.9821	RANSUB_2	0.9971	NO	0.955	SMOTE	0.9648	SMOTE
New-thyroid2	0.9538	SMOTE	1	NO	0.9877	SMOTE	0.9955	RANOVER	0.9448	SMOTE	0.9561	SMOTE
Page-blocks0	0.9504	SMOTE	0.9731	NO	0.8989	RANSUB_2	0.946	RANSUB_2	0.9725	SMOTE	0.9634	SMOTE
Pima	0.7465	RANSUB_2	0.815	SMOTE	0.6881	RANSUB_2	0.7598	NO	0.7786	NO	0.7386	RANOVER
Segment0	0.9917	SMOTE	0.9867	RANSUB_2	0.9935	NO	0.9967	RANSUB_2	0.9913	SMOTE	0.9934	RANOVER
Vehicle0	0.9456	NO	0.8156	SMOTE	0.9399	SMOTE	0.9804	RANSUB_2	0.9503	NO	0.9422	SMOTE
Vehicle1	0.7345	RANSUB_2	0.7335	RANSUB_1	0.7118	SMOTE	0.775	SMOTE	0.7695	RANSUB_2	0.7681	SMOTE
Vehicle2	0.9566	SMOTE	0.8654	SMOTE	0.9554	SMOTE	0.9821	SMOTE	0.9637	SMOTE	0.9595	SMOTE
Vehicle3	0.742	SMOTE	0.722	SMOTE	0.6865	SMOTE	0.7629	SMOTE	0.788	SMOTE	0.7696	SMOTE
Vowel0	0.966	NO	0.9802	RANOVER	1	NO	1	NO	0.966	SMOTE	0.9734	SMOTE
Wisconsin	0.9617	RANSUB_2	0.9853	RANOVER	0.98	RANOVER	0.9885	RANOVER	0.9642	SMOTE	0.9668	SMOTE
Yeast1	0.7226	RANSUB_2	0.7804	NO	0.6651	SMOTE	0.7308	SMOTE	0.7663	NO	0.7206	SMOTE
Yeast2vs8	0.8001	SMOTE	0.8152	RANOVER	0.7699	SMOTE	0.8309	SMOTE	0.7932	RANOVER	0.7962	RANOVER
Yeast3	0.9212	SMOTE	0.9637	NO	0.8706	SMOTE	0.9374	RANSUB_2	0.9402	NO	0.9343	SMOTE
Yeast4	0.8304	RANSUB_2	0.863	RANOVER	0.7879	SMOTE	0.8676	RANSUB_1	0.8308	SMOTE	0.8519	RANSUB_2
Yeast5	0.9567	RANSUB_2	0.9865	NO	0.961	RANSUB_2	0.9858	RANSUB_2	0.9575	RANSUB_2	0.947	RANSUB_2
Yeast6	0.8469	RANSUB_1	0.944	SMOTE	0.8336	RANSUB_2	0.8955	RANSUB_1	0.8307	RANSUB_1	0.8612	RANSUB_2

Tabla 10: *Resultado de la mejor técnica de remuestreo específica para cada clasificador en cada una de las bases de datos de estudio bajo el criterio de evaluación de la métrica área ROC.*

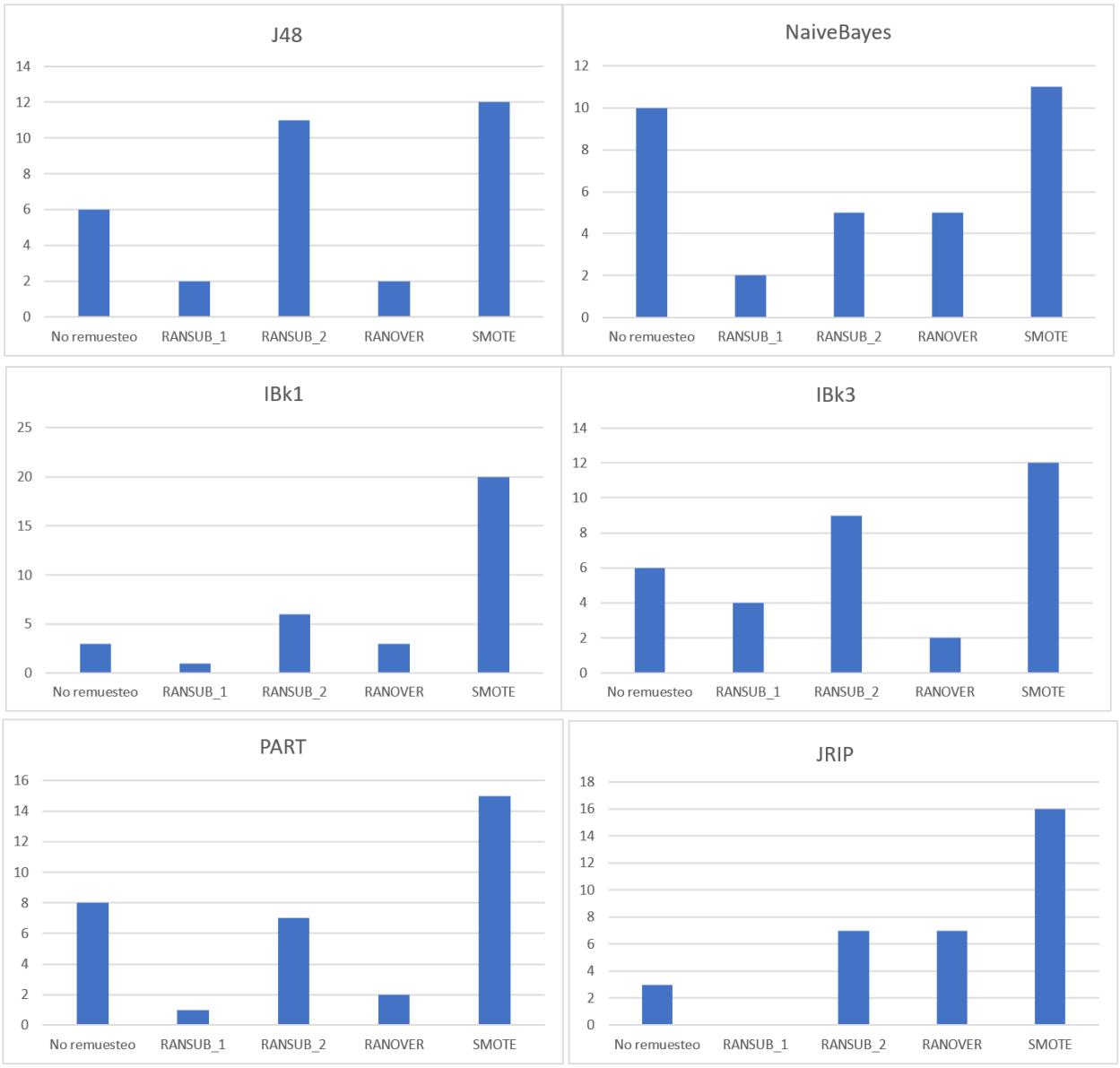

Fig. 11: *Número de bases de datos que obtuvieron su óptimo de distribución basado en cada una de las técnicas de remuestreo evaluadas, para cada uno de los clasificadores previamente determinados, bajo métrica de área ROC*

Revisando la **Fig.11** y estableciendo la comparación con la **Fig.10** para determinar las diferencias que impone la métrica elegida, podemos observar algunos puntos comunes, como que SMOTE es de nuevo la técnica de remuestreo que más veces da con el óptimo de la base de datos dentro de cada uno de los clasificadores elegidos.

Por un lado, la técnica del submuestreo aleatorio de tamaño mínimo (etiquetado en **Fig.11** también como RANSUB_1) es de nuevo el que menos veces tiene éxito, aunque, excepto para el JRIP, siempre logra algún caso de éxito a diferencia de lo observado en la métrica KAPPA. Por otro lado, para la métrica de área bajo la curva ROC, el submuestreo aleatorio de tamaño

máximo es la segunda técnica de remuestreo en número de éxitos, relegando el sobremuestreo aleatorio al tercer lugar, a diferencia de lo observado con la mérica KAPPA.

En definitiva, tal y como se comentó para la métrica KAPPA, las tablas y figuras confirman que para la métrica basada en el área bajo la curva ROC tampoco hay una técnica universal de remuestreo que sea en todos los casos la óptima, dependiendo de la estructura de la base de datos y del algoritmo clasificador usado. Además, podemos añadir por la comparación de ambos criterios de bondad, que la selección de la métrica es otra variable de contexto que condiciona el óptimo.

5.2 Diferencias estadísticamente significativas

En el apartado anterior se presenta el análisis de los valores medios obtenidos para las 33 bases de datos de nuestro universo muestral. Asimismo, se presentan los resultados de la mejor alternativa de remuestreo (incluyendo la opción de no remuestrear y utilizar la base de datos inicial) y en los **Apéndices,** en los apartados *7.3* y *7.4*, se incluyen las tablas completas de los resultados de los experimentos para cada base de datos.

Quedaría responder a la pregunta de si las diferencias encontradas son estadísticamente significativas. En una primera aproximación se puede usar el propio T-test implementado en Weka para establecer diferencias significativas al comparar las distintas bases de datos. De esta manera, podemos definir como referencia los resultados de cada base de datos original sin remuestreo y compararla con cada una de las opciones de remuestreo aplicadas para cada uno de los criterios de bondad, definiendo un intervalo de confianza en la propia implementacion de WEKA. Así, analizaríamos si, para cada base de datos, es decir, para cada problema determinado al que nos enfrentamos, encontraríamos una diferencia estadísticamente significativa al aplicar alguna de las técnicas de remuestreo.

Sin embargo, al realizar el experimento sobre un conjunto experimental de 33 bases de datos, parece más adecuado realizar el test de significancia estadística teniendo en cuenta los resultados en el conjunto de todas las bases de datos que pertenecen al experimento, en vez de aplicar un test específico para cada una de ellas.

En este sentido, respecto al método de comparación utilizado en al apartado anterior, esto es, la media de los resultados obtenidos en todas las bases de datos, se considera como una mera aproximación bruta a la caracterización del desempeño de un objeto de estudio. En algunos trabajos [48], se cuestiona si los errores en distintos dominios a los que pertenecen cada base de datos son realmente comparables y si por tanto la media resultante es significativa.

Demsar propone el uso de tests no paramétricos [49] para evaluar si existe una diferencia significativa y, dentro de las alternativas que propone, y teniendo en cuenta que el objetivo es tomar como referencia el caso de la aplicación del algoritmo de clasificación sobre el conjunto de datos original sin remuestreo, las comparaciones se harán pareadas siempre con esa referencia y por tanto usaremos el Test de los Rangos con signo de Wilcoxon, que ordenará las diferencias

de desempeño entre la referencia (la base de datos sin remuestreo) y cada una de las alternativas, ignorando los signos y comparará los valores asignados tras la ordenación para las diferencias positivas y negativas.

Utilizaremos un nivel de confianza del 95% (α=0.05) y, teniendo en cuenta que comparamos 33 bases de datos, podemos usar una aproximación del cálculo del valor *z*, asumiendo que se aproximará a una distribución tipo *normal* [49], donde *z* se calcula como:

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}}$$

Calculando estos valores para cada criterio de bondad y para cada algoritmo de clasificación, el valor *z* obtenido al comparar cada método de remuestreo con la base de datos original sin remuestreo se muestra en ***Tabla 11*** y ***Tabla 12,*** donde se resalta en negrita los casos con una mejora estadísticamente significativa y en cursiva y con el símbolo *(*)*, los casos de un empeoramiento estadísticamente significativo.

Criterio Bondad **Kappa**	α=0.05		Valor crit: -1.96	
Algoritmo de Clasificación	Método de remuestreo			
	RANSUB Min	**RANSUB Max**	**RANOVER**	**SMOTE**
J48	*-4.31(*)*	-1.78	-1.30	**-2.18**
NaiveBayes	*-2.69 (*)*	-0.11	-0.64	-0.86
IBk1	*-4.59 (*)*	*-2.45 (*)*	-0.01	-1.59
IBk3	*-4.41 (*)*	-1.52	-0.59	-1.59
PART	*-4.10 (*)*	-1.37	-1.64	**-2.73**
JRIP	*-4.98 (*)*	*-2.75 (*)*	-1.55	**-2.30**

Tabla 11: *Tabla de valores z del test de Rangos de Wilcoxon para Kappa*

Criterio Bondad **Area ROC**	α=0.05		Valor crit: -1.96	
Algoritmo de Clasificación	Método de remuestreo			
	RANSUB Min	**RANSUB Max**	**RANOVER**	**SMOTE**
J48	-0.68	**-2.79**	-0.28	**-3.61**
NaiveBayes	*-3.15 (*)*	*-2.81 (*)*	-0.75	-0.93
IBk1	-0.43	**-3.82**	-1.11	**-4.17**
IBk3	-1.23	**-2.17**	*-2.97 (*)*	**-3.22**
PART	*-2.83 (*)*	-0.96	*-2.18 (*)*	**-3.20**
JRIP	0.00	**-3.65**	**-4.1**	**-4.41**

Tabla 12*: Tabla de valores z del test de Rangos de Wilcoxon para el área bajo la curva ROC*

En general, se puede observar una cierta correlación, que cabría esperar, entre los valores de las medias presentados en el apartado *5.1 Análisis de la evaluación de resultados finales para cada clasificador tipo* (**Tabla 7** y **Tabla 9**) y las correspondientes salidas del test de los Rangos de Wilcoxon.

Como ya observábamos, el criterio de bondad bajo el que se evalúa tiene impacto en la caracterización del comportamiento de los remuestreos. En cualquier caso, hay algunas generalizaciones que pueden hacerse al método que hemos aplicado.

- El método propuesto obtiene resultados significativamente mejores cuando seleccionamos como alternativa de remuestreo SMOTE para el criterio de bondad del área bajo la curva ROC para todos los algoritmos de clasificación, excepto para el Naive Bayes, para el que no se dan resultados significativos.

- Si el criterio de bondad elegido es Kappa, el método propuesto con remuestreo SMOTE sigue dando diferencias estadísticamente significativas para J48, PART y JRIP, aunque para los algoritmos de clasificación de Naive Bayes, y IBk (kNN) con las variantes de k=1 y k=3 obtiene mejoras no significativas.

- La alternativa de seleccionar el algoritmo de submuestreo aleatorio de tamaño mínimo (RANSUB Min Size) para la segunda fase no es aconsejable, con esta aproximación se detectaron comportamientos significativamente peores a usar el conjunto de datos original sin remuestreo en todos los algoritmos de clasificación para el criterio de bondad Kappa y en Naive Bayes y PART para la métrica del área ROC bajo la curva.

- Por otro lado, para el criterio de bondad Kappa, el método RANOVER obtuvo sólo mejoras no significativas y el método RANSUB Max Size empeoramientos que, para los clasificadores JRIP y IBk (kNN) con k=1 llegaron a ser significativos.

- Para el criterio de bondad del área bajo la curva ROC, las alternativas de muestreo en segunda fase RANSUB Max Size y RANOVER muestran un comportamiento muy variable en función del algoritmo de clasificación al que se aplican.

Como recomendación general, el método SMOTE sería el más prometedor para aplicar en segunda fase para las dos métricas contempladas y en cualquier clasificador entre el conjunto evaluado. Del mismo modo, se debería evitar utilizar el método de RANSUB Size Min en la segunda fase ya que, pese a ser el método con menor carga computacional, tiende a empeorar los resultados obtenidos sobre la propia base de datos original.

5.3 Análisis de la evaluación de resultados óptimos con todos los grados de libertad

5.3.1 La combinación óptima de algoritmo de clasificación y método de remuestreo

Si, para cada una de las bases de datos analizadas, se permite seleccionar cualquier tipo de clasificador y método de remuestreo de cara a obtener los resultados de desempeño óptimos de cada uno de los dos criterios de bondad tenidos en cuenta, obtenemos la **Tabla 13** y **Tabla 14**

Al otorgar todos los grados de libertad (selección del clasificador y de la técnica de remuestreo) podemos comprobar que tales combinaciones son totalmente distintas para cada métrica y para cada base de datos, confirmando la hipótesis de la dependencia de contexto y la necesidad de un método flexible como el propuesto para poder encontrar el óptimo en cada problema al que nos enfrentemos, representado por cada conjunto de datos.

Criterio de evalución - Estadístico KAPPA			
Nombre BBDD	***Valor***	***Remuestreo***	***Clasificador***
Abalone19	0.0563	RANOVER	PART
Abalone9vs18	0.4666	NO	IBk1
Ecoli0vs1	0.9696	NO	JRIP
Ecoli1	0.7347	RANOVER	JRIP
Ecoli2	0.8412	NO	IBk3
Ecoli3	0.612	SMOTE	JRIP
Ecoli4	0.8389	NO	IBk3
Glass0	0.6291	NO	IBk1
Glass0123vs456	0.8519	SMOTE	IBk3
Glass1	0.561	SMOTE	IBk1
Glass2	0.3709	SMOTE	JRIP
Glass4	0.7101	SMOTE	IBk1
Glass5	0.8863	RANOVER	J48
Glass6	0.8109	RANSUB_2	J48
Haberman	0.2673	RANSUB_2	JRIP
Iris0	1	NO	NaiveBayes
New-thyroid1	0.9458	RANOVER	IBk3
New-thyroid2	0.9429	RANSUB_2	NaiveBayes
Page-blocks0	0.8473	RANSUB_2	JRIP
Pima	0.4499	SMOTE	NaiveBayes
Segment0	0.9848	NO	IBk1
Vehicle0	0.852	SMOTE	PART
Vehicle1	0.4059	SMOTE	J48
Vehicle2	0.9013	SMOTE	JRIP
Vehicle3	0.4101	SMOTE	PART
Vowel0	1	NO	IBk1
Wisconsin	0.9441	SMOTE	IBk3
Yeast1	0.3989	SMOTE	J48
Yeast2vs8	0.6473	NO	IBk3
Yeast3	0.7578	RANSUB_2	J48
Yeast4	0.3817	RANSUB_2	IBk3
Yeast5	0.763	NO	J48
Yeast6	0.549	NO	IBk3

Tabla 13: *Resultado de la mejor combinación de clasificador y método de remuestreo para cada base de datos bajo la evaluación de métrica KAPPA*

Criterio de evalución -Area bajo ROC			
Nombre BBDD	***Valor***	***Remuestreo***	***Clasificador***
Abalone19	0.7088	RANSUB_2	JRIP
Abalone9vs18	0.7797	SMOTE	IBk3
Ecoli0vs1	0.9965	NO	NaiveBayes
Ecoli1	0.9318	RANSUB_2	IBk3
Ecoli2	0.9499	SMOTE	NaiveBayes
Ecoli3	0.9354	SMOTE	NaiveBayes
Ecoli4	0.9935	RANOVER	NaiveBayes
Glass0	0.8669	SMOTE	IBk3
Glass0123vs456	0.9642	SMOTE	IBk3
Glass1	0.8331	RANSUB_2	IBk3
Glass2	0.7657	SMOTE	IBk3
Glass4	0.9579	SMOTE	IBk3
Glass5	0.9846	NO	J48
Glass6	0.9316	SMOTE	JRIP
Haberman	0.6654	RANSUB_2	JRIP
Iris0	1	NO	NaiveBayes
New-thyroid1	0.9998	NO	NaiveBayes
New-thyroid2	1	NO	NaiveBayes
Page-blocks0	0.9731	NO	NaiveBayes
Pima	0.815	SMOTE	NaiveBayes
Segment0	0.9967	RANSUB_2	IBk3
Vehicle0	0.9804	RANSUB_2	IBk3
Vehicle1	0.775	SMOTE	IBk3
Vehicle2	0.9821	SMOTE	IBk3
Vehicle3	0.788	SMOTE	PART
Vowel0	1	NO	IBk1
Wisconsin	0.9885	RANOVER	IBk3
Yeast1	0.7804	NO	NaiveBayes
Yeast2vs8	0.8309	SMOTE	IBk3
Yeast3	0.9637	NO	NaiveBayes
Yeast4	0.8676	RANSUB_1	IBk3
Yeast5	0.9865	NO	NaiveBayes
Yeast6	0.944	SMOTE	NaiveBayes

Tabla 14: *Resultado de la mejor combinación de clasificador y método de remuestreo para cada base de datos bajo la evaluación del área bajo la curva ROC*

Si analizamos las figuras con los datos agregados de la **Tabla 13** y **Tabla 14**, registrando el número de casos en el que cada técnica de remuestreo obtuvo el valor óptimo global con independencia del clasificador asociado al mismo, obtenemos la **Fig. 12** y **Fig. 13.**

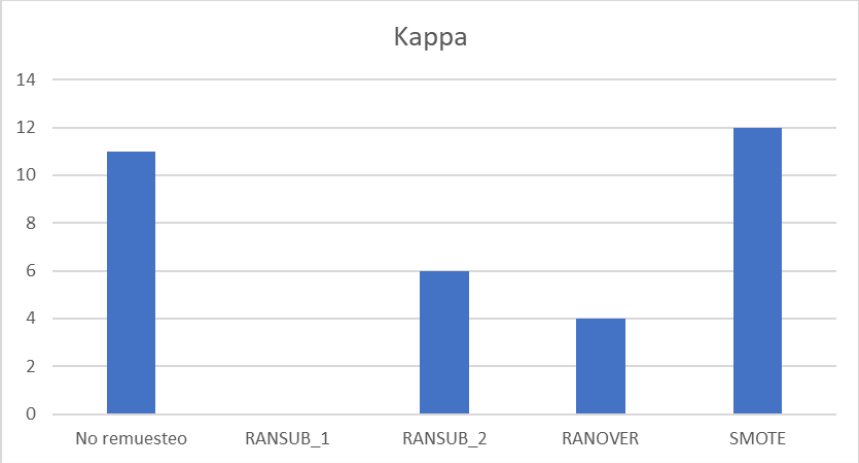

Fig. 12: *Número de bases de datos que obtuvieron su óptimo de distribución basado en cada una de las técnicas de remuestreo evaluadas, para el mejor caso de entre los clasificadores testados, bajo métrica KAPPA*

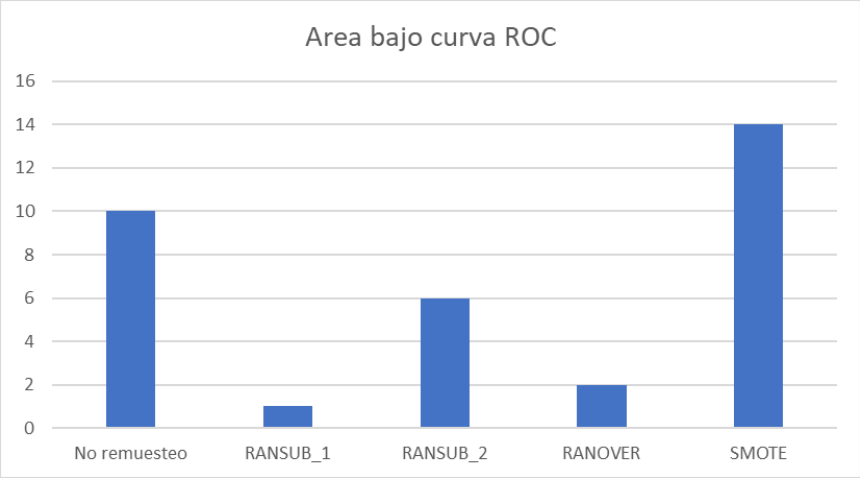

Fig. 13: *Número de bases de datos que obtuvieron su óptimo de distribución basado en cada una de las técnicas de remuestreo evaluadas, para el mejor caso de entre los clasificadores testados, bajo métrica de área ROC*

Revisando la **Fig. 12** y **Fig. 13.** podemos confirmar que para los óptimos globales, pudiendo elegir cualquier clasificador, SMOTE sigue siendo la técnica que más éxitos encuentra y el submuestreo aleatorio de tamaño mínimo la menos interesante, hasta el punto de no lograr ningún caso para la métrica KAPPA. Por otro lado, en este escenario en el que podemos elegir libremente el clasifcador, el método de submuestreo aleatorio de tamaño máximo se consolida

como la segunda mejor técnica de remuestreo por encima del sobremuestreo aleatorio. Estos patrones se repiten tanto para la métrica KAPPA como para la de ROC.

5.3.2 Consideraciones sobre el clasificador óptimo

En el análisis presentado inicialmente, se ponía el foco en revisar cuál era el método de remuestreo óptimo cuando el clasificador se ha fijado previamente, ya que suele ser el caso más habitual teniendo en cuenta las implicaciones que tiene la selección del clasificador, según las propias características del mismo.

Si nos enfrentáramos al problema opuesto, esto es, cuál sería el clasificador óptimo para una métrica concreta, insistiendo en la dependencia del contexto, ya sabemos que no existe un clasificador óptimo universal para cualquier base de datos. Pero más aún, a partir del análisis efectuado, podemos demostrar que, sobre una métrica prefijada, la propia técnica de remuestreo utilizada, puede hacer variar el clasificador óptimo a la hora de enfrentarnos a una base de datos concreta.

Así, a modo de ejemplo la **Fig.14** y **Fig.15** muestran el número de veces que cada tipo de clasificador es el óptimo en cada caso cuando tenemos libertad total para elegir el tipo de remuestreo, es decir, a partir de los datos de la **Tabla 13** y **Tabla 14**, y también la comparación con los resultados del clasificador que mejor resultados obtiene cuando se parte de las distribuciones originales sin remuestreo.

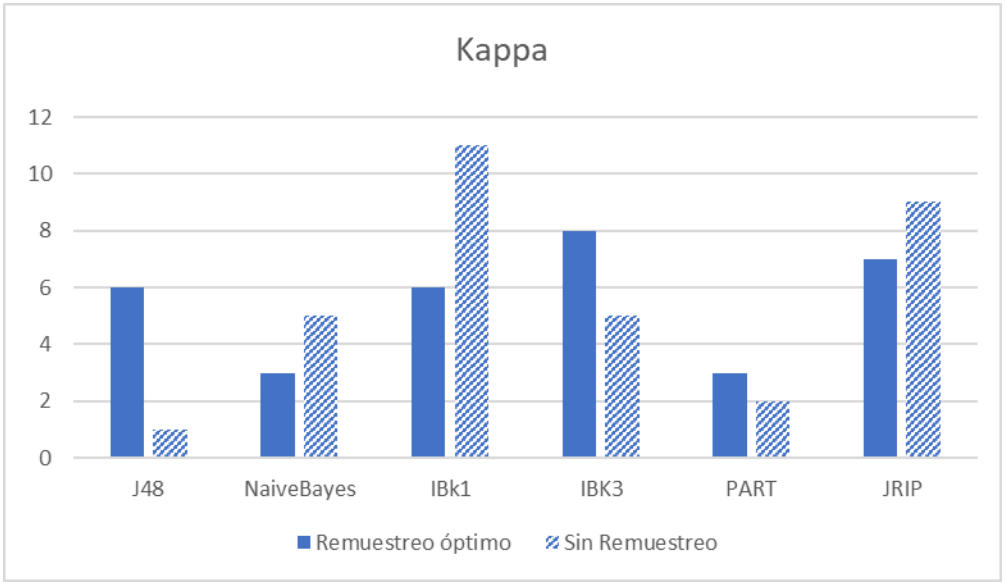

Fig. 14: *Número de bases de datos que obtuvieron su óptimo de para cada tipo de clasificador después del remuestreo para el mejor caso y resultado sobre las distribuciones originales, bajo métrica KAPPA*

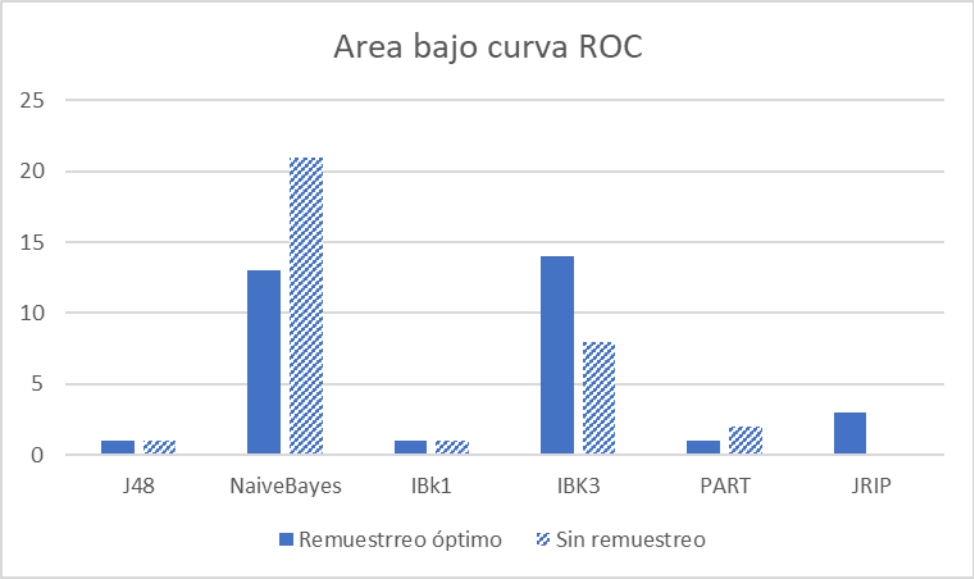

Fig. 15: *Número de bases de datos que obtuvieron su óptimo de para cada tipo de clasificador después del remuestreo para el mejor caso y resultado sobre las distribuciones originales, bajo métrica ROC*

Podemos observar que parecen detectarse clasificadores más adecuados para optimizar determinadas métricas, aspecto que se observa ya partiendo de las métricas originales y se mantiene después de aplicar las técnicas de remuestreo para encontrar la distribución óptima.

Además, se puede observar que **la aplicación de un remuestreo u otro puede modificar cuáles son los "clasificadores óptimos" para cada base de datos** de manera distinta, de modo que, cuando se incluye una fase de remuestreo, el propio tipo de remuestreo es una variable de contexto para encontrar el tipo de clasificador óptimo.

5.4 Análisis adicionales relativos a los resultados de los valores de las distribuciones óptimas en los remuestreos sobre las Bases de Datos disponibles

A partir de los resultados del experimento, extraemos la distribución óptima de la primera fase, aplicando la técnica de submuestreo aleatorio con mínimo tamaño de la submuestra, siendo este el tamaño de la clase minoritaria en la distribución original, con los porcentajes de distribución objetivo para el remuestreo del barrido de la lista predefinida al que se añade en cada caso el porcentaje de la distribución original, es decir, 14 objetivos de distribución de remuestreo: *Original, 2%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 95%, 98%*.

En la segunda fase tomamos la distribución óptima de la primera fase y probamos sobre una lista que puede llegar a tener hasta 6 elementos, el valor óptimo de la primera fase, un intervalo de 5%

y 10% arriba y abajo y la muestra del 50% si no estaba ya incluida. Es decir, si llamamos al porcentaje de la distribución óptima de Fase 1 OF1, la lista de objetivos de distribución de remuestreo de la Fase 2 es: *50%, OF1-10%, OF1-5%, OF1, OF1+5%, OF1+10%*

El valor que se obtendrá como distribución óptima va a tener una alta dependencia de la métrica que estemos usando y también del tipo de clasificador que hayamos elegido. Para el óptimo de la fase 2 también dependerá el tipo de remuestreo que hayamos elegido. Esto no sucede con el OF1 porque este es calculado siempre usando el submuestreo aleatorio de tamaño mínimo (RANSUB Min Size).

El *Experimenter* de WEKA proporciona un número de muestras total, sobre las 33 bases de datos objeto de análisis, de 825 evaluaciones en cada caso, al haber lanzado el experimento con una validación cruzada de 5 repeticiones y 5 subdivisiones, y puesto que en el código del clasificador *NewDistribution2PhasesClassifier*, tal y como se describe en el apartado *4.3 Información devuelta por el clasificador de dos fases implementado en WEKA,* se introdujeron las instrucciones para obtener información de qué valores óptimos se encuentran en cada fase (y en realidad los valores evaluados para cada distribución testada), podemos disponer de estos valores para analizar los resultados de las distribuciones óptimas obtenidas en cada caso.

Para cada una de las métricas de evaluación por separado (estadístico KAPPA y área bajo la curva ROC) presentamos los siguientes valores:

- Porcentaje de casos en los que el remuestreo óptimo es distinto al de la muestra original.
- Porcentaje de casos en los que el remuestreo óptimo en fase 2 mejora al conseguido en fase 1.
- Porcentaje de casos en los que el remuestreo óptimo en fase 2 es distinto a la distribución balanceada al 50%.

5.4.1 Métrica basada en el estadístico KAPPA

Revisando la primera cuestión, es decir, el **porcentaje de casos en los que el remuestreo óptimo es distinto al de la muestra original,** vemos que en general esto sucede en un alto porcentaje de veces, en la **Fig. 16** se puede observar que para los clasificadores basados en los k vecinos cercanos (k-NN implementado como IBk en WEKA), el porcentaje de casos en los que el óptimo es distinto al de la distribución original disminuye, mientras que en JRIP o PART el porcentaje se encuentra por encima del 99.5%. Por otro lado cuando remuestreamos en segunda fase con SMOTE o RANOVER hay un gran salto respecto al número de veces que la Fase 1 (mostrados en la gráfica con línea punteada) obtenía un valor distinto a la distribución original para los algoritmos de IBk.

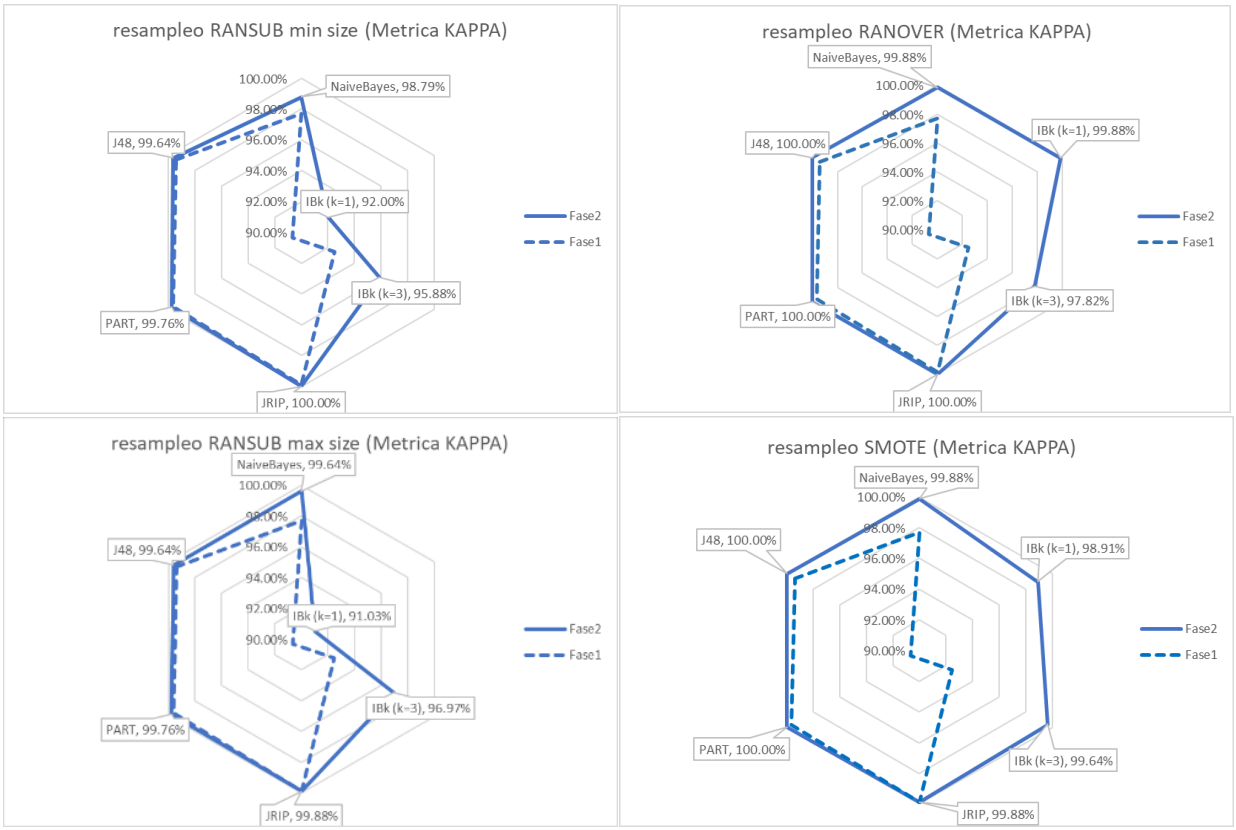

Fig. 16: *Porcentaje de veces en el que el óptimo de la distribución de clases es distinto a la distribución original bajo métrica del estadístico KAPPA*

En cualquier caso, los porcentajes que muestra la **Fig. 16** son elevados, obteniendo en el agregado de todos los clasificadores un valor del 96.40% ya para el remuestreo de fase 1 mejora a los resultados de la distribución original y encontrando para los agregados en fase 2 valores de 97.38%, para RANSUB Size Min, 97.68% para RANSUB Size Max, 99.60% para RANOVER y 99.72% para SMOTE, por lo que se puede comprobar que la distribución óptima en un remuestreo rara vez coincide con la distribución del conjunto de datos original.

A continuación, revisamos **el número de veces en las que el óptimo de la distribución de fase 2 cambia respecto al calculado en fase 1.**

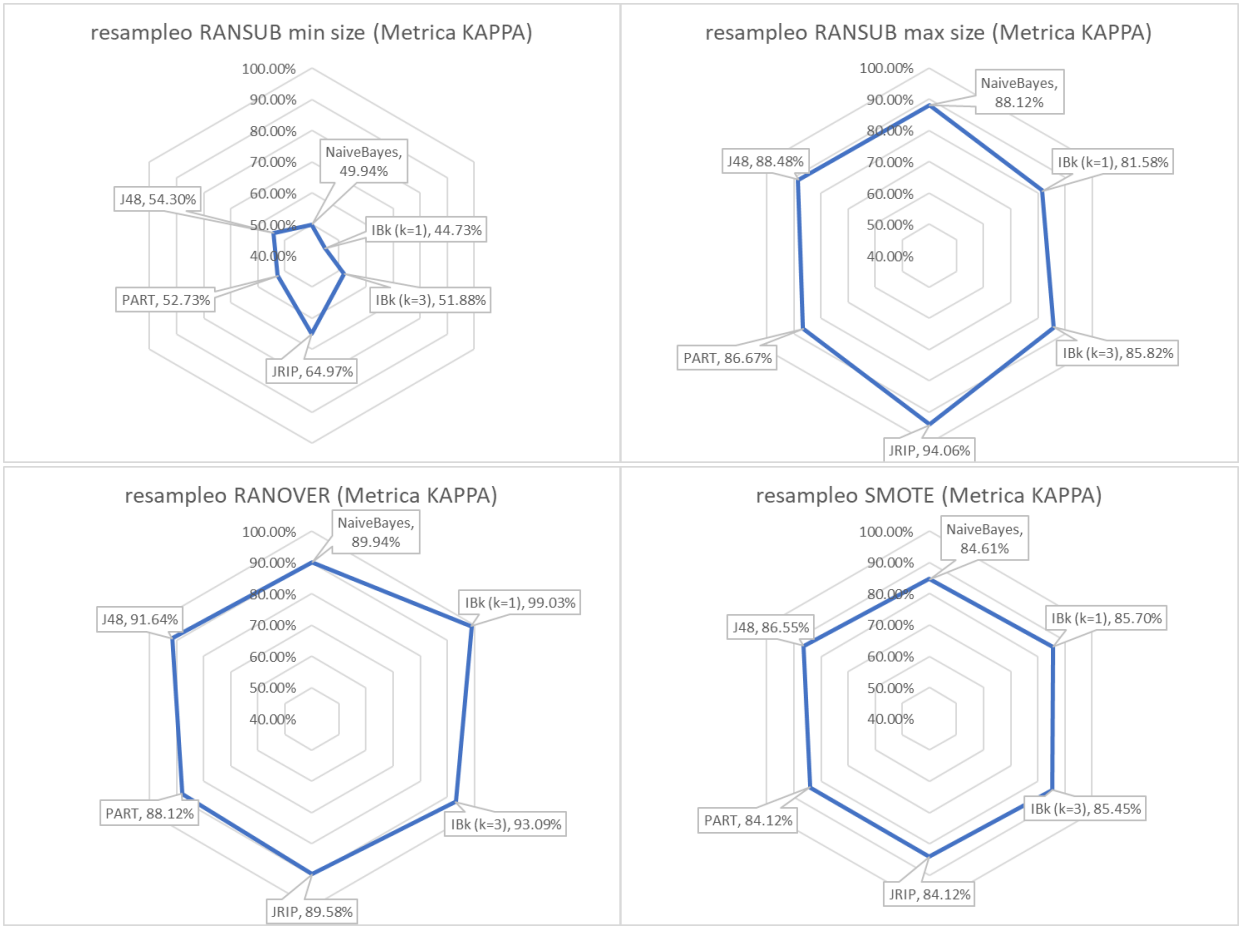

Fig. 17: *Porcentaje de veces en el que el óptimo de la distribución de clases en fase 2 mejoró al óptimo encontrado en la fase 1 para cada una de las técnicas de remuestreo evaluado bajo métrica del estadístico KAPPA*

En este caso, podemos observar que el método de remuestreo basado en el submuestreo aleatorio de tamaño de ventana mínimo es el que menos veces muestra un cambio desde el óptimo hallado en fase 1 al óptimo definitivo, lo cual sería lógico teniendo en cuenta que únicamente aportamos el "ajuste fino" del intervalo alrededor del óptimo de Fase 1[7], mientras que el método de remuestreo es el mismo. En cambio, las otras técnicas de remuestreo muestra resultados de cambio mucho mayores, tal y como muestran la **Fig. 17**, lo que muestra que el método de remuestreo forma parte del contexto cuando analizamos la distribución.

Así, evaluando los agregados para todos los clasificadores, la técnica del submuestreo aleatorio de tamaño mínimo solo mejora en fase 2 un 53.09% de los casos evaluados sobre todas las bases

[7] Hay que recordar que en Fase 1 se utiliza siempre el submuestreo aleatorio de tamaño mínimo para calcular una primera aproximación al óptimo, al tratarse del método que requiere un menor esfuerzo computacional.

de datos, mientras que el submuestreo de tamaño máximo mejora en el 87.45% de los casos; el SMOTE, en el 85.09% de los casos y en sobremuestreo aleatorio, hasta en un 91.90% de los casos. Si agrupamos los resultados de todas las técnicas de remuestreo, la segunda fase aporta mejoras en el 79.38% de los casos, aunque como hemos señalado, este valor esta lastrado por los bajos valores que aporta hacer la segunda fase con el método de submuestreo aleatorio de tamaño mínimo que, por otra parte, es el de menor coste computacional.

Por lo tanto, al margen de los resultados finales obtenidos por el clasificador que evaluamos en el apartado *5.1 Análisis de la evaluación de resultados finales para cada clasificador tipo,* sí que podemos constatar que la Fase 2 modifica, el óptimo de la distribución, observándose incluso cuando utilizamos el mismo método de remuestreo que en fase 1 (RANSUB Size Min) un 53.09% de ocasiones en el que se ajusta la distribución, pero, además, el método de remuestreo supone una condición de contexto para obtener la distribución óptima, vista la diferencia del porcentaje de cambios de RANSUB Size Min y el resto de las opciones.

Finalmente vamos a comprobar **cuantas veces la distribución óptima final es distinta a la distribución balanceada (50%)**

Fig. 18: *Porcentaje de veces en el que el óptimo de la distribución de clases en fase 2 es distinto a la distribución balanceada (50%) para cada una de las técnicas de remuestreo evaluado bajo métrica del estadístico KAPPA*

Podemos comprobar en las gráficas de la **Fig.12** que, en general, algunos clasificadores responden mejor a la distribución balanceada que otros, pero además el método con el que se realiza el remuestreo puede hacer que estos resultados varíen, no sólo cuantitativamente (el valor del porcentaje indicado) sino relativamente, así JRIP es el clasificador que más veces encuentra el óptimo en la distribución balanceada cuando usamos un sobremuestreo aleatorio, pero J48 es el clasificador que más veces encuentra el óptimo en la distribución balanceada al usar un remuestro SMOTE.

En cualquier caso, para la métrica Kappa, agregando todos los clasificadores, el RANOVER encuentra el óptimo de la distribución distinto a la muestra balanceada en el 81.76% de las veces, el SMOTE en el 83.74% de las ocasiones, el RANSUB Min Size en el 92.57% de las veces y en el RANSUB Max Size en el 91.58% de las veces, obteniendo un agregado global de las cuatro técnicas de remuestreo del 87.41% lo que muestra que la distribución balanceada sólo es la óptima en el 12.59% de las ocasiones, por lo que, en general no es una opción buena remuestrear definiendo como objetivo único la distribución balanceada.

5.4.2 Métrica basada en al área bajo la curva ROC

Repetimos el análisis, esta vez bajo el criterio de bondad del área bajo la curva ROC, empezando por el **porcentaje de casos en los que el remuestreo óptimo es distinto al de la muestra original,** mostrando en la **Fig. 19** dichos porcentajes sobre el óptimo definitivo (tras la fase2) y sobre el óptimo que se encuentra en la fase1, como referencia, teniendo en cuenta que esa fase 1 siempre utiliza el método del submuestreo aleatorio de tamaño mínimo (RANSUB Min Size).

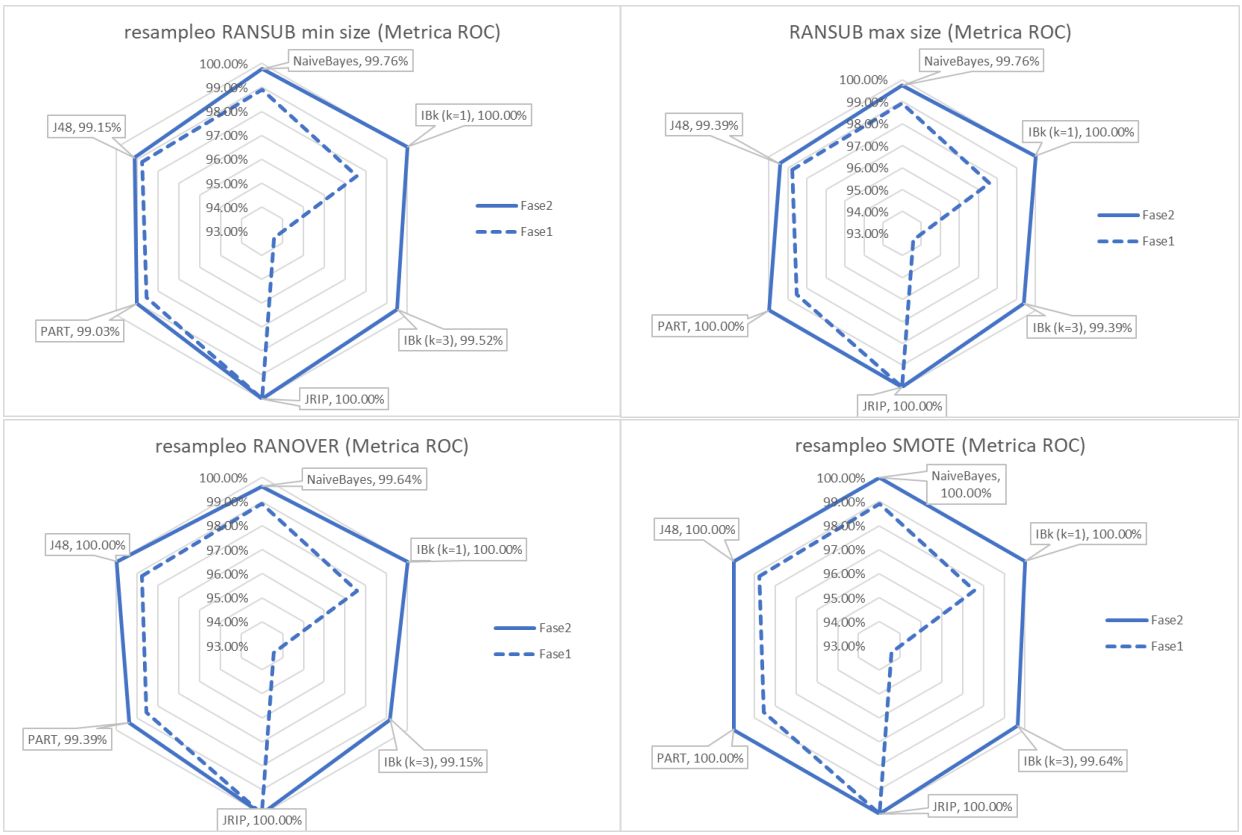

Fig. 19: *Porcentaje de veces en el que el óptimo de la distribución de clases en fase 1 es distinto a la distribución original bajo métrica del área ROC*

Se puede observar de nuevo que, para los clasificadores basados en los k vecinos cercanos, el porcentaje de casos en los que el óptimo es distinto al de la distribución original en fase 1 es un poco más bajo, mientras que en JRIP en todos los casos se obtuvo un mejor resultado remuestreando. Sin embargo, revisando los valores, se puede observar que el porcentaje de casos en los que la distribución óptima es distinta a la de partida es muy alto.

Como diferencias respecto a la métrica KAPPA anteriormente comentada, ahora es el k-NN con 3 vecinos el que menor porcentaje muestra en fase 1, en vez del que usa el vecino más cercano, mientras que en fase 2, todos los clasificadores arrojan porcentajes altísimos en cuanto a encontrar una distribución óptima distinta a la distribución del conjunto de datos inicial. En este caso, un total del 97.90% teniendo en cuenta el agregado de todos los clasificadores, obtienen una distribución óptima en fase 1 distinta a la distribución original, mientras que en fase 2 los valores agregados alcanzan valores de 99.58% para el RANSUB Min Size, 99.76% para RANSUB Max Size, 99.70 para RANOVER y hasta 99.94% para SMOTE.

Revisando **el número de veces en las que el óptimo de la distribución de fase 2 cambia respecto al calculado en fase 1** para este criterio de bondad, se representa en la **Fig.20** los porcentajes de cada caso.

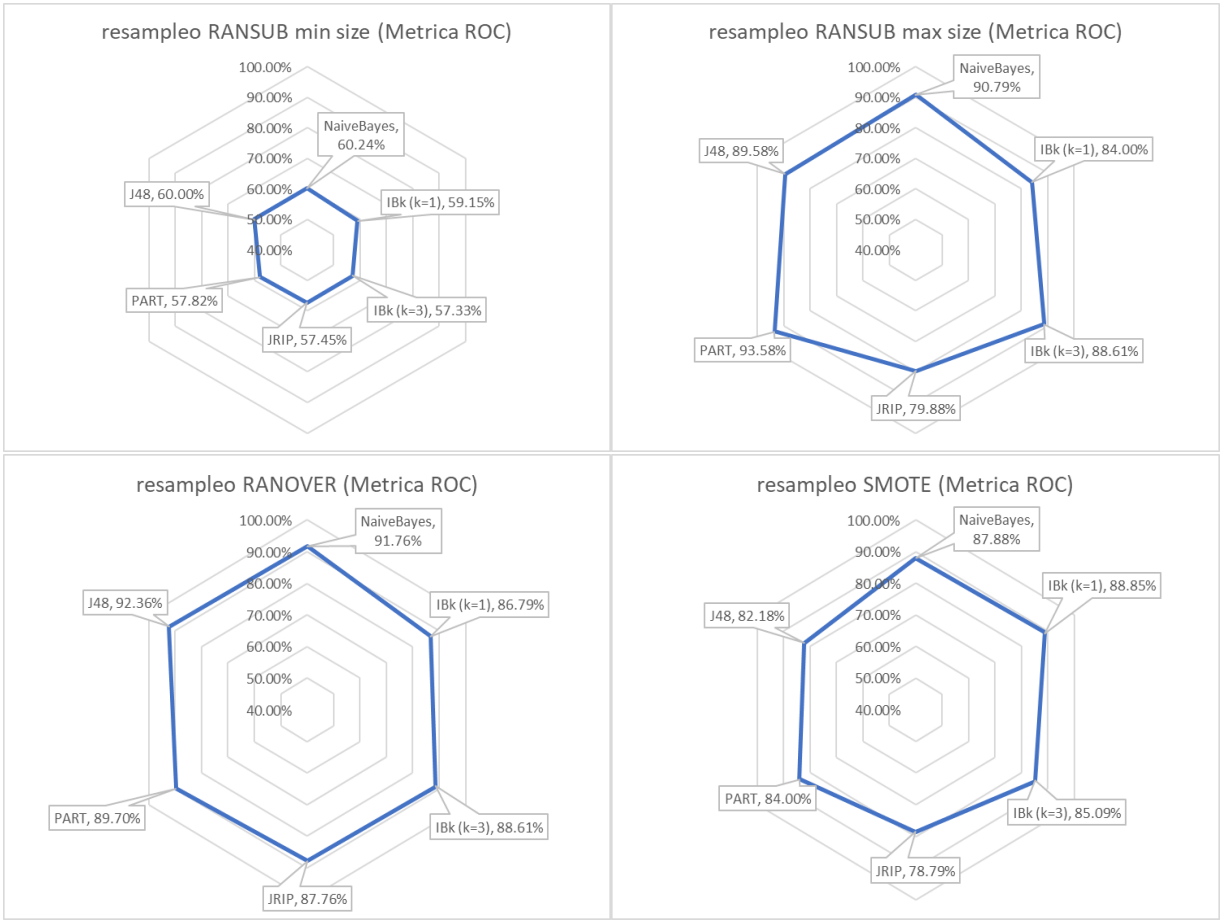

Fig. 20: *Porcentaje de veces en el que el óptimo de la distribución de clases en fase 2 mejoró al óptimo encontrado en la fase 1 para cada una de las técnicas de remuestreo evaluado bajo métrica del área ROC*

Las consideraciones respecto al algoritmo de remuestro basado en el RANSUB Min Size realizadas sobre el análisis de la métrica KAPPA aplican también aquí y así, podemos registrar que el porcentaje de mejora al aplicar este algoritmo para el agregado de todos los clasificadores sólo llega al 58.67%, mientras que los porcentajes de mejora para los otros algoritmos suben al 89.49% para el RANOVER, al 84.46% para el SMOTE y al 87.74% para el RANSUB Max Size. En total obtenemos un 80.09% de casos de mejora en la segunda fase respecto a la primera para el agregado de todas las técnicas de remuestreo.

De nuevo, el hecho de que para el resto de opciones de remuestreo no sólo cambiamos los porcentajes objetivos de distribución, sino también la técnica de remuestreo en sí, muestra esa dependencia del tipo de remuestreo.

Finalmente, atendemos al aspecto de **cuantas veces la distribución óptima final es distinta a la distribución balanceada (50%)**

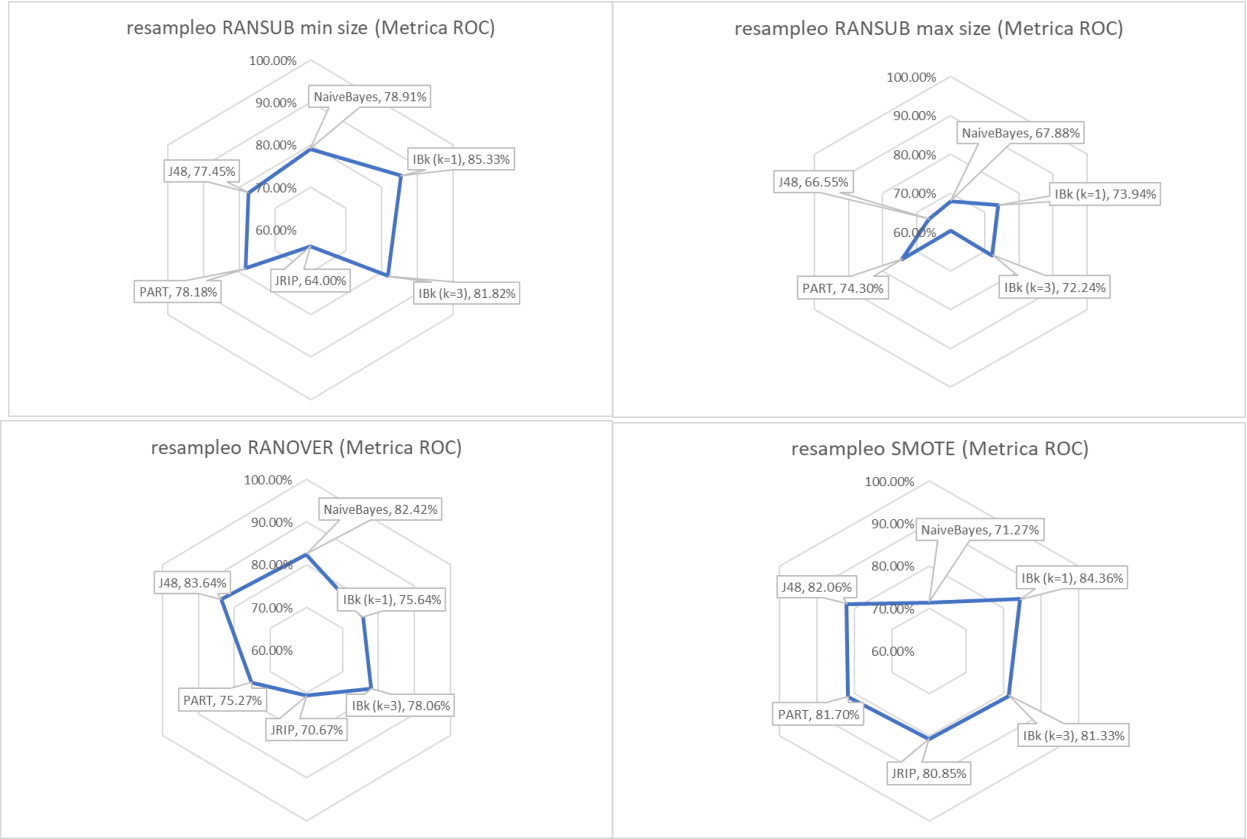

Fig. 21: *Porcentaje de veces en el que el óptimo de la distribución de clases en fase 2 es distinto a la distribución balanceada (50%) para cada una de las técnicas de remuestreo evaluado bajo métrica del área ROC*

En general, observamos que, para la métrica del área ROC, la distribución balanceada funciona mejor que para la métrica del estadístico KAPPA, sobre todo al usar la técnica de submuestreo aleatorio de tamaño máximo, donde sólo en el 69.09% de los casos el óptimo es distinto al 50%. Esto lo podíamos ya comprobar en el aspecto de las gráficas en el análisis preliminar de comparativa de métricas que se puede encontrar en los **Apéndices,** en el apartado *7.1*, donde podemos observar que, para el área bajo la curva ROC el máximo tiende a encontrarse en el la zona central del barrido, mientras que, para KAPPA, el máximo es más variable y dependiente del algotimo de clasificación.

Para SMOTE, por el contrario, no se da esa diferencia tan evidente y encontramos un valor de 71.27% para el clasificador Naive-Bayes pero el resto de clasificadores tienen su óptimo fuera de la distribución balanceada más del 80% de las veces. Para el sobremuestreo aleatorio y submuestro aleatorio de tamaño mínimo el agregado coincide con un 77.62% de casos en los que la distribución óptima es distinta del 50%.

En cualquier caso, para la métrica basada en el área ROC, pese a ser un porcentaje notablemente inferior a la evaluación basada en la métrica KAPPA, todavía en tan solo un 23.85% de los casos la distribución óptima es la balanceada si agregamos todos los clasificadores y todos los métodos de remuestreo sobre todas las bases de datos analizadas.

6 Conclusiones y posibles extensiones del estudio

6.1 Conclusiones

Uno de los objetivos principales definidos para este trabajo era ofrecer a la comunidad un procedimiento basado en la automatización del proceso, para poder aplicarlo en la búsqueda de la distribución óptima aplicada a la técnica de remuestreo sobre una base de datos desbalanceada. Para ello, el código necesario ha sido generado de modo que, al importar las clases así generadas en WEKA, podamos usar este entorno de trabajo para, aplicando un método de remuestreo específico y seleccionable por el usuario, obtener una distribución óptima por remuestreo para un clasificador y un criterio de bondad dado aplicado a un cierto problema.

Este código ha sido probado en la fase de experimentación posterior, testando su robustez al aplicarlo de manera combinada a distintos clasificadores, para las distintas técnicas de remuestreo implementadas y sobre un universo de conjuntos de datos de distintas características.

Algoritmos	**Método de Remuestreo**	**Métrica de Evaluación**
J48	Sin Remuestreo	Estadístico Kappa
NaiveBayes	Submuestreo aleatorio de tamaño mínimo	Area ROC
IBk (k=1)	Submuestreo aleatorio de tamaño máximo	
IBk (k=3)	Sobremuestreo aleatorio	
PART	SMOTE	
JRIP		

***Tabla 15**: Alternativas testadas en la propuesta de investigación*

Revisando los resultados de la experimentación, la dependencia de contexto se ha mostrado alta y se puede constatar que, tomando como variable objetivo, por ejemplo, la distribución óptima para una técnica de remuestreo, podemos considerar como variables de contexto a tener en cuenta cada uno de los grados de libertad que hemos definido como elecciones del experimento, esto es, el tipo de algoritmo de clasificación, el método de remuestreo y el criterio de bondad seleccionado, además de, obviamente, el conjunto de datos que representa el problema y sobre el que aplicamos el clasificador.

Atendiendo a los resultados el análisis de diferencias estadísticamente significativas, podemos concluir que el método de remuestreo SMOTE es el más prometedor para enfrentarse a un conjunto de datos para los algoritmos de clasificación estudiados y sobre las dos métricas de evaluación contempladas, comportándose mejor para el criterio de bondad del área bajo la curva ROC en general para todos los clasificadores y teniendo menos éxito en el clasificador Naive-Bayes para ambas métricas.

Por otro lado, el submuestreo aleatorio de tamaño mínimo, por sus características de bajo coste computacional, puede ser adecuado para el barrido de la primera fase, pero no es recomendable para implementarlo en la segunda fase para las métricas y clasificadores contemplados, ya que refleja degradaciones estadísticamente significativas en comparación a utilizar el conjunto de datos original sin remuestreo para todos los algoritmos de clasificación, bajo el criterio de bondad Kappa y también para los algoritmos de clasificación Naive-Bayes y PART usando el área bajo la curva ROC, no siendo capaz de mostrar ninguna mejora estadísticamente significativa en el resto de los casos.

Este aspecto es importante porque demuestra que el uso de cualquier tipo de remuestreo no implica necesariamente la mejora de los resultados del problema de clasificación respecto a la aplicación a la base de datos original y, de hecho, puede llegar a empeorarlo, por lo que es importante seleccionar los métodos de remuestreo adecuados.

De esta forma, se muestra que los métodos de remuestreo inteligentes, computacionalmente más complejos, logran un mejor desempeño. El método SMOTE, siendo aquel capaz de generar nuevas instancias sintéticas, es el que, además de mostrar una diferencia significativa aplicando el test de Wilcoxon, también obtuvo más veces (en mayor número de bases de datos) el resultado óptimo para cada clasificador y presentó el mejor resultado en la media agregada de todas las bases de datos bajo estudio.

Por otro lado, dentro del conjunto de experimentación, aquellas bases de datos con una distribución moderadamente desbalanceada (entre el 30% y el 35%), también disfrutaron de una mejora del desempeño cuando se aplicaron los clasificadores sobre unas versiones remuestreadas según la metodología planteada. Por ello, podemos concluir que este procedimiento no está restringido a su aplicación a los conjuntos de datos desbalanceados, sino que puede ser un método de mejora del desempeño de los clasificadores genérico para cualquier set de datos.

En este sentido, y teniendo en cuenta la alta dependencia del contexto, que no permite generalizaciones, la posibilidad de disponer de un procedimiento automatizado, en este caso sobre la plataforma WEKA, para la búsqueda de óptimos específicos para ciertos casos de uso, puede aportar a la comunidad que trabaja en el problema de la clasificación, una herramienta útil para mejorar los resultados del desempeño de los clasificadores.

Los resultados obtenidos en la fase de experimentación muestran que la distribución óptima que determina el método propuesto es, en la mayoría de los casos, distinta a la distribución original de la base de datos y distinta a la distribución balanceada al 50% para los 4 métodos de remuestreo y los 6 métodos de clasificación testados, ya fueran evaluados por cualquiera de las 2 métricas preseleccionadas

6.2 Lineas abiertas

Relativo a las **extensiones de este trabajo**, por un lado, tendríamos aquellas derivadas de la ampliación del código que permite la automatización de la metodología para que pueda aplicar más escenarios de remuestreo o nuevas funcionalidades y por otro, la extensión y profundización del escenario de experimentación.

En cuanto a su ambición de ofrecer una automatización en la búsqueda de la distribución óptima lo más amplia posible sobre las distintas variantes de remuestreo que se conocen, el trabajo es susceptible de ser permanentemente ampliado. Para empezar, la variante de SMOTE-Borderline descrita, no ha sido implementada en el código en ninguna de sus variantes (con construcción sintética a partir de sólo instancias de la clase minoritaria o la posibilidad de construirlas a partir de instancias de ambas clases). Además, la variante de ENN y la posibilidad de combinarla con otras técnicas de remuestreo también quedaría pendiente.

En cuanto a la funcionalidad sobre el entorno de trabajo WEKA, sería posible la inclusión de un nuevo panel en Weka-Explorer para que muestre gráficamente los resultados del barrido de la primera fase o la tabla resumen de los resultados de primera y segunda fase[8] para distintos clasificadores, siempre sobre la misma base de datos y sobre misma métrica para resultar comparables.

Otra mejora del método podría estar relacionada con implementar algún heurístico de búsqueda del óptimo u otro algoritmo de búsqueda inteligente en la segunda fase de la metodología que sustituiría al mecanismo de introducción de intervalos discretos de prueba, que debe configurar el usuario en la ventana del clasificador, sobre el óptimo de la primera fase. Esto aumentaría la automatización de la metodología. Por otro lado, si el tiempo de procesado es crítico, el heurístico podría ser parametrizado desde la ventana del clasificador o podría incluso ser implementado como opción alternativa o combinada sobre la variante actual.

En cuanto a la extensión del escenario de experimentación, más allá de probar el resultado de los distintos métodos de remuestreo que deberían ser implementados, el análisis también podría ser extendido para caracterizar el comportamiento de otras métricas.

En cuanto a la profundización del análisis, una posibilidad que queda abierta es realizar un análisis de las diferencias estadísticamente significativas basado en las últimas sugerencias publicadas, como el análisis bayesiano, ante los riesgos de aplicar Wilcoxon a una distribución que no sea necesariamente simétrica [50].

Por otro lado, tal y como hemos mencionado, queda pendiente confirmar que las técnicas de remuestreo pueden mejorar el desempeño de los clasificadores ante bases de datos levemente desbalanceadas, lo que abre a su vez la posibilidad de comprobar para estos casos, otras métricas

[8] Similares a los mostrados en el apartado *7.1 Análisis del comportamiento de métricas de bondad al variar la distribución de la clase minoritaria aplicando remuestreo* de los Apéndices.

habituales que habíamos desechado en nuestro estudio por su mala caracterización de los conjuntos de datos desbalanceados.

7 Apéndices

7.1 Análisis del comportamiento de métricas de bondad al variar la distribución de la clase minoritaria aplicando remuestreo

Se realizó un análisis inicial aplicado un subconjunto de bases de datos del set de experimentación, con el objetivo de comprobar la evolución de las métricas al variar la distribución objetivo al aplicar remuestreo.

Inicialmente se seleccionaron para este estudio:

- Accuracy, para comprobar su comportamiento, pese a que el análisis del estado del arte ya desaconsejaba su uso, tal y como se refiere en *2.5 Métricas de evaluación utilizadas para la evaluación de clasificadores aplicados sobre conjuntos de datos desbalanceados.*
- Kappa
- F-Measure
- Area bajo la curva ROC

Se presenta aquí, a modo de ejemplo, las gráficas de dicha evolución y las tablas resultantes para dos bases de datos[9], en los que se observa que las métricas Accuracy y F-Measure tienden a presentar los máximos en valores de distribuciones extremas a favor de la clase mayoritaria, mientras que el Area bajo la curva ROC los encuentra en zonas más balanceadas y Kappa muestra un comportamiento más variable.

Estas gráficas y tablas podrían considerarse como una propuesta para el diseño inicial de un prototipo para ser implementado sobre el Explorer de WEKA para facilitar el análisis del método propuesto.

[9] Cualitativamente, el comportamiento mostrado puede, en líneas generales, hacerse extensible a las otras BBDD testadas

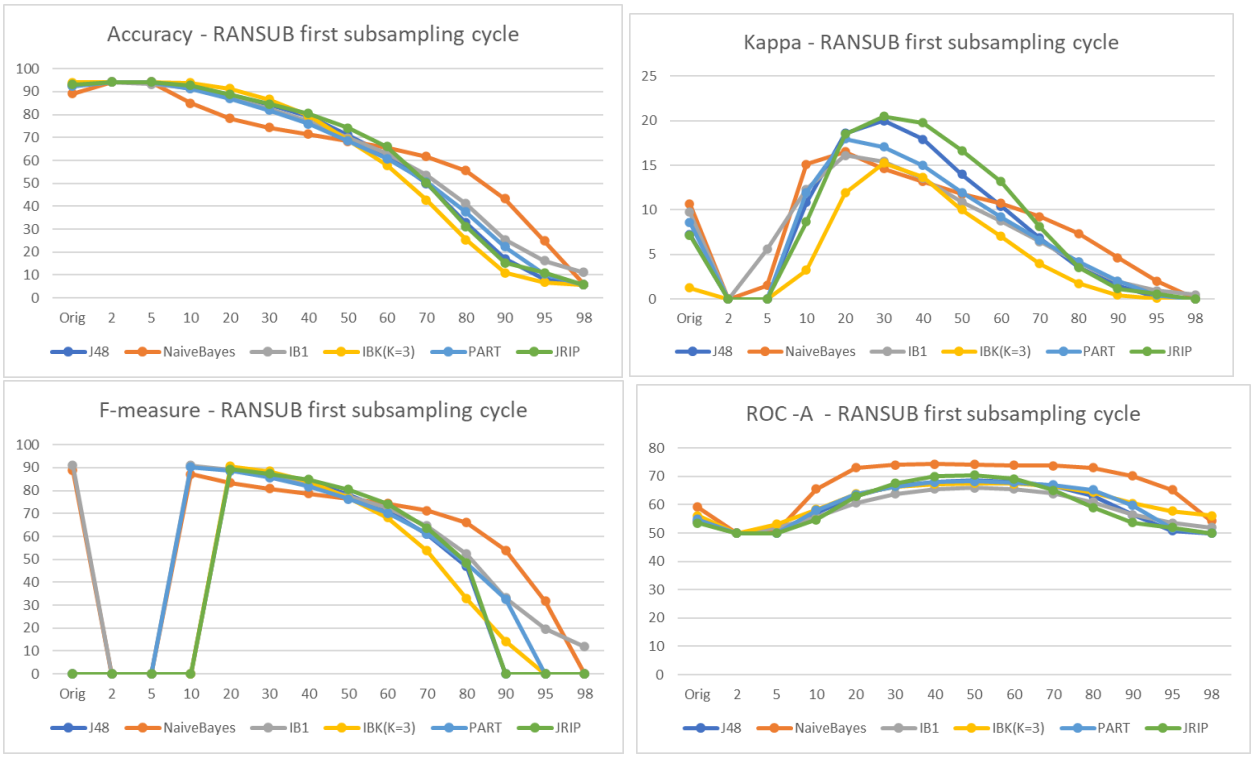

***Fig. 22**: Evolución de 4 métricas al aplicar el barrido de remuestreo de Fase 1 – Abalone 9_18*

	Fase 1		Fase 2		
Accuracy	**Max**	**dist(Max)**	**Max**	**dist(Max)**	**Sampling Method**
J48	94.26	2	94.64	2	RANOVER
NaiveBayes	94.31	5	94.3	5	RANSUB
IB1	94.26	2	94.42	2	SMOTE
IBK(K=3)	94.26	2	94.54	2	RANOVER
PART	94.26	2	94.5	2	RANOVER
JRIP	94.26	2	95.34	2	RANOVER
Alg(Max)			**Max**	**dist(Max)**	**dist(Max)**
JRIP			95.34	2	RANOVER

	Fase 1		Fase 2		
Kappa	**Max**	**dist(Max)**	**Max**	**dist(Max)**	**Sampling Method**
J48	20.00	30.00	40.9	20	RANOVER
NaiveBayes	16.51	20.00	21.88	10	SMOTE
IB1	16.11	20.00	37.32	10	SMOTE
IBK(K=3)	15.23	30.00	26.4	35	RANOVER
PART	17.95	20.00	41.36	50	RANOVER
JRIP	20.48	30.00	41.78	50	RANOVER
Alg(Max)			**Max**	**dist(Max)**	**dist(Max)**
JRIP			41.78	50	RANOVER

	Fase 1		Fase 2		
F-Measure	**Max**	**dist(Max)**	**Max**	**dist(Max)**	**Sampling Method**
J48	89.45	20.00	93.5	10	SMOTE
NaiveBayes	89.03	Orig	91.52	0.75	RANOVER
IB1	91.19	Orig	93.82	0.75	RANOVER
IBK(K=3)	90.61	20.00	93.02	10	SMOTE
PART	90.20	10.00	93.62	20	SMOTE
JRIP	89.43	20.00	94.42	10	SMOTE
Alg(Max)			**Max**	**dist(Max)**	**dist(Max)**
JRIP			94.42	10	SMOTE

	Fase 1		Fase 2		
ROC-A	**Max**	**dist(Max)**	**Max**	**dist(Max)**	**Sampling Method**
J48	68.52	50.00	71.86	60	RANOVER
NaiveBayes	74.36	40.00	74.48	30	RANOVER
IB1	65.93	50.00	68.62	45	SMOTE
IBK(K=3)	67.64	50.00	68.32	40	RANOVER
PART	68.37	50.00	72.82	60	RANOVER
JRIP	70.49	50.00	73.62	60	RANOVER
Alg(Max)			**Max**	**dist(Max)**	**dist(Max)**
NaiveBayes			74.48	30	RANOVER

***Tabla 16:** Resultados del óptimo de 2 fases – Abalone 9_18*

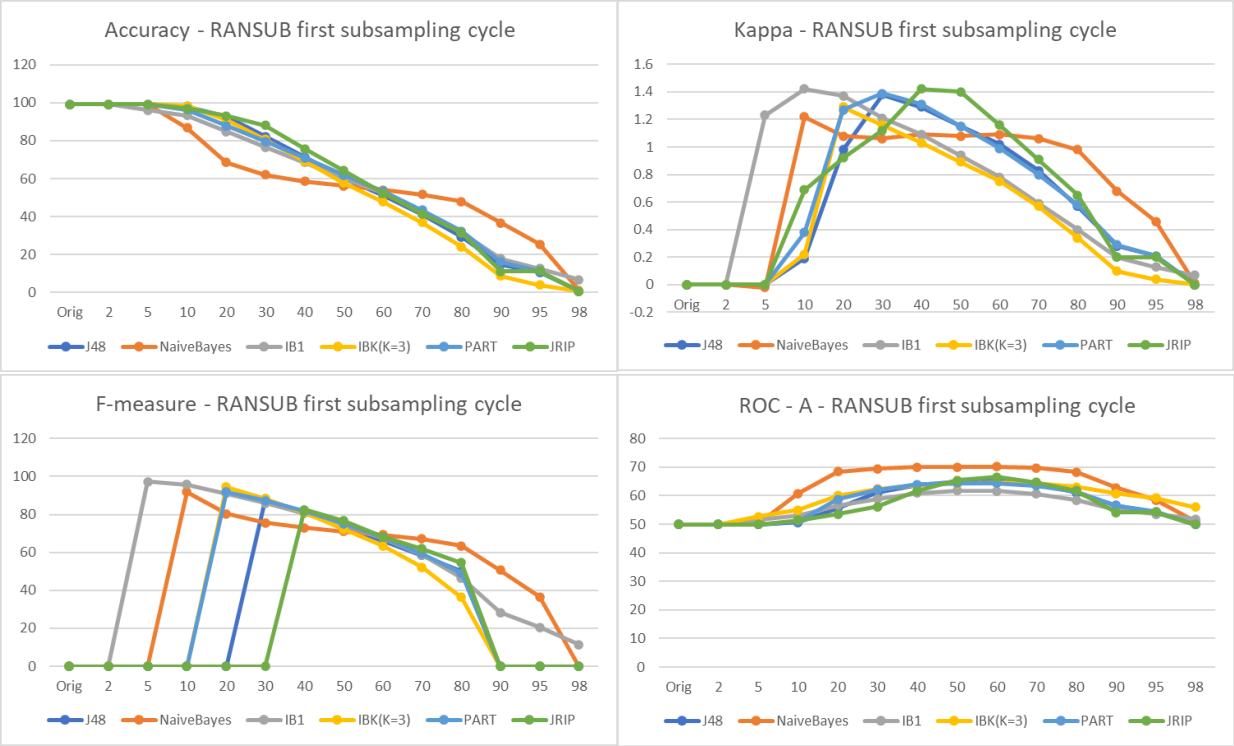

***Fig. 23**: Evolución de 4 métricas al aplicar el barrido de remuestreo de Fase 1 – Abalone 19*

	Fase 1		Fase 2		
Accuracy	**Max**	**dist(Max)**	**Max**	**dist(Max)**	**Sampling Method**
J48	99.23	Orig	99.24	0.77	RANSUB
NaiveBayes	99.23	Orig	99.24	0.77	RANSUB
IB1	99.23	Orig	99.24	0.77	RANSUB
IBK(K=3)	99.23	Orig	99.24	0.77	RANSUB
PART	99.23	Orig	99.24	0.77	RANSUB
JRIP	99.23	Orig	99.24	0.77	RANSUB
Alg(Max)			**Max**	**dist(Max)**	**dist(Max)**
J48			99.24	0.77	RANSUB

	Fase 1		Fase 2		
Kappa	**Max**	**dist(Max)**	**Max**	**dist(Max)**	**Sampling Method**
J48	1.38	30.00	4.48	50	RANOVER
NaiveBayes	1.22	10.00	1.28	10	RANSUB
IB1	1.42	10.00	1.48	15	RANSUB
IBK(K=3)	1.29	20.00	1.32	20	RANSUB
PART	1.39	30.00	4.14	25	RANOVER
JRIP	1.42	40.00	3.98	35	RANOVER
Alg(Max)			**Max**	**dist(Max)**	**dist(Max)**
J48			4.48	50	RANOVER

	Fase 1		Fase 2		
F-Measure	**Max**	**dist(Max)**	**Max**	**dist(Max)**	**Sampling Method**
J48	87.38	30.00	98.16	50	RANOVER
NaiveBayes	91.75	10.00	92.38	5	SMOTE
IB1	97.27	5.00	98.46	5	RANOVER
IBK(K=3)	94.42	20.00	97.74	10	RANOVER
PART	91.90	20.00	98	50	RANOVER
JRIP	82.45	40.00	97.84	50	RANOVER
Alg(Max)			**Max**	**dist(Max)**	**dist(Max)**
IB1			98.46	5	RANOVER

	Fase 1		Fase 2		
ROC	**Max**	**dist(Max)**	**Max**	**dist(Max)**	**Sampling Method**
J48	65.18	60.00	65.24	60	RANSUB
NaiveBayes	70.11	60.00	70.11	60	RANSUB
IB1	61.76	50.00	62.02	55	RANSUB
IBK(K=3)	64.49	60.00	64.52	60	RANSUB
PART	64.42	50.00	64.7	50	RANSUB
JRIP	66.57	60.00	67.14	60	RANSUB
Alg(Max)			**Max**	**dist(Max)**	**dist(Max)**
NaiveBayes			70.11	60	RANSUB

***Tabla 17:** Resultados del óptimo de 2 fases – Abalone 19*

7.2 Programación del experimento propuesto en la ventana Algorithms del Experimenter de WEKA

weka.classifiers.trees.J48 -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistOversample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistOversample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.bayes.NaiveBayes -num-decimal-places 3

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0" -ND-D

0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.bayes.NaiveBayes

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.bayes.NaiveBayes

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistOversample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.bayes.NaiveBayes

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.bayes.NaiveBayes

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.bayes.NaiveBayes

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.bayes.NaiveBayes

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistOversample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.bayes.NaiveBayes

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.bayes.NaiveBayes

weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S

100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistOversample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistOversample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.lazy.IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S

100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistOversample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistOversample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.lazy.IBk -- -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

weka.classifiers.rules.PART -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.PART -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.PART -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistOversample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.PART -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.PART -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.PART -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.PART -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistOversample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.PART -- -C 0.25 -M 2

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.PART -- -C 0.25 -M 2

weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S 1

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.JRip -- -F 3 -N 2.0 -O 2 -S 1

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.JRip -- -F 3 -N 2.0 -O 2 -S 1

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistOversample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.JRip -- -F 3 -N 2.0 -O 2 -S 1

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 3 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.JRip -- -F 3 -N 2.0 -O 2 -S 1

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -1 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.JRip -- -F 3 -N 2.0 -O 2 -S 1

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSubsample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.JRip -- -F 3 -N 2.0 -O 2 -S 1

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistOversample -S 1 -SM-S -2 -SM-D 50.0" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.JRip -- -F 3 -N 2.0 -O 2 -S 1

weka.classifiers.meta.NewDistribution2PhasesClassifier -ND-E 9 -ND-M "weka.filters.supervised.instance.NewDistSMOTE -SM-D 50.0 -K 5 -S 1" -ND-D 0f,2.0f,5.0f,10.0f,20.0f,30.0f,40.0f,50.0f,60.0f,70.0f,80.0f,90.0f,95.0f,98.0f -ND-I 0f,5.0f,10.0f -ND-S 100 -ND-F 5 -ND-R 5 -ND-B -ND-T 50 -S 1 -W weka.classifiers.rules.JRip -- -F 3 -N 2.0 -O 2 -S 1

7.3 Resultados bajo el criterio de bondad Kappa de las 33 bases de datos bajo estudio para cada clasificador al aplicar cada variante de remuestreo

Se presentan a continuación, bajo el criterio de bondad Kappa, detallados para cada una de las 33 bases de datos bajo estudio y para cada uno de los algoritmos de clasificación contemplados en una tabla separada, los resultados que ofrecen cada una de las alternativas de remuestreo evaluadas, incluyendo la evaluación directa del conjunto de datos original sin remuestreo. La última línea muestra el resultado de la media de los valores sobre el conjunto de las bases de datos presentadas, que es el valor resumen que se presenta en el apartado *5.1.1 Métrica basada en el estadístico KAPPA.*

En cada tabla se resalta el mejor resultado, correspondiente al algoritmo de remuestreo que mejor resultado obtuvo para la base de datos específica sobre el algoritmo de clasificación contemplado en la tabla en cuestión.

Metric	KAPPA	J48				
Dataset Id	Nombre del Dataset	Sin Remuestreo	RANSUB Size Min	RANSUB Size Max	RANOVER	SMOTE
1	Abalone19	0	0.0105	0.0133	**0.0356**	0.0186
2	Abalone9vs18	0.3233	0.2049	0.276	0.4224	**0.43**
3	Ecoli0vs1	0.9679	0.9219	0.9679	0.9679	**0.9679**
4	Ecoli1	0.7038	0.6546	0.692	**0.7161**	0.664
5	Ecoli2	0.7496	0.6148	0.7318	0.7584	**0.7561**
6	Ecoli3	0.5798	0.4324	0.5015	0.5535	**0.5876**
7	Ecoli4	0.6619	0.3921	0.6326	0.6302	**0.6856**
8	Glass0	0.5638	0.477	0.543	0.5454	**0.5651**
9	Glass0123vs456	**0.7978**	0.6691	0.7303	0.7407	0.7902
10	Glass1	0.3991	0.3206	0.38	**0.4261**	0.3968
11	Glass2	0.2202	0.0622	0.1246	0.2991	**0.3458**
12	Glass4	0.4986	0.2173	0.412	**0.6418**	0.6318
13	Glass5	0.7961	0.3858	0.4457	**0.8863**	0.8263
14	Glass6	0.7886	0.7012	**0.8109**	0.7729	0.7527
15	Haberman	0.1564	0.1838	0.2548	0.175	**0.2665**
16	Iris0	**0.9749**	0.9718	**0.9749**	**0.9749**	**0.9749**
17	New-thyroid1	0.8674	0.6969	0.7645	0.888	**0.8974**
18	New-thyroid2	0.8719	0.653	0.8047	0.8943	**0.9019**
19	Page-blocks0	0.8423	0.7284	0.8432	**0.8465**	0.843
20	Pima	0.4039	0.3748	0.4072	0.4108	**0.4124**
21	Segment0	0.9687	0.911	0.9675	**0.9737**	0.9735
22	Vehicle0	**0.8489**	0.7652	0.8454	0.8348	0.8233
23	Vehicle1	0.3367	0.3326	0.3943	0.329	**0.4059**
24	Vehicle2	0.8913	0.8251	0.8903	**0.8975**	0.8894
25	Vehicle3	0.354	0.3043	**0.3942**	0.3785	0.3757
26	Vowel0	**0.9369**	0.6696	0.86	0.9135	0.9191
27	Wisconsin	**0.8936**	0.8801	0.885	0.8974	**0.8986**
28	Yeast1	0.3513	0.3131	0.375	0.3643	**0.3989**
29	Yeast2vs8	0.055	0.3988	0.5141	**0.5554**	0.4441
30	Yeast3	0.7315	0.6836	**0.7578**	0.7274	0.7559
31	Yeast4	0.3218	0.2269	0.3256	**0.3268**	0.3165
32	Yeast5	**0.763**	0.3713	0.6855	0.7316	0.692
33	Yeast6	0.4979	0.2266	0.4034	0.4163	0.4801
AVE	**AVERAGE**	0.6096	0.5025	0.5942	0.6343	**0.6390**

Tabla 18: *Resultados de las 33 bases de datos bajo estudio para el clasificador J48 al aplicar cada variante de remuestreo. bajo el criterio de bondad Kappa*

Metric	KAPPA	NaiveBayes				
Dataset Id	Nombre del Dataset	Sin Remuestreo	RANSUB Size Min	RANSUB Size Max	RANOVER	SMOTE
1	Abalone19	0.0064	0.0043	0.0109	0.0108	**0.0123**
2	Abalone9vs18	**0.2187**	0.1738	0.1917	0.1876	0.1774
3	Ecoli0vs1	0.9454	0.9414	0.9453	0.9473	**0.9474**
4	Ecoli1	0.6327	0.6571	0.6586	**0.6635**	0.6579
5	Ecoli2	0.8026	0.7246	0.8046	0.7988	**0.806**
6	Ecoli3	**0.5452**	0.5028	0.5293	0.5434	0.5434
7	Ecoli4	**0.7922**	0.6489	0.7047	0.7538	0.7868
8	Glass0	0.3307	0.3098	0.3039	**0.345**	0.3447
9	Glass0123vs456	0.7219	0.7157	0.7119	**0.7347**	0.6694
10	Glass1	**0.2539**	0.1548	0.2388	0.248	0.2392
11	Glass2	0.0573	**0.082**	0.0677	0.071	0.0666
12	Glass4	0.1563	0.233	0.1889	0.2702	**0.3632**
13	Glass5	0.5895	0.2582	0.3757	0.5822	**0.6915**
14	Glass6	0.7902	0.7357	**0.7913**	0.7778	0.7783
15	Haberman	0.1788	0.2358	0.2515	0.252	**0.2597**
16	Iris0	**1**	0.9969	**1**	**1**	**1**
17	New-thyroid1	0.9199	0.8941	**0.9345**	0.9283	0.9028
18	New-thyroid2	0.9367	0.899	**0.9429**	0.9306	0.9287
19	Page-blocks0	0.6894	0.6902	0.6953	0.7011	**0.7034**
20	Pima	0.4426	0.4269	0.4299	0.4336	**0.4499**
21	Segment0	0.5569	**0.62**	0.5878	0.588	0.5766
22	Vehicle0	0.3249	0.3441	0.3419	**0.348**	0.3434
23	Vehicle1	0.3321	**0.3445**	0.3439	0.343	0.3322
24	Vehicle2	0.485	0.49	0.491	0.4779	**0.4914**
25	Vehicle3	0.3309	0.3242	**0.3404**	0.3338	0.3401
26	Vowel0	**0.6906**	0.6017	0.676	0.6782	0.6811
27	Wisconsin	0.9181	0.9149	0.9175	0.9162	**0.919**
28	Yeast1	0.3702	0.3465	0.3748	**0.378**	0.3722
29	Yeast2vs8	**0.587**	0.1801	0.4752	0.4859	0.5704
30	Yeast3	0.7265	0.7051	0.7428	**0.7447**	0.7385
31	Yeast4	0.2635	0.2319	0.3005	**0.3111**	0.2937
32	Yeast5	**0.6215**	0.4913	0.5609	0.5468	0.5617
33	Yeast6	**0.4296**	0.3492	0.3513	0.3627	0.3949
AVE	**AVERAGE**	0.5348	0.4918	0.5237	0.5362	**0.5438**

Tabla 19: *Resultados de las 33 bases de datos bajo estudio para el clasificador Naive Bayes al aplicar cada variante de remuestreo. bajo el criterio de bondad Kappa*

Metric	KAPPA	IBk1				
Dataset Id	Nombre del Dataset	Sin Remuestreo	RANSUB Size Min	RANSUB Size Max	RANOVER	SMOTE
1	Abalone19	-0.0075	-0.0003	**0.0282**	-0.0075	0.017
2	Abalone9vs18	**0.4666**	0.1549	0.3779	**0.4666**	0.4354
3	Ecoli0vs1	0.9378	0.9343	0.9321	0.9378	**0.938**
4	Ecoli1	0.6468	0.629	0.6359	0.6468	**0.6601**
5	Ecoli2	**0.7766**	0.697	0.7749	**0.7766**	0.7689
6	Ecoli3	0.4923	0.5202	0.4785	0.4923	**0.5608**
7	Ecoli4	0.8013	0.7359	0.7862	0.8013	**0.8121**
8	Glass0	**0.6291**	0.5093	0.6181	**0.6291**	0.6226
9	Glass0123vs456	0.8287	0.7787	0.8245	0.8287	**0.8454**
10	Glass1	0.5456	0.4734	0.5261	0.5456	**0.561**
11	Glass2	0.2101	0.0654	0.1783	0.2101	**0.2564**
12	Glass4	0.6816	0.4105	0.5297	0.6816	**0.7101**
13	Glass5	**0.7196**	0.2567	0.5811	**0.7196**	0.6955
14	Glass6	**0.8088**	0.7439	0.7997	**0.8088**	0.7917
15	Haberman	0.0813	0.0905	0.0957	0.0832	**0.1009**
16	Iris0	**1**	**1**	**1**	**1**	**1**
17	New-thyroid1	**0.9416**	0.8748	0.9147	**0.9416**	0.9405
18	New-thyroid2	**0.9398**	0.8783	0.9315	**0.9398**	0.9283
19	Page-blocks0	0.7395	0.643	0.7356	0.7391	**0.7398**
20	Pima	0.3411	0.3231	**0.3509**	0.3411	0.3508
21	Segment0	**0.9848**	0.9367	**0.9848**	**0.9848**	0.9829
22	Vehicle0	**0.8332**	0.7367	0.8191	**0.8332**	0.8328
23	Vehicle1	0.3085	0.2642	0.3172	0.3085	**0.3495**
24	Vehicle2	**0.885**	0.7652	0.8844	**0.885**	0.8804
25	Vehicle3	0.2836	0.2774	0.3063	0.2836	**0.3155**
26	Vowel0	**1**	0.7462	0.9917	**1**	**1**
27	Wisconsin	0.9173	0.9235	**0.9326**	0.9173	0.9265
28	Yeast1	0.3038	0.2569	0.3055	0.3038	**0.3071**
29	Yeast2vs8	**0.5349**	0.205	0.3993	**0.5349**	0.4226
30	Yeast3	0.6377	0.594	0.6347	0.6377	**0.6465**
31	Yeast4	0.3208	0.2427	0.3126	0.3208	**0.3358**
32	Yeast5	0.6721	0.4652	0.6792	0.6721	**0.6972**
33	Yeast6	**0.4552**	0.3573	0.4186	**0.4552**	0.4398
AVE	**AVERAGE**	0.6278	0.5300	0.6087	0.6279	**0.6325**

Tabla 20: *Resultados de las 33 bases de datos bajo estudio para el clasificador IBk (kNN) con k=1 al aplicar cada variante de remuestreo. bajo el criterio de bondad Kappa*

Metric	KAPPA	IBk3				
Dataset Id	Nombre del Dataset	Sin Remuestreo	RANSUB Size Min	RANSUB Size Max	RANOVER	SMOTE
1	Abalone19	-0.0005	0.0138	**0.0302**	-0.005	0.0146
2	Abalone9vs18	0.1739	0.1387	0.2555	0.3225	**0.3387**
3	Ecoli0vs1	0.9612	0.9381	0.9593	0.9326	**0.9637**
4	Ecoli1	**0.7225**	0.6794	0.7118	0.6894	0.709
5	Ecoli2	**0.8412**	0.7245	0.8359	0.836	0.8406
6	Ecoli3	0.5394	0.534	0.5277	0.5267	**0.5539**
7	Ecoli4	**0.8389**	0.7274	0.8111	0.7509	0.8146
8	Glass0	**0.5863**	0.5117	0.573	0.5756	0.5073
9	Glass0123vs456	0.8144	0.7913	0.8001	0.8377	**0.8519**
10	Glass1	**0.5147**	0.4525	0.5122	0.5006	0.5087
11	Glass2	0.0763	0.0429	0.0439	0.2345	**0.2372**
12	Glass4	0.5579	0.338	0.4864	0.5669	**0.6633**
13	Glass5	0.6687	0.1911	0.3072	**0.6726**	0.6544
14	Glass6	0.7558	0.7493	0.7482	0.7543	**0.7988**
15	Haberman	0.1042	0.1275	0.1141	0.1175	**0.1314**
16	Iris0	**1**	**1**	**1**	**1**	**1**
17	New-thyroid1	0.8962	0.8911	0.9138	**0.9458**	0.9286
18	New-thyroid2	0.8967	0.8668	0.9347	0.9406	**0.942**
19	Page-blocks0	**0.7435**	0.6505	0.7383	0.7174	0.7411
20	Pima	0.4181	0.3715	0.4103	0.3825	**0.4222**
21	Segment0	0.975	0.9256	0.9763	0.9771	**0.9801**
22	Vehicle0	**0.8203**	0.7219	0.8141	0.8137	0.8184
23	Vehicle1	0.3122	0.3135	0.3461	0.3356	**0.3668**
24	Vehicle2	**0.8659**	0.7324	0.8636	0.8658	0.8645
25	Vehicle3	0.2936	0.2904	0.3425	0.3562	**0.3647**
26	Vowel0	0.9937	0.6322	0.9439	0.9928	**1**
27	Wisconsin	0.9396	0.9295	0.9395	0.9371	**0.9441**
28	Yeast1	0.3208	0.2904	0.3156	0.312	**0.3303**
29	Yeast2vs8	**0.6473**	0.3909	0.4951	0.3987	0.4007
30	Yeast3	0.7141	0.6664	**0.7171**	0.7107	0.6991
31	Yeast4	0.2833	0.3069	**0.3817**	0.312	0.3773
32	Yeast5	**0.7232**	0.5136	0.6509	0.6885	0.7067
33	Yeast6	**0.549**	0.3516	0.4464	0.3969	0.4895
AVE	**AVERAGE**	0.6226	0.5396	0.6044	0.6181	**0.6353**

Tabla 21: *Resultados de las 33 bases de datos bajo estudio para el clasificador IBk (kNN) con k=3 al aplicar cada variante de remuestreo. bajo el criterio de bondad Kappa*

Metric	KAPPA	PART				
Dataset Id	Nombre del Dataset	Sin Remuestreo	RANSUB Size Min	RANSUB Size Max	RANOVER	SMOTE
1	Abalone19	-0.0005	0.007	0.0101	**0.0563**	0.0236
2	Abalone9vs18	0.3545	0.1646	0.301	**0.4633**	0.3692
3	Ecoli0vs1	0.9484	0.9176	0.9603	0.9582	**0.9679**
4	Ecoli1	0.6825	0.6492	0.6991	0.6858	**0.7059**
5	Ecoli2	0.7384	0.6013	0.7307	**0.7649**	0.7467
6	Ecoli3	0.4939	0.4392	0.4465	0.5378	**0.553**
7	Ecoli4	0.6701	0.3921	0.6363	0.6557	**0.72**
8	Glass0	0.5716	0.4944	0.5415	**0.5858**	0.545
9	Glass0123vs456	0.7842	0.6615	0.7324	0.7596	**0.7945**
10	Glass1	0.3674	0.3329	0.3448	0.4227	**0.4242**
11	Glass2	0.2052	0.047	0.1243	0.2768	**0.2882**
12	Glass4	0.5307	0.2173	0.4041	0.633	**0.6745**
13	Glass5	**0.7961**	0.3858	0.4303	0.7604	0.7289
14	Glass6	0.7832	0.7012	**0.7997**	0.7615	0.7651
15	Haberman	0.1734	0.1879	0.2319	0.1825	**0.2322**
16	Iris0	**0.9749**	0.9718	**0.9749**	**0.9749**	**0.9749**
17	New-thyroid1	0.8798	0.6897	0.7871	0.8608	**0.9003**
18	New-thyroid2	**0.8872**	0.6584	0.8011	0.8679	0.8694
19	Page-blocks0	0.8205	0.7217	**0.8344**	0.8322	0.8276
20	Pima	0.4008	0.3766	0.4076	0.401	**0.4152**
21	Segment0	0.9744	0.9236	0.9714	0.9736	**0.9746**
22	Vehicle0	0.8484	0.7763	0.8315	0.8333	**0.852**
23	Vehicle1	0.2629	0.2884	0.3775	0.3824	**0.4035**
24	Vehicle2	0.8978	0.8548	0.8976	**0.9005**	0.9
25	Vehicle3	0.2662	0.2706	0.3374	0.3618	**0.4101**
26	Vowel0	**0.9116**	0.6917	0.854	0.9075	0.9002
27	Wisconsin	0.8954	0.8903	0.8905	**0.9007**	0.8957
28	Yeast1	0.3513	0.3378	0.3569	0.3291	**0.3657**
29	Yeast2vs8	0.0648	0.3752	0.4369	**0.542**	0.3814
30	Yeast3	0.6926	0.6704	0.7162	0.7258	**0.7446**
31	Yeast4	0.2965	0.2247	0.3171	**0.3502**	0.2952
32	Yeast5	**0.717**	0.3713	0.6778	0.6605	0.7038
33	Yeast6	**0.5077**	0.2319	0.4093	0.4184	0.4628
AVE	**AVERAGE**	0.5985	0.5007	0.5840	0.6281	**0.6308**

Tabla 22: *Resultados de las 33 bases de datos bajo estudio para el clasificador PART al aplicar cada variante de remuestreo. bajo el criterio de bondad Kappa*

Metric	KAPPA	JRIP				
Dataset Id	Nombre del Dataset	Sin Remuestreo	RANSUB Size Min	RANSUB Size Max	RANOVER	SMOTE
1	Abalone19	-0.0003	0.0123	0.0182	**0.0486**	0.0215
2	Abalone9vs18	0.4001	0.2292	0.276	**0.4138**	0.3869
3	Ecoli0vs1	**0.9696**	0.9464	**0.9696**	0.9456	0.9616
4	Ecoli1	0.7271	0.6765	0.6928	**0.7347**	0.7203
5	Ecoli2	0.752	0.5752	0.6823	**0.7525**	0.7349
6	Ecoli3	0.5365	0.3933	0.4898	0.5931	**0.612**
7	Ecoli4	0.6569	0.3982	0.5553	0.7021	**0.7138**
8	Glass0	0.5088	0.4659	0.5215	0.6036	**0.6051**
9	Glass0123vs456	0.7429	0.6715	0.7509	0.75	**0.7817**
10	Glass1	0.3799	0.2731	0.3833	**0.4165**	0.4092
11	Glass2	0.0892	0.0431	0.0666	0.3146	**0.3709**
12	Glass4	0.4936	0.1799	0.3388	0.5776	**0.578**
13	Glass5	0.4953	0.0829	0.2963	**0.6495**	0.6208
14	Glass6	**0.7787**	0.6245	0.7749	0.742	0.7748
15	Haberman	0.198	0.1921	**0.2673**	0.2089	0.2644
16	Iris0	0.9906	0.9721	0.9937	**0.9937**	0.9906
17	New-thyroid1	0.8513	0.7107	0.884	**0.9117**	0.9059
18	New-thyroid2	0.8428	0.689	0.8229	0.8452	**0.8695**
19	Page-blocks0	0.8388	0.721	**0.8473**	0.8361	0.8389
20	Pima	0.4236	0.3977	0.4175	0.4159	**0.4382**
21	Segment0	0.9698	0.8945	0.9576	0.9708	**0.9791**
22	Vehicle0	0.8169	0.7045	0.7881	**0.8397**	0.8211
23	Vehicle1	0.3644	0.3177	0.3549	0.3739	**0.3977**
24	Vehicle2	0.8804	0.8137	0.8838	0.8977	**0.9013**
25	Vehicle3	0.3581	0.3061	0.3482	**0.4014**	0.38899
26	Vowel0	**0.9083**	0.7163	0.8355	0.8989	0.9048
27	Wisconsin	0.9126	0.8929	**0.9136**	0.9017	0.9135
28	Yeast1	0.3511	0.3116	0.3608	0.3572	**0.3739**
29	Yeast2vs8	**0.6189**	0.16	0.5092	0.5249	0.4004
30	Yeast3	0.7485	0.6663	**0.752**	0.7431	0.7497
31	Yeast4	**0.3663**	0.2865	0.2996	0.3194	0.314
32	Yeast5	**0.7322**	0.4161	0.59	0.6708	0.7119
33	Yeast6	**0.5325**	0.2608	0.3043	0.422	0.4278
AVE	**AVERAGE**	0.6132	0.4849	0.5741	0.6296	**0.6328**

Tabla 23: *Resultados de las 33 bases de datos bajo estudio para el clasificador JRIP al aplicar cada variante de remuestreo. bajo el criterio de bondad Kappa*

7.4 Resultados bajo el criterio de bondad del área bajo la curva ROC de las 33 bases de datos bajo estudio para cada clasificador al aplicar cada variante de remuestreo

Se presentan a continuación, bajo el criterio de bondad del área bajo la curva ROC (ROC_A), detallados para cada una de las 33 bases de datos bajo estudio y para cada uno de los algoritmos de clasificación contemplados en una tabla separada, los resultados que ofrecen cada una de las alternativas de remuestreo evaluadas, incluyendo la evaluación directa del conjunto de datos original sin remuestreo. La última línea muestra el resultado de la media de los valores sobre el conjunto de las bases de datos presentadas, que es el valor resumen que se presenta en el apartado *5.1.2 Métrica basada en al área bajo la curva ROC*.

En cada tabla se resalta el mejor resultado, correspondiente al algoritmo de remuestreo que mejor resultado obtuvo para la base de datos específica sobre el algoritmo de clasificación contemplado en la tabla en cuestión.

Metric	ROC A	J48				
Dataset Id	**Dataset Name**	**Sin Remuestreo**	**RANSUB Size Min**	**RANSUB Size Max**	**RANOVER**	**SMOTE**
1	Abalone19	0.5	**0.6561**	0.6412	0.529	0.529
2	Abalone9vs18	0.6462	0.7043	**0.7257**	0.7194	0.7239
3	Ecoli0vs1	**0.9829**	0.9605	**0.9829**	**0.9829**	**0.9829**
4	Ecoli1	**0.9152**	0.879	0.8913	0.8845	0.9037
5	Ecoli2	0.8478	0.86	0.8709	**0.8795**	0.877
6	Ecoli3	0.8259	0.8161	**0.8569**	0.8032	0.8507
7	Ecoli4	0.8194	0.8164	**0.8882**	0.8473	0.8638
8	Glass0	0.8019	0.7704	0.7781	0.7874	**0.8064**
9	Glass0123vs456	0.8781	0.8913	0.8877	0.8862	**0.8955**
10	Glass1	0.6999	0.6602	0.7041	**0.7321**	0.7193
11	Glass2	0.6938	0.5924	0.6597	0.6445	**0.7326**
12	Glass4	0.7548	0.7882	0.8359	0.8634	**0.8741**
13	Glass5	**0.9846**	0.9059	0.9259	0.9471	0.9444
14	Glass6	0.9025	0.8925	**0.921**	0.8836	0.8872
15	Haberman	0.5689	0.6133	**0.6302**	0.6006	0.6251
16	Iris0	**0.984**	0.982	**0.984**	**0.984**	**0.984**
17	New-thyroid1	0.9344	0.9248	0.9229	0.9473	**0.9595**
18	New-thyroid2	0.9265	0.9151	0.9372	0.9523	**0.9538**
19	Page-blocks0	0.9403	0.9367	0.9467	0.9304	**0.9504**
20	Pima	0.7441	0.7035	**0.7465**	0.7328	0.7364
21	Segment0	0.9844	0.9734	0.9845	0.9897	**0.9917**
22	Vehicle0	**0.9456**	0.9222	0.9394	0.9256	0.9409
23	Vehicle1	0.7121	0.6999	**0.7345**	0.6701	0.7343
24	Vehicle2	0.9435	0.9266	0.9505	0.9533	**0.9566**
25	Vehicle3	0.7366	0.7082	0.7413	0.7051	**0.742**
26	Vowel0	**0.966**	0.9332	0.9525	0.9633	0.9641
27	Wisconsin	0.9528	0.9435	**0.9617**	0.9527	0.9587
28	Yeast1	0.715	0.7049	**0.7226**	0.6936	0.7146
29	Yeast2vs8	0.5165	0.7087	0.7487	0.7941	**0.8001**
30	Yeast3	0.8877	0.8851	0.9205	0.8969	**0.9212**
31	Yeast4	0.7578	0.8182	**0.8304**	0.6952	0.7897
32	Yeast5	0.8995	0.9295	**0.9567**	0.888	0.9212
33	Yeast6	0.7867	**0.8469**	**0.8469**	0.735	0.8049
AVE	**AVERAGE**	0.8229	0.8263	0.8493	0.8303	**0.8497**

Tabla 24: *Resultados de las 33 bases de datos bajo estudio para el clasificador J48 al aplicar cada variante de remuestreo. bajo el criterio de bondad del área bajo la curva ROC*

Metric	ROC A	NaiveBayes				
Dataset Id	**Dataset Name**	**Sin Remuestreo**	**RANSUB Size Min**	**RANSUB Size Max**	**RANOVER**	**SMOTE**
1	Abalone19	0.6921	0.6896	**0.6952**	0.6945	0.6803
2	Abalone9vs18	0.7456	0.7425	0.7472	0.7452	**0.7476**
3	Ecoli0vs1	**0.9965**	0.9959	**0.9965**	**0.9965**	0.9963
4	Ecoli1	0.9233	0.9221	**0.9246**	0.9222	0.9229
5	Ecoli2	0.9496	0.9436	0.9469	0.949	**0.9499**
6	Ecoli3	0.9337	0.9278	0.9321	0.9321	**0.9354**
7	Ecoli4	0.9929	0.9827	0.9924	**0.9935**	0.9924
8	Glass0	0.7589	0.7488	0.7527	0.7621	**0.7644**
9	Glass0123vs456	0.9586	0.9331	0.9486	0.959	**0.9606**
10	Glass1	**0.68**	0.6263	0.667	0.6634	0.6568
11	Glass2	0.7386	0.7153	0.7328	0.7402	**0.7419**
12	Glass4	0.7527	0.7713	**0.7848**	0.7557	0.7813
13	Glass5	**0.9663**	0.8534	0.8985	0.962	0.9351
14	Glass6	0.9147	0.925	**0.931**	0.9172	0.8917
15	Haberman	0.6406	**0.6444**	0.6395	0.6396	0.6285
16	Iris0	**1**	**1**	**1**	**1**	**1**
17	New-thyroid1	**0.9998**	0.999	0.9995	**0.9998**	**0.9998**
18	New-thyroid2	**1**	0.9997	0.9997	0.9998	0.9998
19	Page-blocks0	**0.9731**	0.968	0.9718	0.973	0.9722
20	Pima	0.8137	0.7991	0.8129	0.8138	**0.815**
21	Segment0	0.9865	0.9857	**0.9867**	0.9862	0.9817
22	Vehicle0	0.812	0.8097	0.811	0.8125	**0.8156**
23	Vehicle1	0.7277	**0.7335**	0.7255	0.7283	0.7282
24	Vehicle2	0.8532	0.8605	0.8502	0.8563	**0.8654**
25	Vehicle3	0.7203	0.7182	0.7201	0.7205	**0.722**
26	Vowel0	0.9801	0.9709	0.9782	**0.9802**	0.9769
27	Wisconsin	0.9847	0.983	0.9846	**0.9853**	0.9822
28	Yeast1	**0.7804**	0.769	0.7773	0.779	0.7745
29	Yeast2vs8	0.81	0.7535	0.7951	**0.8152**	0.8109
30	Yeast3	**0.9637**	0.9579	0.9636	0.9632	0.9567
31	Yeast4	0.8621	0.8334	0.8604	**0.863**	0.8466
32	Yeast5	**0.9865**	0.9668	0.9848	0.9857	0.9813
33	Yeast6	0.939	0.9405	0.937	0.9397	**0.944**
AVE	**AVERAGE**	**0.8738**	0.8627	0.8712	0.8737	0.8715

Tabla 25: *Resultados de las 33 bases de datos bajo estudio para el clasificador Naive Bayes al aplicar cada variante de remuestreo. bajo el criterio de bondad del área bajo la curva ROC*

Metric	ROC A	IBk1				
Dataset Id	**Dataset Name**	**Sin Remuestreo**	**RANSUB Size Min**	**RANSUB Size Max**	**RANOVER**	**SMOTE**
1	Abalone19	0.496	0.611	**0.6423**	0.496	0.5413
2	Abalone9vs18	0.7214	0.6865	0.7257	0.7217	**0.7512**
3	Ecoli0vs1	0.9682	0.9561	0.9652	**0.9707**	0.97
4	Ecoli1	0.8265	0.8603	0.8524	0.8243	**0.8658**
5	Ecoli2	0.8957	0.8739	0.896	**0.8967**	0.8949
6	Ecoli3	0.7471	0.8248	0.8376	0.7481	**0.8536**
7	Ecoli4	0.8859	**0.9394**	0.9386	0.8872	0.9171
8	Glass0	0.8245	0.7726	0.8162	0.8226	**0.825**
9	Glass0123vs456	0.9173	0.9128	0.9319	0.9112	**0.9334**
10	Glass1	0.771	0.7418	0.7687	0.7674	**0.7856**
11	Glass2	0.5943	0.6197	0.6703	0.5961	**0.6781**
12	Glass4	0.864	0.8752	0.8648	0.864	**0.9282**
13	Glass5	0.8837	0.8466	0.9107	0.8834	**0.9183**
14	Glass6	0.8965	0.8672	0.9053	0.8949	**0.9145**
15	Haberman	0.5647	0.5576	0.5676	0.5625	**0.5701**
16	Iris0	**1**	**1**	**1**	**1**	**1**
17	New-thyroid1	0.9721	0.9623	**0.9821**	0.9717	0.9814
18	New-thyroid2	0.9762	0.9561	0.9799	0.9776	**0.9877**
19	Page-blocks0	0.8557	0.8926	**0.8989**	0.8558	0.8954
20	Pima	0.6674	0.6766	**0.6881**	0.6664	0.6875
21	Segment0	**0.9935**	0.9814	0.9928	0.9927	**0.9935**
22	Vehicle0	0.9219	0.9022	0.925	0.9218	**0.9399**
23	Vehicle1	0.6527	0.6801	0.6923	0.6526	**0.7118**
24	Vehicle2	0.9493	0.9021	0.945	0.9482	**0.9554**
25	Vehicle3	0.6382	0.6728	0.6864	0.6349	**0.6865**
26	Vowel0	**1**	0.951	0.9882	**1**	**1**
27	Wisconsin	0.9791	0.9755	0.9797	0.98	0.9781
28	Yeast1	0.6482	0.6459	0.6577	0.6504	**0.6651**
29	Yeast2vs8	0.7654	0.6836	0.7657	0.7651	**0.7699**
30	Yeast3	0.8179	0.8531	0.8612	0.817	**0.8706**
31	Yeast4	0.666	0.7782	0.7844	0.6658	**0.7879**
32	Yeast5	0.8372	0.9486	**0.961**	0.8373	0.9342
33	Yeast6	0.749	0.8326	**0.8336**	0.7491	0.8242
AVE	**AVERAGE**	0.8166	0.8255	0.8459	0.8162	**0.8490**

Tabla 26: *Resultados de las 33 bases de datos bajo estudio para el clasificador IBk (kNN) con N=1 al aplicar cada variante de remuestreo. bajo el criterio de bondad del área bajo la curva ROC*

Metric	ROC A	IBk3				
Dataset Id	Dataset Name	Sin Remuestreo	RANSUB Size Min	RANSUB Size Max	RANOVER	SMOTE
1	Abalone19	0.4953	0.6355	**0.6742**	0.4952	0.557
2	Abalone9vs18	0.7108	0.6716	0.7373	0.7179	**0.7797**
3	Ecoli0vs1	0.9862	**0.9867**	0.9861	0.9854	0.9854
4	Ecoli1	0.9132	0.9165	**0.9318**	0.9	0.9179
5	Ecoli2	**0.9488**	0.9412	0.9432	0.9447	**0.9448**
6	Ecoli3	0.8689	**0.9118**	0.9071	0.8628	0.8901
7	Ecoli4	0.9224	0.976	**0.9903**	0.9209	0.9187
8	Glass0	0.8626	0.8219	0.8581	0.8603	**0.8669**
9	Glass0123vs456	0.9618	0.9457	0.9628	0.954	**0.9642**
10	Glass1	0.828	0.802	**0.8331**	0.829	0.8267
11	Glass2	0.6087	0.6087	0.6699	0.6632	**0.7657**
12	Glass4	0.9195	0.8902	0.905	0.9282	**0.9579**
13	Glass5	0.9395	0.8385	0.9073	0.9385	**0.9841**
14	Glass6	0.9089	0.9076	0.9192	0.9055	**0.9285**
15	Haberman	**0.6043**	0.5687	0.5951	0.5996	0.5997
16	Iris0	**1**	**1**	**1**	**1**	**1**
17	New-thyroid1	**0.9971**	0.9939	0.9943	0.9936	0.994
18	New-thyroid2	0.9951	0.9829	0.9912	**0.9955**	0.9937
19	Page-blocks0	0.9067	0.9407	**0.946**	0.9031	0.9432
20	Pima	**0.7598**	0.7431	0.756	0.7462	0.7535
21	Segment0	0.9955	0.9942	**0.9967**	0.9957	0.9955
22	Vehicle0	0.9732	0.9438	**0.9804**	0.9722	0.9737
23	Vehicle1	0.7559	0.7394	0.7595	0.7475	**0.775**
24	Vehicle2	0.9819	0.9482	0.9792	0.9803	**0.9821**
25	Vehicle3	0.745	0.7213	0.7572	0.7367	**0.7629**
26	Vowel0	**1**	0.9637	0.9927	**1**	**1**
27	Wisconsin	0.9883	0.987	0.9876	**0.9885**	0.9879
28	Yeast1	0.7194	0.7115	0.7286	0.7142	**0.7308**
29	Yeast2vs8	0.7943	0.7527	0.7823	0.7922	**0.8309**
30	Yeast3	0.9086	0.9268	**0.9374**	0.9001	0.9251
31	Yeast4	0.7457	**0.8676**	0.8669	0.7448	0.8485
32	Yeast5	0.9656	0.9706	**0.9858**	0.9633	0.9781
33	Yeast6	0.8221	**0.8955**	0.8932	0.8201	0.8721
AVE	**AVERAGE**	0.8646	0.8638	0.8835	0.8636	**0.8859**

Tabla 27: *Resultados de las 33 bases de datos bajo estudio para el clasificador IBk (kNN) con N=3 al aplicar cada variante de remuestreo. bajo el criterio de bondad del área bajo la curva ROC*

Metric	ROC A	PART				
Dataset Id	Dataset Name	Sin Remuestreo	RANSUB Size Min	RANSUB Size Max	RANOVER	SMOTE
1	Abalone19	0.5389	0.6531	**0.6554**	0.514	0.6262
2	Abalone9vs18	0.702	0.7032	0.7275	0.7488	**0.7705**
3	Ecoli0vs1	0.9794	0.9605	**0.9829**	0.9794	**0.9829**
4	Ecoli1	**0.9201**	0.8797	0.9012	0.888	0.9163
5	Ecoli2	0.8683	0.8784	**0.891**	0.8732	0.8805
6	Ecoli3	0.8338	0.826	0.852	0.7841	**0.8826**
7	Ecoli4	0.8597	0.8152	**0.9014**	0.8542	0.8986
8	Glass0	**0.8314**	0.7694	0.8119	0.815	0.7999
9	Glass0123vs456	0.9055	0.89	0.8834	0.905	**0.9221**
10	Glass1	0.6825	0.6674	0.7054	**0.7234**	0.7188
11	Glass2	0.6847	0.6038	0.6197	0.6345	**0.6933**
12	Glass4	0.813	0.7882	0.8359	0.8479	**0.8838**
13	Glass5	**0.9846**	0.9059	0.9273	0.8846	0.8907
14	Glass6	0.888	0.892	**0.9156**	0.8762	0.9081
15	Haberman	0.6078	0.6064	0.6361	0.6315	**0.6476**
16	Iris0	**0.984**	0.982	**0.984**	**0.984**	**0.984**
17	New-thyroid1	0.9406	0.9248	0.9148	0.9325	**0.955**
18	New-thyroid2	0.9429	0.9162	0.9363	0.9325	**0.9448**
19	Page-blocks0	0.9632	0.9506	0.9625	0.9294	**0.9725**
20	Pima	**0.7786**	0.7428	0.7701	0.7768	0.7763
21	Segment0	0.9875	0.9791	0.9883	0.9899	**0.9913**
22	Vehicle0	**0.9503**	0.9316	0.9395	0.9397	0.9468
23	Vehicle1	0.7622	0.7149	**0.7695**	0.7295	0.7689
24	Vehicle2	0.9517	0.9402	0.9592	0.963	**0.9637**
25	Vehicle3	0.779	0.7153	0.7683	0.7579	**0.788**
26	Vowel0	0.9507	0.9338	0.948	0.9535	**0.966**
27	Wisconsin	0.9635	0.9516	0.959	0.9627	**0.9642**
28	Yeast1	**0.7663**	0.7321	0.7451	0.7543	0.7554
29	Yeast2vs8	0.5213	0.6812	0.7398	**0.7932**	0.7769
30	Yeast3	**0.9402**	0.8998	0.9323	0.89	0.9362
31	Yeast4	0.7853	0.8127	0.8134	0.6942	**0.8308**
32	Yeast5	0.9124	0.9318	**0.9575**	0.8598	0.9398
33	Yeast6	0.8248	**0.8307**	0.828	0.7629	0.8268
AVE:	**AVERAGE**	0.8426	0.8306	0.8534	0.8353	**0.8639**

Tabla 28: *Resultados de las 33 bases de datos bajo estudio para el clasificador PART al aplicar cada variante de remuestreo. bajo el criterio de bondad del área bajo la curva ROC*

Metric	ROC A	JRIP				
Dataset Id	**Dataset Name**	**Sin Remuestreo**	**RANSUB Size Min**	**RANSUB Size Max**	**RANOVER**	**SMOTE**
1	Abalone19	0.4998	0.6604	**0.7088**	0.5286	0.4849
2	Abalone9vs18	0.6678	0.7102	0.7265	**0.7527**	0.7516
3	Ecoli0vs1	**0.9807**	0.9611	0.9764	0.9789	0.9763
4	Ecoli1	0.8794	0.887	0.8965	**0.8995**	0.8984
5	Ecoli2	**0.8711**	0.836	0.8546	0.8648	0.8684
6	Ecoli3	0.7726	0.8198	**0.8489**	0.7998	0.8452
7	Ecoli4	**0.8986**	0.8182	0.868	0.8712	0.8745
8	Glass0	0.7531	0.7629	0.797	**0.8237**	0.8117
9	Glass0123vs456	0.8801	0.8493	0.8925	0.8811	**0.9177**
10	Glass1	0.6928	0.656	0.6995	0.7226	**0.7347**
11	Glass2	0.5388	0.5739	0.6573	0.6672	**0.7437**
12	Glass4	0.7771	0.6665	0.8036	0.8126	**0.8724**
13	Glass5	0.7902	0.6212	0.8388	**0.9168**	0.9132
14	Glass6	0.8914	0.8494	0.8664	0.8991	**0.9316**
15	Haberman	0.5934	0.6113	**0.6654**	0.6011	0.6594
16	Iris0	0.994	0.989	**0.996**	0.994	0.994
17	New-thyroid1	0.9209	0.894	0.941	0.9552	**0.9648**
18	New-thyroid2	0.9141	0.8921	0.9268	0.9529	**0.9561**
19	Page-blocks0	0.9217	0.931	0.9565	0.9455	**0.9634**
20	Pima	0.7126	0.7224	0.7347	**0.7386**	0.73
21	Segment0	0.9845	0.9777	0.9836	**0.9934**	0.9921
22	Vehicle0	0.928	0.9102	0.942	0.9324	**0.9422**
23	Vehicle1	0.6727	0.7102	0.7362	0.7127	**0.7681**
24	Vehicle2	0.9468	0.9409	0.944	0.9576	**0.9595**
25	Vehicle3	0.6712	0.7079	0.7388	0.7149	**0.7696**
26	Vowel0	0.9469	0.9179	0.9519	0.9622	**0.9734**
27	Wisconsin	0.9662	0.9563	0.9648	0.9598	**0.9668**
28	Yeast1	0.6636	0.6855	0.7091	0.7137	**0.7206**
29	Yeast2vs8	0.7632	0.6149	0.7286	**0.7962**	0.7582
30	Yeast3	0.893	0.8999	0.9173	0.8913	**0.9343**
31	Yeast4	0.6585	0.8207	**0.8519**	0.7088	0.7909
32	Yeast5	0.8878	0.9402	**0.947**	0.8792	0.9315
33	Yeast6	0.7339	0.8431	**0.8612**	0.7424	0.8032
AVE	**AVERAGE**	0.8081	0.8072	0.8464	0.8355	**0.8546**

Tabla 29: *Resultados de las 33 bases de datos bajo estudio para el clasificador JRIP al aplicar cada variante de remuestreo. bajo el criterio de bondad del área bajo la curva ROC*

8 Referencias

[1] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., . . . Steinberg, D. (2008). Top 10 algorithms in data mining. Knowledge and Information Systems, 14(1), 1-37. doi:10.1007/s10115-007-0114-2

[2] QUINLAN, J. R. (1990). Decision trees and decision-making. Ieee Transactions on Systems Man and Cybernetics, 20(2), 339-346. doi:10.1109/21.52545

[3] Deng, Z., Zhu, X., Cheng, D., Zong, M., & Zhang, S. (2016). Efficient kNN classification algorithm for big data. Neurocomputing, 195, 143-148. doi:10.1016/j.neucom.2015.08.112

[4] Vapnik, V. N. (1999). An overview of statistical learning theory. IEEE Transactions on Neural Networks, 10(5), 988-999. doi:10.1109/72.788640

[5] Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. Machine Learning, 29(2-3), 131-163. doi:10.1023/A:1007465528199

[6] Grzymala-Busse, J. W., Stefanowski, J., & Wilk, S. (2005). A comparison of two approaches to data mining from imbalanced data. Journal of Intelligent Manufacturing, 16(6), 565-573. doi:10.1007/s10845-005-4362-2

[7] Rhusi, L., Snehlata, S. D., Latesh, M. Class Imbalance Problem. International Journal of Computer Science and Network (IJCSN). Volume 2, Issue 1 (2013). ISSN 2277-5420

[8] Barandela, R., Sánchez, J.S, García, V., Rangel, E. Strategies for learning in class imbalance problems, Pattern Recognition 36 (3) (2003) 849–851.

[9] Q. Yang, X. Wu, 10 challenging problems in data mining research, International Journal of Information Technology and Decision Making 5 (4) (2006) 597–604.

[10] Bi, J., & Zhang, C. (2018). An empirical comparison on state-of-the-art multi-class imbalance learning algorithms and a new diversified ensemble learning scheme. Knowledge-Based Systems, 158, 81-93. doi:10.1016/j.knosys.2018.05.037

[11] Japkowicz, N. (2000). In Arabnia H. R. (Ed.), The class imbalance problem: Significance and strategies. ATHENS; 115 AVALON DR, ATHENS, GA 30606 USA: C S R E A PRESS.

[12] G.M. Weiss, Mining with rarity: a unifying framework, SIGKDD Explorations 6 (1) (2004) 7–19.

[13] Nguyen, Giang & Bouzerdoum, Abdesselam & Phung, Son. (2009). Learning Pattern Classification Tasks with Imbalanced Data Sets. 10.5772/7544.

[14] Sun, Y., Kamel, M. S., Wong, A. K. C., & Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition, 40(12), 3358-3378. doi:10.1016/j.patcog.2007.04.009

[15] Frank, E., Hall, M., Holmes, G., Kirkby, R., Pfahringer, B., Witten, I. H., . . . Trigg, L. (2010). In Maimon O. R.,L. (Ed.), Weka-A machine learning workbench for data mining. NEW YORK; 233 SPRING STREET, NEW YORK, NY 10013, UNITED STATES: SPRINGER. doi:10.1007/978-0-387-09823-4_66

[16] Frank, E., Hall, M., Trigg, L., Holmes, G., & Witten, I. H. (2004). Data mining in bioinformatics using weka. Bioinformatics, 20(15), 2479-2481. doi:10.1093/bioinformatics/bth261

[17] Arganda-Carreras, I., Kaynig, V., Rueden, C., Eliceiri, K. W., Schindelin, J., Cardona, A., & Seung, H. S. (2017). Trainable weka segmentation: A machine learning tool for microscopy pixel classification. Bioinformatics, 33(15), 2424-2426. doi:10.1093/bioinformatics/btx180

[18] Yadav, A. K., Malik, H., & Chandel, S. S. (2014). Selection of most relevant input parameters using WEKA for artificial neural network based solar radiation prediction models. Renewable & Sustainable Energy Reviews, 31, 509-519. doi:10.1016/j.rser.2013.12.008

[19] Bouckaert, R. R., Frank, E., Hall, M. A., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2010). WEKA-experiences with a java open-source project. Journal of Machine Learning Research, 11, 2533-2541.

[20] bin Othman, M. F., & Shan Yau, T. M. (2007). Comparison of different classification techniques using WEKA for breast cancer. 3rd Kuala Lumpur International Conference on Biomedical Engineering 2006, 15, 520-+.

[21] Garg, T., & Khurana, S. S. (2014). Comparison of classification techniques for intrusion detection dataset using WEKA

[22] Goyal, A., Khandelwal, I., Anand, R., Srivastava, A., & Swarnalatha, P. (2018). A comparative analysis of the different data mining tools by using supervised learning algorithms. Proceedings of the Eighth International Conference on Soft Computing and Pattern Recognition (Socpar 2016), 614, 105-112. doi:10.1007/978-3-319-60618-7_11

[23] Kowalczyk, A., Raskutti, B., & Ferra, H. (2004). Exploring potential of leave-one-out estimator for calibration of SVM in text mining. Advances in Knowledge Discovery and Data Mining, Proceedings, 3056, 361-372.

[24] Williams, D. P., Myers, V., & Silvious, M. S. (2009). Mine classification with imbalanced data. Ieee Geoscience and Remote Sensing Letters, 6(3), 528-532. doi:10.1109/LGRS.2009.2021964

[25] Zhao, Z. (2009). A novel modular neural network for imbalanced classification problems. Pattern Recognition Letters, 30(9), 783-788. doi:10.1016/j.patrec.2008.06.002

[26] Huang, S. (2015). Supervised feature selection: A tutorial. Artificial Intelligence Research – March 2015. doi: 10.5430/air.v4n2p22

[27] Blagus,R., Lusa, L. Class prediction for high-dimensional class-imbalanced data. MC Bioinformatics 11, 523 (2010). https://doi.org/10.1186/1471-2105-11-523

[28] Reunanen J. Overfitting in Making Comparisons between Variable Selection Methods. Journal of Machine Learning Research. 2003; 3(Mar): 1371-1382.

[29] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321-357. doi:10.1613/jair.953

[30] Fernandez, A., Garcia, S., Herrera, F., & Chawla, N. V. (2018). SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. Journal of Artificial Intelligence Research, 61, 863-905. doi:10.1613/jair.1.11192

[31] Han, H., Wang, W. Y., & Mao, B. H. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. Advances in Intelligent Computing, Pt 1, Proceedings, 3644, 878-887. doi:10.1007/11538059_91

[32] Yen, S., & Lee, Y. (2009). Cluster-based under-sampling approaches for imbalanced data distributions. Expert Systems with Applications, 36(3), 5718-5727. doi:10.1016/j.eswa.2008.06.108

[33] Yoon, K., & Kwek, S. (2005). In Nedjah, N Mourelle, LM Vellasco, MMB Abraham, A Koppen,M. (Ed.), An unsupervised learning approach to resolving the data imbalanced issue in supervised learning problems in functional genomics. LOS ALAMITOS; 10662 LOS VAQUEROS CIRCLE, PO BOX 3014, LOS ALAMITOS, CA 90720-1264 USA: IEEE COMPUTER SOC.

[34] Wilson, D. R., & Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. Machine Learning, 38(3), 257-286. doi:10.1023/A:1007626913721

[35] Prati, R. C., Batista, G., & Monard, M. C. (2004). Learning with class skews and small disjuncts. Advances in Artificial Intelligence - Sbia 2004, 3171, 296-306.

[36] Weiss, G. M., & Provost, F. (2003). Learning when training data are costly: The effect of class distribution on tree induction. Journal of Artificial Intelligence Research, 19, 315-354. doi:10.1613/jair.1199

[37] Albisua, I., Arbelaitz, O., Gurrutxaga, I., Martin, J. I., Muguerza, J., Perez, J. M., & Perona, I. (2010). Obtaining optimal class distribution for decision trees: Comparative analysis of CTC and C4.5. Current Topics in Artificial Intelligence, 5988, 101-110.

[38] Japkowicz, N. (2013) Assessment metrics for imbalanced learning, in Imbalanced Learning: Foundations, Algorithms, and Applications. Wiley IEEE Press, 2013, pp. 187-210. Doi: 10.1002/9781118646106.ch8

[39] N.V. Chawla, N. Japkowicz and A. Kolcz, (2004) Editorial: Special issue on learning from imbalanced data sets, SIGKDD Explorations, 6 1-6. doi: 10.1145/1007730.1007733

[40] Provost, F. and Domingos, P. (2003) Tree induction for probability-based ranking. Machine Learning, 52,199-215. Doi: 10.1023/A:1024099825458

[41] Rakotomamonyj,A. (2003) Optimizing area under ROC with SVMs, in J. Hernandez-Orallo, C. Ferri, N. Lachiche and P. A. Flach (Eds.) 1st Int. Workshop on ROC Analysis in Artificial Intelligence (ROCAI 2004), Valencia, Spain, 2004, pp. 71-80.

[42] Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology, 143, 29–36. doi: 10.1148/radiology.143.1.7063747

[43] Carletta, J. (1996). Assessing Agreement on Classification Tasks: The Kappa Statistic. Computational Linguistics, Volume 22, Number 2, June 1996. url = "https://aclanthology.org/J96-2004"

[44] Hossin, M., Sulaiman. M.N. (2015) A review on evaluation metrics for data classification evaluations. Int. J. Data Min. Knowl. Manage. Process, 5

[45] Williams, Christopher K. I. (2021). The Effect of Class Imbalance on Precision-Recall Curves. Neural Computation. 33 (4): 853–857. doi:10.1162/neco_a_01362.

[46] Albisua, I., Arbelaitz, O., Gurrutxaga, I., Lasarguren, A., Muguerza, & J., Perez, J. M. (2012) The quest for the optimal class distribution: an approach for enhancing the effectiveness of learning via resampling methods for imbalanced data sets Prog Artif Intell (2013) 2:45-63. doi: 10.1007/s13748-012-0034-6

[47] Alcala-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., Garcia, S., Sanchez, L., & Herrera, F. (2011). KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic and Soft Computing, 17(2-3), 255-287.

[48] Webb, I. (2000) Multiboosting: A technique for combining boosting and wagging, Machine Learning, 40: 159-197

[49] Demsar, J. (2006) Statistical Comparisons of Classifiers over Multiple Data Sets, Journal of Machine Learning Research, vol. 7, pp. 1–30, 2006. 15, 16

[50] Benavoli, Alessio & Corani, Giorgio & Demsar, Janez & Zaffalon, Marco. (2016). Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis. Journal of Machine Learning Research. 18.