ALGORITMO PARA LA SELECCIÓN DE INSTANCIAS EN PROBLEMAS DE CLASIFICACIÓN BASADO EN ARREGLOS DE COBERTURA

Universidad
del Cauca

Jhonattan Solarte Martínez
Tesis en Maestría en Computación

Director: PhD. Carlos Alberto Cobos Lozada
Codirectora: PhD. Martha Eliana Mendoza Becerra

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Maestría en Computación
Popayán
2019

JHONATTAN SOLARTE MARTINEZ

ALGORITMO PARA LA SELECCIÓN DE INSTANCIAS EN PROBLEMAS DE CLASIFICACIÓN BASADO EN ARREGLOS DE COBERTURA

Tesis presentada a la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca para la obtención del título de

Magíster en Computación

Director: PhD. Carlos Alberto Cobos Lozada
Codirectora: PhD. Martha Eliana Mendoza Becerra

Popayán 2019

Resumen Estructurado

La extracción de conocimiento en bases de datos soporta la toma de decisiones y la mejora de diversas tareas del quehacer humano. Un Modelo de Aprendizaje Supervisado (MAS) se obtiene de un conjunto de datos, que representan situaciones específicas de la vida cotidiana, los cuales han sido recopilados y registrados (datos de entrenamiento) con dos componentes principales, una variable de interés denominada variable de salida, objetivo o de respuesta y otras variables denominadas variables de entrada o explicativas. El proceso de obtención de un MAS para un problema del mundo real tiene una fase de "Preprocesamiento de los datos", donde se aplica la selección de instancias (reducción de datos), si el dataset lo amerita. La selección de instancias busca obtener datos de calidad, que al ser utilizados en la etapa de modelamiento permita definir modelos de calidad similar o superior con menos cantidad datos, en menor tiempo y esfuerzo computacional.

Los Covering Array (CA) o algoritmos de cobertura son objetos matemáticos que ha sido usados en el diseño de experimentos con aplicaciones en biología, análisis de fallas en ingeniería y recientemente en las pruebas de calidad de software y hardware. Sin embargo, a la fecha no han sido utilizados para la selección de instancias en datasets. Por lo anterior, en esta investigación se propone un algoritmo basado en Covering Arrays binarios, que permite seleccionar instancias o filas de datasets que se utilizan en procesos de clasificación (aprendizaje supervisado), propuesta denominada ISCA.

La evaluación de la propuesta se realizó con 26 datasets del repositorio de aprendizaje de máquina de la Universidad de California en Irvine, usando dos medidas de comparación: la calidad de la clasificación (menor porcentaje de error en la clasificación) y el porcentaje de reducción de instancias. Los datasets reducidos con ISCA se usaron con dos clasificadores (1NN y C4.5) y la calidad de los modelos se comparó frente al uso del dataset completo (original), como resultado se observó que ISCA version 1 permite reducir el dataset en promedio a un 39,5% del tamaño original y mejora la calidad de la clasificación en promedio un 2,5 %.

ISCA versión 2 se comparó con cuatro algoritmos del estado del arte de selección (reducción) de instancias, CNN, ELH, ENN y RMHC, permitiendo obtener una reducción de instancias mucho mayor, de 11,5% en promedio, mientras que para los otros fue de 38,3%, 25,7%, 77,2% y 9,8% respectivamente. En cuanto al promedio del % de error, ISCA obtiene un valor de 19,460% mientras que el mejor resultado de los otros algoritmos comparados lo obtiene CNN con 12,622. Los resultados son prometedores y alientan el desarrollo de nuevos trabajos de investigación en el área.

Palabras clave: Algoritmos de clasificación, arreglos de cobertura, Selección de instancias, KNN, C4.5, RMHC (Random Mutation Hill Climbing), ELH (Encoding Length Heuristic), CNN (Condensed Nearest Neighbor Rule), ENN (Edited Nearest Neighbour).

Structured Abstract

Knowledge Discovery in Databases (KDD) supports decision-making and the improvement of various tasks of human activity. A Supervised Learning Model (SLM) is obtained from a dataset which represents specific daily life situations. The data has been collected and recorded (training data) with two main components: a variable of interest called the output, objective or response variable and other variables called the input or explanatory variables. The process of obtaining an SLM for a real-life problem has a “Data preprocessing” phase, in which instance selection is applied if the dataset needs it. Instance selection seeks to obtain quality data, which, when used in the modeling stage, allows the definition of models of similar or higher quality with less data, using less time and computational effort.

Covering Arrays (CA) are mathematical objects that have been used in the design of experiments with applications in biology, analysis of engineering failures, and recently in the quality testing of software and hardware. However, they have not been used to date for instance selection in datasets. Therefore, this research proposes an algorithm based on binary Covering Arrays, which allows the selection of instances or rows of datasets that are used in classification processes (supervised learning). This is a proposal called ISCA.

The evaluation of the proposal was carried out with 26 datasets from the machine learning repository of the University of California at Irvine, using two comparison measures: the quality of the classification (lower error percentage in the classification) and the percentage of instance reduction. The datasets reduced with ISCA were used with two classifiers (1NN and C4.5) and the quality of the models was compared against the use of the complete (original) dataset. As a result, it was observed that on average ISCA version 1 allows the reduction of the dataset to 39.5% of the original size and improves the quality of the classification by 2.5%.

ISCA version 2 was compared with four state-of-the-art instance selection (reduction) algorithms: CNN, ELH, ENN and RMHC, allowing for a much greater instance reduction, 11.5% on average, while with the others it was 38.3%, 25.7%, 77.2%, and 9.8%, respectively. Regarding the average % error, ISCA obtains a value of 19.460% while the best result of the other algorithms is obtained by CNN, which is 12.622%. The results are promising and encourage the implementation of new research projects in the area.

Keywords: Classification algorithms, covering arrays, Instance selection, KNN, C4.5, RMHC (Random Mutation Hill Climbing), ELH (Encoding Length Heuristic), CNN (Condensed Nearest Neighbor Rule), ENN (Edited Nearest Neighbor).

Tabla de Contenido

Lista de Tablas

Lista de Ilustraciones

1. INTRODUCCIÓN

La extracción de conocimiento en bases de datos (Knowledge Discovery in Databases, KDD) es el área que permite extraer conocimiento útil de grandes conjuntos de datos, los cuales primero deben ser preprocesados o depurados. Uno de los pasos más importantes del preprocesamiento en KDD es la selección de instancias o selección de filas.

El problema de selección de instancias, busca eliminar instancias ruidosas, repetidas, erróneas o poco útiles, creando un conjunto de datos más pequeño que permite predecir la clase de una instancia nueva con la misma precisión (o más alta) que con el conjunto original completo [1]. Esto, está directamente relacionado con la reducción de datos y se vuelve cada vez más importante en muchas aplicaciones, debido a la necesidad de eficiencia de procesamiento y de almacenamiento [2].

El algoritmo kNN, es uno de los más importantes en el aprendizaje automático [3]. Este algoritmo calcula la distancia entre una instancia a clasificar y cada instancia del conjunto de entrenamiento. A la nueva instancia a clasificar se le asigna la clase de la instancia vecina más cercana. El algoritmo kNN no es practico para trabajar con dataset muy grandes, ya que estos demandan muchos recursos computacionales en almacenamiento y tiempo de procesamiento [4]. En el estado del arte se presentan diferentes estrategias o métodos para la selección de instancias, muchos de ellos aplicados a kNN. Estos métodos se organizan en dos grandes grupos, los basados en muestreo y los basados en selección de prototipos.

Teniendo en cuenta que uno de los enfoques de solución para la selección de instancias es el muestreo, en este trabajo se consideró el uso de los arreglos de cobertura como una alternativa factible para proponer una nueva solución. Los arreglos de cobertura denotados como CA (Covering Arrays), son objetos combinatorios derivados de los arreglos ortogonales que pueden ser usados y aplicados para diversos fines [5], entre ellos, el diseño de experimentos y la automatización de pruebas de hardware y software.

Los CA permiten generar el menor número de casos de prueba para cubrir todos los conjuntos de interacciones entre los parámetros objeto de estudio. Estos arreglos tienen cardinalidad mínima (reducen al mínimo el número de casos de prueba) y cobertura máxima (garantizan cubrir todas las combinaciones entre los parámetros de entrada basado en un parámetro denominado fuerza, que define el nivel de la interacción que se desee cubrir) [6]. Por esto último, es que en esta tesis se les consideró como una estrategia viable para la selección de instancias.

1.1 PLANTEAMIENTO DEL PROBLEMA

Hoy en día, a nivel personal y empresarial se realiza una acumulación exponencial de datos e información como resultado del avance en los procesos informáticos y tecnológicos (registro de navegación en internet, comentarios en Twitter, Facebook y otras redes sociales, perfiles de compras, datos de campañas de marketing, resultados de experimentos en investigaciones en diferentes áreas, entre otros). Estos avances han

propiciado así mismo el desarrollo y perfeccionamiento de bases de datos cada vez más eficientes que facilitan al ser humano el tratamiento de grandes cantidades de datos. En este contexto, uno de los retos más importantes en el campo de la tecnología consiste en el análisis inteligente de estos grandes volúmenes de datos que permita la extracción de información útil para soportar la toma de decisiones y aumentar la eficiencia de diferentes procesos operativos y de gestión que se requieren en las sociedad del siglo XXI también conocida como la "Era de los Datos" o de la "Cuarta Revolución Industrial" [7].

La extracción de conocimiento en bases de datos (KDD) es el área de estudio que busca extraer conocimiento útil de grandes conjuntos de datos, para ayudar o soportar a las personas a realizar tareas de una manera más satisfactoria y eficiente [4]. En esta área, los datos se utilizan luego de preprocesarlos o depurarlos, y al igual que su análisis, no se hace en forma manual ya que es poco viable [3]. Las actividades de preprocesamiento incluyen la limpieza de los datos, la eliminación de datos redundantes, irrelevantes o con ruido, la solución de inconsistencias, la integración y transformación de datos, entre otros [8]. Luego con los datos procesados se realiza la ejecución de los algoritmos de modelamiento, aquellos que definen los modelos de clasificación, regresión, agrupamiento, reglas de asociación y de análisis de series temporales [4]. En esta investigación, el interés se centró únicamente en la selección de instancias para modelos de clasificación.

Estas actividades de preprocesamiento cada día son más complejas, ya que a pesar de que crece el poder de cómputo y los algoritmos de modelamiento son más eficientes, el ritmo de crecimiento de los datos es mucho mayor [2]. La **selección de instancias** como una actividad de **reducción de datos** es uno de los problemas más relevantes en la investigación actual, como una estrategia clave en el manejo de estos grandes volúmenes de datos [9].

La selección de instancias o selección de filas se enfoca en suprimir registros o datos que son poco útiles, redundantes o irrelevantes, sintetizando los datos más importantes para mantener o mejorar la calidad del modelo que se busca realizar, reducir el espacio de almacenamiento de los datos y reducir el tiempo de construcción de dicho modelo [10]. En cualquier caso, siempre se debe tener en consideración, que aplicar un algoritmo de reducción de instancias tendrá un coste computacional, y la disminución de datos, a veces produce menor calidad en la clasificación, por eso debe existir una relación entre la cantidad de los datos y la calidad de los resultados [4].

Los algoritmos de selección de instancias inician su proceso desde un conjunto de datos de entrenamiento T, del cual el algoritmo elimina las instancias que él considera poco relevantes, dando como salida un subconjunto S de T. La selección de dichas instancias es un proceso complejo, ya que seleccionar el subconjunto perfecto S de T, usando un algoritmo de fuerza bruta (esto es, evaluando todos los posibles subconjuntos de registros), el cual tiene un coste computacional de orden $O(2^n)$, en donde n es el cardinal de T (número de registros en T) es inviable para conjuntos de datos grandes, por esta razón, es necesario encontrar mecanismos que resuelvan este problema en tiempo polinomial [11].

La literatura reporta diversos algoritmos para reducción de instancias, que se basan en varios enfoques como muestreo, selección de prototipos, aprendizaje activo y boosting, los cuales ofrecen resultados prometedores [12][2]. Sin embargo, estás técnicas aún son susceptibles de mejora (medida en exactitud de la clasificación, porcentaje de reducción del conjunto de datos o tiempo de procesamiento), realizando modificaciones a los algoritmos existentes o explorando nuevos enfoques.

Por otro lado, los arreglos de cobertura (Covering Arrays, CAs) son objetos matemáticos que permiten, gracias a sus propiedades, hacer la evaluación de la interacción de diferentes parámetros con la mayor cobertura y el menor esfuerzo posible basado en un parámetro denominado fuerza, que es una medida de la interacción entre los parámetros que se busca muestrear al menos una vez. Su uso ha logrado resultados muy positivos en el diseño de experimentos, las pruebas de hardware, las pruebas de software y la selección de atributos [13], por lo que se les considera una alternativa viable y posiblemente más eficiente para el proceso de selección de instancias que las propuestas existentes.

Partiendo de lo anterior, en este trabajo de investigación se planteó la siguiente pregunta de investigación ¿Cómo adaptar los Covering Arrays (CAs) como mecanismo base de un algoritmo para selección de instancias en problemas de clasificación, con el fin de evaluar la efectividad (medida en la exactitud de clasificación y el porcentaje de reducción de instancias) en comparación con algoritmos del estado del arte?

La hipótesis principal de esta investigación fue:
H1: adaptar los Covering Arrays (CAs) como mecanismo base de un algoritmo para selección de instancias en problemas de clasificación, mejora o iguala la efectividad (medida en la exactitud de clasificación y el porcentaje de reducción de instancias) en comparación con dos de los mejores algoritmos reportados en el estado del arte.

Y de esta forma la hipótesis alterna o nula fue:
H0: adaptar los Covering Arrays (CAs) como mecanismo base de un algoritmo para selección de instancias en problemas de clasificación, no mejora ni iguala la efectividad (medida en la exactitud de clasificación y el porcentaje de reducción de instancias) en comparación con dos de los mejores algoritmos reportados en el estado del arte.

1.2 APORTES

Desde el punto de vista de investigación, los aportes de esta tesis se enmarcan en la consecución de nuevo conocimiento para la comunidad académica y científica, en el área de aprendizaje de maquina y minería de datos, relacionado con la propuesta de un nuevo algoritmo de selección o reducción de instancias basado en arreglos de cobertura (covering arrays) para resolver problemas de clasificación, que es competitivo con los algoritmos del estado del arte. Esta propuesta es nueva, ya que el autor no conoce una investigación similar, que este reportada en las bases de datos de Scopus, ACM, IEEE, ScienceDirect y Springer Link). Además, establece opciones de mejora a la propuesta que el Grupo de I+D en Tecnologías de la Información (GTI) de la Universidad del Cauca espera abordar en el futuro cercano.

Desde el punto de vista de innovación, se hace entrega en el CD del código fuente del algoritmo propuesto para que los investigadores puedan repetir los experimentos, además para que los expertos en minería de datos lo puedan empezar a utilizar en sus proyectos.

1.3 OBJETIVOS

A continuación, se presentan los objetivos de la tesis tal como fueron aprobados por el Consejo de Facultad de la Facultad de Ingeniería Electrónica y Telecomunicaciones.

1.3.1 Objetivo general

Proponer un método de selección de instancias en modelos de MAS que resuelven problemas de clasificación, basado en arreglos de cobertura y comparar su efectividad (medida por la calidad de la clasificación y el porcentaje de reducción de instancias) frente a dos algoritmos del estado del arte.

1.3.2 Objetivos Específicos

- Definir los conceptos más importantes y el estado del arte en el proceso de selección de instancias, en el marco de modelos de aprendizaje supervisado (algoritmos de clasificación), usando la propuesta de investigación documental de Serrano [15].

- Definir un método para selección de instancias basado en arreglos de cobertura, teniendo en cuenta diferentes niveles de interacción (fuerza) y con la habilidad de trabajar sobre grandes volúmenes de datos buscando mantener o mejorar la calidad de la clasificación y el porcentaje de reducción de instancias.

- Evaluar y comparar el método propuesto frente a mínimo dos algoritmos del estado del arte (RMHC, Explore, LVQ, MC1, DEL o C-Pruner), usando al menos 20 dataset reconocidos por la comunidad académica y científica (extraídos en su mayoría del repositorio de la UCI), con dos métricas de efectividad, la primera basada en el porcentaje de instancias correctamente clasificadas para evaluar la calidad de la clasificación y la segunda es la tasa de reducción de instancias.

1.4 RESULTADOS OBTENIDOS

Al terminar esta tesis de investigación se obtuvieron los siguientes resultados:

- Monografía del trabajo de grado. Corresponde al presente documento, que recopila los logros más importantes del trabajo de investigación.
- CD con el código fuente del algoritmo para selección de instancias utilizando Covering Array, implementado en java y soportado por Weka (ver **Anexo 1**).
- Artículo presentado en el II Congreso Internacional en Ciencias de la Computación INCICS 2019 realizado del 16 al18 de octubre de 2019 en Ibarra, Ecuador. Las memorias de este evento se encuentran en turno de publicación en la revista RISTI (Revista Ibérica de Sistemas e Tecnologias de Informação) indexada en categoría B del PUBLINDEX de COLCIENCIAS. La referencia del artículo a la fecha es: Jhonattan Solarte-Martínez, Carlos Cobos, Martha Mendoza (2019). Algoritmo para la selección de instancias en problemas de clasificación basado en arreglos de cobertura. Revista Ibérica de Sistemas e Tecnologias de Informação, In Press (ver **Anexo 2**).

1.5 ORGANIZACIÓN DEL RESTO DEL DOCUMENTO

El resto de la monografía está estructurada de la siguiente manera:

Capítulo 2. Muestra el marco teórico, que incluye los conceptos necesarios para abordar el problema de investigación y entender el resto de la monografía.

Capítulo 3. Presenta el estado del arte, mostrando las técnicas y algoritmos más representativos la literatura para selección de instancias. Haciendo énfasis en las dos

estrategias principales: Muestreo y selección de prototipos (vecinos cercanos, eliminación ordenada, algoritmos evolutivos y codificación de longitud).

Capítulo 4. Presenta la primera versión del algoritmo propuesto para la reducción de instancias, basado en Covering Array. Se evalúan los resultados de la experimentación en diversos datasets utilizando niveles de fuerza 2, 3, 4, 5. Y se comparan los resultados de la propuesta usando dos clasificadores del estado del arte, 1NN y C.45 (J48).

Capítulo 5. Presenta la segunda versión del algoritmo propuesto, utilizando ejecuciones iterativas, y un umbral de perdida de error definido por el usuario. Se evalúan los diferentes resultados en comparación con otros algoritmos de selección de instancias del estado del arte: RMHC, ELH, CNN.

Capítulo 6. Presenta las conclusiones del trabajo realizado y el trabajo futuro que el grupo de investigación espera adelantar en el corto plazo.

Capítulo 7. Presenta las referencias que soportaron el trabajo realizado.

Página intencionalmente dejada en blanco

2. MARCO TEÓRICO

Para una mejor comprensión de los temas abordados en esta investigación, a continuación, se hace una introducción de los siguientes conceptos: modelos de aprendizaje supervisado y se enmarca el proceso de selección de instancias en KDD, finalmente, se introducen los arreglos de cobertura.

2.1 Modelos de aprendizaje supervisado (MAS)

El aprendizaje automático en la actualidad es una de las áreas de más rápido crecimiento en la informática, con aplicaciones de gran alcance [16]. Un MAS se obtiene de un conjunto de datos, que representan situaciones específicas de la vida cotidiana que han sido recopilados y registrados (datos de entrenamiento) con dos componentes principales, una variable de interés denominada variable de salida, objetivo o de respuesta y otras variables denominadas variables de entrada o explicativas [17][18]. Las relaciones entre las variables explicativas y la variable de salida se maximizan por medio de una función, que corresponde al MAS, que emplea algoritmos de optimización tales como las redes neuronales, los árboles de clasificación, las máquinas de soporte vectorial, entre otros [16][19]. Es así como un MAS busca deducir con base en los datos de entrenamiento la relación de las variables de entrada con la variable de salida [20][21][22].

Los MAS se dividen en dos grupos principales: Clasificación, para variables de salida categóricas (discretas) y Regresión, para variables de salida cuantitativas (numéricas continuas).

La tarea de clasificación es una de las tareas más frecuentemente realizadas en los proyectos de minería de datos y Big data. Aunque existe una amplia cantidad de algoritmos para resolver problemas de clasificación, a continuación se mencionan algunos de ellos, que tienen diferentes enfoques: Tablas de decisión, redes neuronales, Jrip, arboles de decisión, Naïve Bayes, Máquinas de soporte vectorial y Random Forest [16].

El proceso de obtención de un MAS para un problema del mundo real se resume en la **Ilustración 1**, donde se resalta en rojo la fase de “Preprocesamiento de los datos”, donde se aplica la actividad de reducción de datos (selección de instancias) si el dataset lo amerita.

2.2 Preprocesamiento de los datos

Esta etapa busca obtener datos de calidad que al ser utilizados en el proceso de modelamiento permitan definir modelos de calidad superior. Cabe resaltar que el preprocesamiento de los datos es una de las fases que demanda más tiempo en el KDD [23]. La importancia de la preparación de los datos se basa en [24] que los datos recolectados pueden tener inconsistencias, valores erróneos, datos faltantes, con ruido o sin importancia (irrelevantes). Al limpiar los datos, se generan conjuntos de datos que son de menor tamaño que el dataset original, y con ello se puede mejorar el rendimiento del algoritmo, medido por la calidad del modelo y el tiempo usado para la creación del modelo.

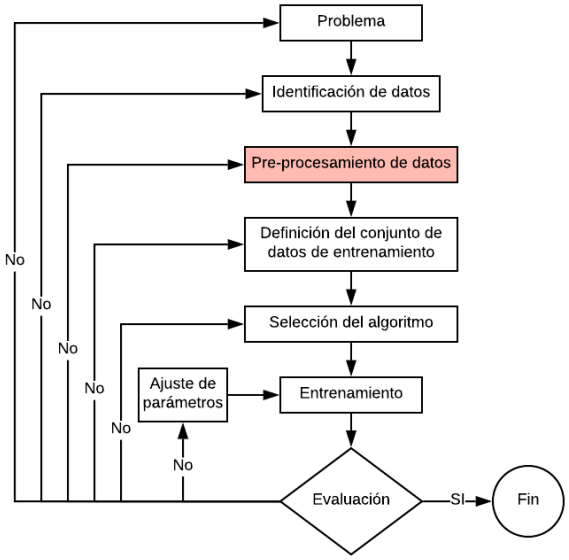


Ilustración 1. Proceso de aprendizaje supervisado. Ilustración propia.

Con el fin de incrementar la calidad de los datos, en la fase de preprocesamiento se realizan las actividades que se presentan en la **Ilustración 2** y se describen a continuación.

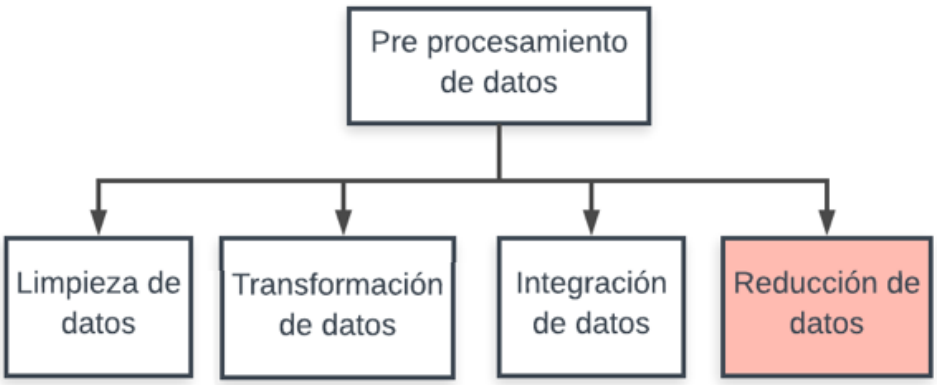


Ilustración 2. Preprocesamiento de datos. Ilustración propia.

- Limpieza de datos: Donde se utilizan técnicas de análisis selectivo para incrementar la calidad de los datos a la requerida y eliminar datos innecesarios o errados [25]. Se completan datos faltantes, con valores coherentes, de acuerdo con la media, la moda o la mediana, entre otros métodos. Además se eliminan los valores atípicos (outliers), y se resuelven las inconsistencias [11].

- Integración de datos: consiste en la combinación de diversas tablas o fuentes para la creación de nuevos registros o valores. Al unir dos o más registros que poseen información similar se obtiene como resultado una tabla que integra las características de las tablas utilizadas, proceso que se denomina combinación [26][27].

- Transformación de datos: Su objetivo principal es la modificación de los datos a utilizar, mediante la normalización, generación de jerarquía de conceptos, discretización, entre otros [11].

- Reducción de datos: El objetivo principal de esta actividad es reducir los volúmenes extensos de datos, buscando que no se pierda calidad en los resultados del MAS [2].

2.3 Reducción de datos

Aplicar MAS desde conjuntos con grandes cantidades de datos conlleva a dificultades como el incremento en el tiempo de entrenamiento o de respuesta si el MAS es perezoso (lazy) como kNN y además el incremento en el almacenamiento de los datos. Al utilizar grandes cantidades de datos, es más probable que las instancias contengan ruido, lo cual afecta negativamente la calidad de los modelos; A veces, la solución que resulta de la extracción de modelos complejos dificulta el entendimiento para la mente humana, puesto que regularmente es difícil comprender algunos resultados. Por tanto, es más fácil entender MAS cuando el tamaño de este es menor [4]. En la **Ilustración 3** se listan las principales técnicas para la reducción de los datos.

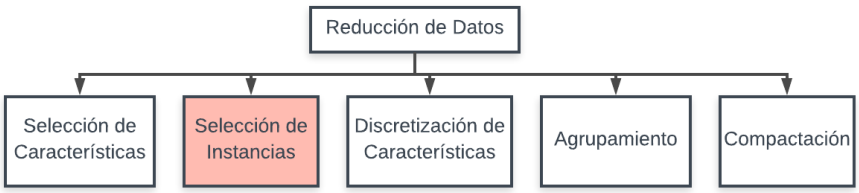


Ilustración 3. Reducción de datos. Ilustración propia.

Este trabajo de grado se enfoca en la selección de instancias o filas. Ya que, con la reducción del conjunto de los datos también se reduce el costo de procesamiento, complejidad y tiempo de definición del MAS [28][4], para ello se deben elegir las filas más relevantes y que mejor representen al conjunto total, y así poder obtener resultados similares o incluso mejores que si se utilizara el conjunto de datos total [4].

2.4 Selección de instancias en MAS

El problema de la selección de instancias se puede definir como "el aislamiento del conjunto más pequeño de instancias que permite predecir la clase de una instancia de consulta con una precisión similar (o más alta) que el conjunto original" [29]. Esto, está directamente relacionado con la reducción de datos y se vuelve cada vez más importante en muchas aplicaciones, debido a la necesidad de eficiencia de procesamiento y eficiencia de almacenamiento [2][8].

En la **Ilustración 4**, se resume la selección de instancias, en donde se identifica un subconjunto (*S*) de las instancias de un conjunto de datos de entrenamiento (*T*), para operar con ellas en lugar de trabajar con los datos completos.

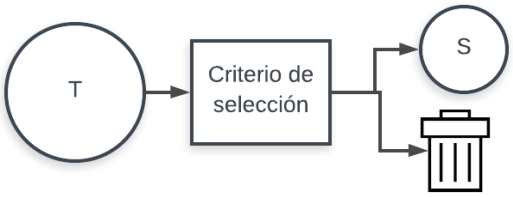


Ilustración 4. Esquema selección de instancias. Ilustración propia.

En muchas situaciones se tendrá un mejor rendimiento del algoritmo de clasificación usando el subconjunto seleccionado que usando el conjunto completo. Esto sucede, porque algunos algoritmos de selección de instancias logran eliminar valores atípicos, ruido u otra información poco relevante, que afectan negativamente los resultados.

2.5 Arreglos de cobertura o covering arrays (CA)

Los CAs son usados en diseño de experimentos, biología, análisis de fallas en ingeniería y otras tareas [30], pero actualmente son muy utilizados en las pruebas de calidad de software y hardware, buscando con estas pruebas, proveer productos de calidad sin incurrir en costos exagerados en su realización. Los CA son una alternativa a las pruebas exhaustivas, ya que dependiendo de un parámetro denominado fuerza (t) del arreglo se puede definir el nivel de cobertura de las interacciones entre los parámetros que se desea evaluar y con ello definir el menor número de casos a realizar.

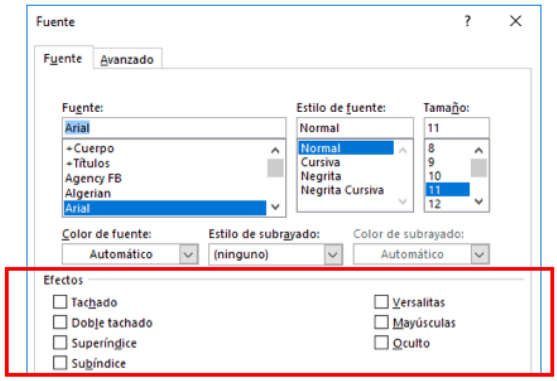
Un CA se expresa a partir de la notación CA (N; t, k, v), que representa una matriz de tamaño N × k, donde N se refiere al número de pruebas, k el número de parámetros, factores o variables, v el alfabeto que indica el número posible de valores que puede tomar cada componente. La **Ilustración 5** muestra el CA (5; 2, 4, 2) que tiene fuerza dos (t = 2), alfabeto binario (v = 2), es decir con valores 0 y 1 en cada celda, cuatro factores (k = 4) y 5 filas o casos de prueba (N = 5). La característica especial que tienen los CA es que cualquier conjunto t de columnas que se extraigan del arreglo, contiene todas las combinaciones posibles de v^t tuplas en al menos una de las filas [31], que para este caso significa que siempre existen las posibles tuplas [(0,0), (0,1), (1,0) y (1,1)] en cualquier combinación de a 2 columnas (fuerza 2).

$$CA(N = 5;\ t = 2, k = 4, v = 2) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Ilustración 5. Ejemplo de un CA binario de fuerza 2. Ilustración propia.

Los CAs pueden tener diferentes tamaños de filas y la misma fuerza. En caso de que alguno contenga el menor número de filas reportado en la literatura, se dice que es un CA óptimo y su notación es *CAN (N; t, k, v)*. La construcción de los CAs, en especial los óptimos es un problema complejo y los métodos para su construcción se dividen en: exactos, algebraicos, greddy y basados en meta-heurísticas [32]. En esta investigación no se van a construir CAs, sino que se usarán CAs disponibles en repositorios como base para la propuesta del algoritmo de selección de instancias.

Por ejemplo, si se tiene un sistema con 7 parámetros, cada uno binario (prendido/apagado), tal como la caja roja que se muestra en la **Ilustración 6** para definir los efectos de una fuente en Microsoft Office, y se desea realizar todas las combinaciones de los parámetros, se requieren 128 (2^7) combinaciones. Para simplificar las pruebas se puede utilizar un CA con fuerza 2, de esta forma solo se requieren 6 casos de prueba, como se ve en la **Ilustración 7**.



$2^7 = \mathbf{128}$ combinaciones

1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1
3	0	0	0	0	0	1	0
4	0	0	0	0	0	1	1
5	0	0	0	0	1	0	0
6	0	0	0	0	1	0	1
7	0	0	0	0	1	1	0
8	0	0	0	0	1	1	1
...	...						
126	1	1	1	1	1	0	1
127	1	1	1	1	1	1	0
128	1	1	1	1	1	1	1

Ilustración 6. Ejemplo Covering Array. Ilustración propia

CA(N=6; t=2; k=10; v=2)

Ilustración 7. Ejemplo Covering Array para evaluar los efectos de las fuentes en Microsoft Office. Ilustración propia

Respecto a los CAs y la selección de instancias, en esta investigación se asume que 0 representa la ausencia de una instancia y 1 la presencia, en este sentido, cada fila de un CA binario se puede utilizar como una referencia que sugiere un subconjunto de instancias candidatas a ser parte de un conjunto de entrada en un MAS, de manera que sí se construyen varios modelos dados por dichos subconjuntos y se evalúan con un criterio de calidad apropiado, se tendría un nuevo algoritmo de selección de instancias que utiliza como criterio de búsqueda un CA. El éxito de encontrar el mejor subconjunto dependerá en este caso del número de interacciones posibles que el arreglo haya tomado en cuenta, lo cual puede ser controlado con el parámetro fuerza *(t)*. La **Ilustración 8**, muestra los subconjuntos formados por un CA de fuerza 2 y 4 variables que corresponden a instancias del datasets.

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} => \boldsymbol{Subconjuntos} => \begin{bmatrix} \boldsymbol{Null} \\ \boldsymbol{i_2, i_3, i_4} \\ \boldsymbol{i_1, i_3, i_4} \\ \boldsymbol{i_1, i_2, i_4} \\ \boldsymbol{i_1, i_2, i_3} \end{bmatrix}$$

Ilustración 8. Extracción de un subconjunto de instancias utilizando CA. Ilustración propia.

Página intencionalmente dejada en blanco

3. ESTADO DEL ARTE EN SELECCIÓN DE INSTANCIAS

A continuación, se presentan los trabajos relacionados (estado del arte) más importantes en la selección de instancias teniendo en cuenta las dos estrategias principales, a saber, el muestreo y la selección de prototipos (vecinos cercanos, eliminación ordenada, algoritmos evolutivos y codificación de longitud).

3.1 Muestreo

Uno de los principales medios de selección de instancias es el muestreo, mediante el cual se selecciona una muestra para su análisis y prueba, y la aleatoriedad es un elemento clave en el proceso. La selección de instancias también cubre los métodos que requieren búsqueda. Se pueden encontrar ejemplos en la estimación de densidad (búsqueda de instancias representativas - puntos de datos para un grupo); búsqueda de límites (búsqueda de instancias críticas para formar límites para diferenciar puntos de datos de diferentes clases); Otras aspectos importantes relacionados con la selección de instancias incluyen la eliminación de ruido (valores atípicos), el suavizado de los datos, entre otros [33][1].

El objetivo de todas las técnicas de muestreo es obtener una muestra representativa (es decir, una muestra en la que las instancias seleccionadas representen con precisión la población). Por supuesto, obtener una muestra representativa de un conjunto de datos grandes es especialmente desafiante, dada su naturaleza siempre cambiante y cada vez mayor, así como su complejidad potencial. Sin embargo, los métodos ampliamente aceptados para obtener muestras son múltiples [34]. Las técnicas más comunes se discuten a continuación.

3.1.1 Muestreo Aleatorio

Hay una serie de consideraciones clave antes de obtener una muestra aleatoria adecuada para una tarea de minería de datos. Es esencial comprender las fortalezas y debilidades de cada método de muestreo. También es esencial comprender qué métodos de muestreo son más adecuados para el tipo de datos a procesar y el algoritmo de extracción de datos a emplear [35].

En el muestreo aleatorio, cualquier instancia tiene la misma probabilidad de ser seleccionada. Existen dos tipos de muestreo aleatorio: Muestreo con y sin Reemplazo. La principal diferencia es que en el muestreo con reemplazo, una misma instancia puede ser seleccionada múltiples veces [33][1].

A continuación, se presenta el algoritmo más representativo de selección de instancias por muestreo aleatorio.

3.1.1.1 Random Mutation Hill Climbing (RMHC)

Hill Climbing es una heurística sencilla que prueba si una solución vecina a la solución actual es mejor, si es así, cambia la solución actual por la vecina y repite el proceso. En el

proceso de selección de instancias opera de la siguiente manera: El subconjunto *S* se inicializa mediante *m* instancias seleccionadas aleatoriamente desde el dataset de entrenamiento *T*. Para cada iteración, el algoritmo reemplaza aleatoriamente una instancia en *S* por otra instancia seleccionada aleatoriamente en *T*-*S*. Si este reemplazo puede mejorar la precisión predictiva de las instancias en *T*, el cambio será aceptado, este proceso se repite para *p* veces, donde p es un parámetro proporcionado por el usuario. Este sencillo algoritmo estocástico puede reducir significativamente el almacenamiento sin disminuir la precisión de la clasificación del vecino más cercano e incluso mejorarla [36].

En relación con su complejidad, para codificar un solo prototipo se requiere $\log_2 n$, donde n es el número de registros en el conjunto de entrenamiento. Se requieren $i * \log_2 n$ bits para codificar i prototipos, además en cada iteración muta aleatoriamente un solo bit y evalúa la precisión predictiva del conjunto real de instancias [37].

3.1.2 Muestreo Estratificado

El Muestreo Estratificado (Stratified Sampling, SS) es muy útil, cuando el conjunto de datos tiene instancias homogéneas, en estos casos es más efectivo elegir tuplas de cada grupo [1]. Para ello el muestreo estratificado primero usa una estrategia de estratificación para separar las instancias en un conjunto de estratos Z = {z_1, ..., z_i, ..., z_M}, y luego extrae muestras de cada estrato de forma independiente mediante la aplicación de otras técnicas de muestreo, tales como muestreo aleatorio simple. La estrategia de estratificación implica la selección de variables estratificadas, que deben ser categóricas [38]. Existen cuatro variaciones de muestreo estratificado [38]:

- Muestreo proporcional: garantiza que la proporción de instancias en cada estrato sea la misma, tanto en la muestra como en la población
- Igual tamaño: extrae el mismo número de instancias para cada estrato.
- Asignación de Neyman: asigna unidades de muestreo a estratos proporcionales a la desviación estándar en cada estrato.
- Asignación óptima: Con una asignación óptima, tanto la proporción de instancias como la desviación estándar relativa de una variable específica dentro de los estratos son las mismas que en la población.

En la **Tabla 1**, se esboza el algoritmo de muestreo estratificado.

Entradas:	T	Set de entrenamiento $\{X_1, X_2, X_3, \dots, X_n\}$,
	Z	conjunto de estratos $\{z_1, z_2, \dots, z_M\}$,
	apliqueMuestreo	operación de muestreo
Salida: Conjunto de instancias de muestra S		
1.	**Inicio**	
2.	$S \leftarrow \emptyset$	
3.	**Para todo** z_i **en** Z **haga** // Es decir para todos los estratos o clases	
4.	$X' \leftarrow \sigma_{z_i=true}(T)$	
5.	$S' \leftarrow$ apliqueMuestreo(X')	
6.	$S \leftarrow S \cup S'$	
7.	**Fin Para**	
8.	**Retornar** S	
9.	**Fin**	

Tabla 1. Algoritmo de muestreo estratificado

3.1.3 Muestreo por Agrupamiento

En el muestreo por agrupamiento (clustering), cada unidad de muestreo es una colección o grupo de elementos [39]. Funciona de la siguiente manera: las instancias del dataset se dividen en unidades primarias, cada unidad primaria se compone de unidades secundarias. Para incluir una instancia en la muestra, se compara cada instancia de la muestra, con la instancia candidata y se define si se retiene o elimina [40], el éxito de estas técnicas depende de la premisa que el objeto representativo seleccionado mantenga la información estructural importante de los datos.

La principal diferencia con el muestreo estratificado, es que en el muestreo por agrupamiento es deseable que las instancias sean lo más heterogéneas posibles [1]. Al igual que con otras formas de muestreo, el primer paso consiste en especificar el conjunto de datos y el marco de la muestra, en este caso, el número de grupos. Normalmente, las instancias dentro de un grupo son "cercanas" de alguna manera (por ejemplo, geográficamente, cronológicamente, emotivamente) y por lo tanto tienden a tener características similares [34].

El muestreo por agrupamiento es preferible en dos condiciones:

- Cuando el fenómeno en estudio no se puede muestrear de otra manera, por ejemplo, ver la trazabilidad de los mensajes de un correo electrónico, dado que los mensajes aparecen relacionados entre sí.
- Cuando el fenómeno en cuestión solo se puede observar en grupos, por ejemplo, la interacción de blogs que, por definición, se compone de grupos de publicaciones.

3.1.4 Muestreo Sistemático

El muestreo sistemático, o también llamado "1 en K" busca que todas las instancias del conjunto tengan las mismas oportunidades de ser elegidas [1][41]. En general, el muestreo sistemático implica la selección aleatoria de un numero entre 1 y k, luego se selecciona una instancia aleatoria entre 1 y k del conjunto de datos, después se selecciona la instancia que esta k posiciones adelante y el proceso se repite hasta llegar al final del dataset [39].

Por ejemplo, si una colección de comentarios en un sitio web de juegos contiene un total de 997 publicaciones, se pueden obtener 100 muestras (aproximadamente el 10% del total de publicaciones) de la siguiente manera: Seleccione un número aleatorio entre 0 y 9 (cada 10 registros se selecciona 1 para así muestrear el 10%). Si se asume que, se selecciona el número 6, entonces la muestra se compone por los registros 6, 16, 26, 36, …, 986, 996 [34].

El muestreo sistemático tiene múltiples ventajas frente al muestreo aleatorio simple [42]:

- El tamaño del conjunto de datos puede ser desconocido.
- Es fácil de aplicar y no implica operaciones matemáticas complejas.
- Puede "producir estimaciones más precisas que una muestra aleatoria simple” [39].

3.1.5 Muestreo de múltiples capas

Los grandes conjuntos de datos se adaptan fácilmente a diseños de múltiples capas o etapas, en los que el muestreo se produce de manera repetida a lo largo del tiempo [34]. Por ejemplo, si, después de seleccionar una muestra de unidades primarias, se selecciona una muestra de unidades secundarias de cada una de las unidades primarias

seleccionadas, el diseño se denomina muestreo de dos etapas. Si, a su vez, se selecciona una muestra de unidades terciarias de cada unidad secundaria seleccionada, el diseño es un muestreo en tres etapas. También son posibles diseños de múltiples etapas de orden superior [40], El diseño de múltiples etapas es una alternativa útil para profundizar en un conjunto de datos complejos para encontrar incidencias exactas del fenómeno en estudio [34].

3.1.6 Muestreo doble o en dos fases

En la Fase I, una gran muestra inicial de instancias se recopila y utiliza para estimar de forma rápida y precisa el soporte de cada instancia individual en el conjunto de datos. En la Fase II, se obtiene una pequeña muestra final de la muestra inicial, de tal manera que el soporte de cada instancia en la muestra final es lo más cercano posible al soporte (estimado) de la instancia en todo el conjunto de datos. Se presentan dos enfoques diferentes para obtener la muestra final en la Fase II:

- Recortar: El procedimiento de recorte comienza desde la muestra inicial grande y continúa eliminando instancias “atípicas” hasta que se cumple un criterio de detención específico.
- Crecer: El procedimiento de crecimiento selecciona instancias "representativas" de la muestra inicial y las agrega a un conjunto de datos inicialmente vacío.

El procedimiento de la Fase II ayuda a garantizar que el soporte de cada instancia de la muestra sea similar al del conjunto datos [1], [43], [44].

3.1.7 Muestreo Progresivo

El muestreo progresivo comienza con una muestra pequeña y utiliza muestras progresivamente más grandes hasta que la precisión del modelo ya no mejora [45]. Un componente central del muestreo progresivo es un programa de muestreo $S = \{n_0, n_1, n_2, \dots, n_k\}$ donde cada n_i es un número entero que especifica el tamaño de una muestra que se proporcionará a un algoritmo de inducción.

A diferencia de los anteriores métodos de muestreo, en este, no es necesario determinar el tamaño del subconjunto. Hay tres preguntas fundamentales con respecto al muestreo progresivo:

- ¿Qué es un programa de muestreo eficiente?
- ¿Cómo se puede detectar la convergencia (es decir, que la calidad del modelo ya no aumenta) de manera efectiva y eficiente?
- A medida que el muestreo progresa, ¿se puede adaptar el programa para ser más eficiente?

En la **Ilustración 9** se muestra la relación entre el tamaño de la muestra y la precisión del modelo. El eje horizontal representa n, el número de instancias en un conjunto de entrenamiento dado, que puede variar entre cero y N. El eje vertical representa la precisión del modelo producido por un algoritmo de inducción (o MAS) cuando se le asigna un conjunto de entrenamiento de tamaño n. Las curvas de aprendizaje suelen tener una parte con una pendiente pronunciada al principio de la curva, una parte media con una pendiente más suave y una meseta al final de la curva. La parte media puede ser extremadamente grande en algunas curvas o casi nula en otras. La meseta se produce cuando al agregar instancias adicionales no se mejora la precisión. La meseta, e incluso toda la parte media, puede faltar en las curvas cuando N no es lo suficientemente grande [45].

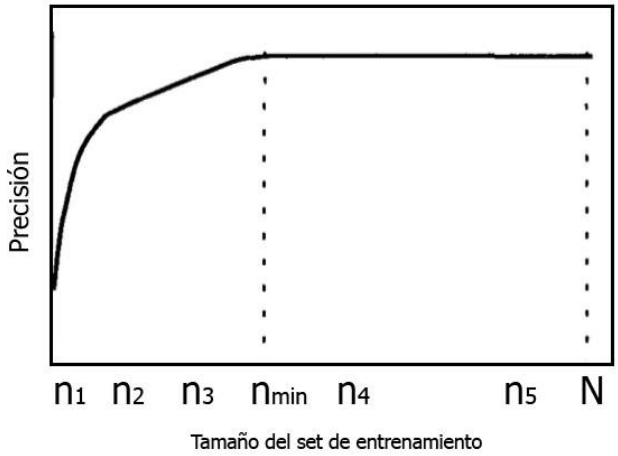


Ilustración 9. Curvas de aprendizaje en muestreo progresivo. Ilustración propia

3.2 Selección de Prototipos

Los métodos de selección de prototipos reducen el conjunto de entrenamiento a pocas muestras altamente representativas. Por lo general, estos prototipos no se extraen de los datos de entrenamiento, sino que se crean en centros de agrupaciones que contienen instancias asociadas con una sola clase. En la mayoría de los casos de métodos basados en prototipos, el número de prototipos resultantes debe ser determinado explícitamente por el usuario [46]. Esta técnica de selección de instancias busca crear subconjuntos de instancias que tengan mejores porcentajes de clasificación empleando la regla del vecino más cercano (1-nn) [4].

El algoritmo kNN, es uno de los más importantes en el aprendizaje automático [3]. Este algoritmo calcula la distancia euclidiana (de Jaccard, manhattan u otra, ponderada o no) entre una instancia a clasificar y cada instancia de entrenamiento vecina. La nueva instancia a clasificar se asigna a la clase de la vecina más cercana [3]. El algoritmo kNN no es practico para trabajar con dataset muy grandes, ya que estos demandan muchos recursos computacionales en almacenamiento y procesamiento, porque necesita almacenar todas las instancias en la memoria o hacer un eficiente uso del intercambio a disco [4][3].

Los algoritmos de selección de prototipos se pueden clasificar como se muestra en la **Ilustración 10** [3], a continuación se describen cada una de ellas.

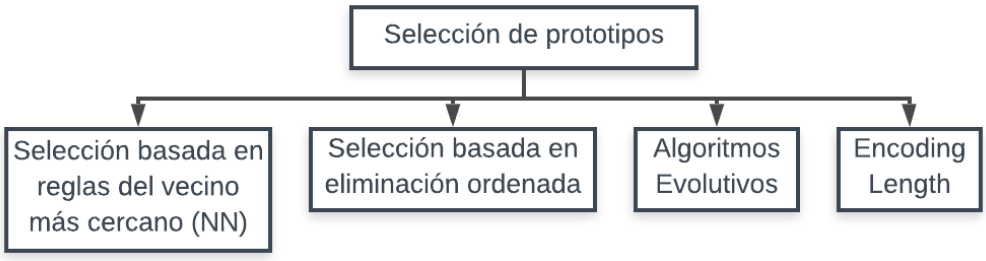


Ilustración 10. Algoritmos de selección de prototipos. Ilustración propia.

3.2.1 Selección basada en reglas del vecino más cercano (NN)

Un algoritmo NN (nearest neighbors) busca conseguir un subconjunto del dataset de entrenamiento, el cual sirva para obtener la tasa de clasificación máxima con el clasificador kNN [3]. La **Ilustración 11** muestra cómo trabajan este algoritmo.

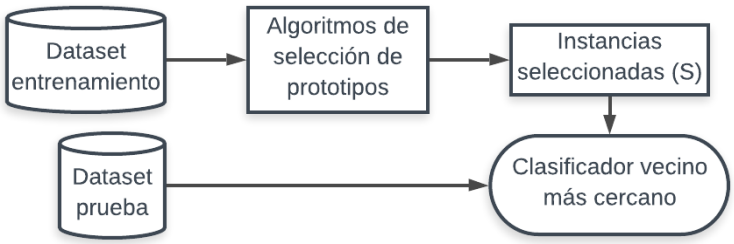


Ilustración 11. Estrategia vecino más cercano. Ilustración propia.

Las desventajas más importantes de los algoritmos NN son [47]–[49]:

- Son computacionalmente costosos, ya que guardan todas las instancias de entrenamiento.
- No son buenos con valores ruidosos.
- No son buenos con atributos irrelevantes.
- El rendimiento depende de la elección de la función de similitud para calcular la distancia entre dos instancias.
- No hay una manera simple o natural de trabajar con atributos de valor nominal o datos faltantes.
- No dicen mucho sobre la estructura de los datos.

A continuación, se presentan los algoritmos más relevantes basados en NN [3].

3.2.1.1 Condensed Nearest Neighbour (CNN)

Busca encontrar un subconjunto, que logre clasificar correctamente todos los datos restantes en el conjunto de muestra [50]. Para ello, obtiene un subconjunto S del dataset de entrenamiento T, buscando que cada instancia presente en S sea lo más parecida posible a una instancia de su clase, que a una instancia de otra clase. Este algoritmo elige aleatoriamente una instancia de cada clase y las agrega a S (que al inicio está vacío). Luego, cada instancia en T se clasifica con kNN usando solamente las instancias de S y si es incorrectamente clasificada se adiciona a S. El proceso se repite hasta que no existan instancias en el subconjunto T que se clasifiquen incorrectamente [50].

CNN se ha convertido en un punto de referencia con el que se comparan la mayoría de los otros algoritmos [51]. Baram [51] realizó estudios sobre las propiedades y la complejidad de este algoritmo, y mostró que, en el caso general, la complejidad es de $\theta(n^3)$, donde n es el tamaño del conjunto original. En la **Tabla 2**, se muestra formalmente el algoritmo.

Este algoritmo es sensible al ruido, ya que las instancias ruidosas logran ser parte del conjunto S, provocando las siguientes dificultades [50]:

- El subconjunto resultante no se reduce eficientemente, porque las instancias ruidosas son innecesarias, pero hacen parte de la solución.
- Las instancias ruidosas no aportan información relevante al clasificador, lo que conlleva a una reducción en la calidad del resultado.

CNN garantiza que todas las instancias en el conjunto de entrenamiento tengan la misma clasificación que en el dataset original, y que el nuevo conjunto no será más grande que el original. En la práctica, el conjunto de clasificación obtenido, denominado CNN, es mucho más pequeño que el conjunto original. Aunque simple y razonablemente eficiente, este algoritmo está lejos de ser óptimo, y tiene varios inconvenientes adicionales [51].

Entradas:	T	Set de entrenamiento $\{X_1, X_2, X_3, \dots, X_n\}$
Salida:	CNN	Conjunto de instancias de muestra (Condensed Nearest Neighbour)

```
1.  Inicio
2.      CNN = {una instancia aleatoria de cada clase}
3.      Repita
4.          Adiciones ← FALSE
5.          Para i ← 2 hasta |T|      /* hasta el número de instancias en E */
6.              Clasifique X_i con kNN usando las instancias de CNN
7.              Si X_i es clasificado incorrectamente entonces
8.                  CNN ← CNN U {X_i}
9.                  Adiciones ← TRUE
10.             Fin Si
11.         Fin Para
12.     Hasta que Adiciones = FLASE
13. Fin
```

Tabla 2. Algoritmo CNN

3.2.1.2 Selective Nearest Neighbor (SNN)

Ritter et al [47] extendieron el método CNN en su Regla de Vecino Cercano Selectivo (SNN), de manera que cada miembro de T debe estar más cerca de un miembro de S de la misma clase que de cualquier miembro de T (en lugar de S) de una clase diferente. Además, el método garantiza un subconjunto mínimo que satisface estas condiciones. El algoritmo para SNN es más complejo que la mayoría de los otros algoritmos de reducción, y el tiempo de aprendizaje es significativamente mayor, debido a la manipulación de una matriz de $n * n$ (donde n es el número de instancias en T) y el uso ocasional de algunos métodos de recursión.

Este algoritmo toma aproximadamente $\theta(mn^2 + n^3)$ tiempo, comparado con el $\theta(mn^2)$ o menos tiempo requerido por la mayoría de los otros algoritmos. También requiere almacenamiento de $\theta(n^2)$ durante la fase de aprendizaje de la matriz, aunque esta matriz se descarta una vez que se completa el aprendizaje. El algoritmo es sensible al ruido, aunque tiende a sacrificar el almacenamiento más que la precisión cuando hay ruido presente [47]. La **Tabla 3** ilustra el algoritmo SNN.

3.2.1.3 Edited Nearest Neighbour (ENN)

Es una modificación del algoritmo 1-NN que edita las instancias ruidosas, así como los casos de borde, dejando límites de decisión más suaves. También retiene todos los puntos internos, lo que evita que reduzca los requisitos de almacenamiento tanto como la mayoría de los otros algoritmos de reducción [3][49].

Este algoritmo toma cada instancia en el conjunto de entrenamiento y la clasifica utilizando la regla k-NN, luego se marca para su eliminación si su clase predicha no está de acuerdo con la clase verdadera. La edición se logra eliminando todas las instancias mal clasificadas

a la vez. Después, cualquier muestra de entrada se clasifica utilizando la regla 1-NN con las instancias restantes [52]. La complejidad de este algoritmo es de $\theta(n^2)$.

Entradas:	A Matriz binaria de Ritter SS Subconjunto selectivo de prototipos, inicializado en ∅ CNN Conjunto de instancias de muestra (Condensed Nearest Neighbour)
Salida: Conjunto de instancias de muestra	
1.	**Inicio**
2.	**Para** todas las **i** correspondientes a las columnas que quedan en **A**
3.	**Si** la columna **i** de **A** tiene un solo 1, **entonces**:
4.	Almacena el índice de la fila j donde se produce ese 1
5.	SS ← SS + $\{\boldsymbol{p_j}\}$
6.	Eliminar todas las columnas de **A** donde la fila **j** tiene valor 1
7.	**Fin Si**
8.	**Fin Para**
9.	**Para** todas las **j** correspondientes a las filas restantes en **A**
10.	**Para** todos los **k ≠ j** correspondientes a las filas restantes en **A**
11.	**Si** para todo i $\boldsymbol{A_{ji} \leq A_{ki}}$, **entonces** borre la fila **j**
12.	**Fin Para**
13.	**Fin Para**
14.	**Para** todas las i correspondientes a las columnas que quedan en **A**
15.	**Para k ≠ i** correspondiente a las columnas que quedan en **A**
16.	**Si** para todos j $\boldsymbol{A_{ji} \leq A_{jk}}$, **entonces** borre la columna **i**
17.	**Fin Para**
18.	**Si** A está vacío, **entonces** termina el algoritmo
19.	**Si** alguna de las eliminaciones en los pasos 1 a 8 se remonta al paso 1, use un algoritmo de ramificación y límite para seleccionar el número mínimo de prototipos, correspondiente a las filas, que garantizará que cada columna tenga al menos un 1.
20.	**Fin Para**
21.	**Fin**

Tabla 3. Algoritmo Selective Nearest Neighbor

3.2.1.4 Repeated Edited Nearest Neighbour (RENN)

Este algoritmo consiste en aplicar repetidamente ENN ampliando la brecha entre clases y suavizando el límite de decisión [3][49]. La complejidad de este algoritmo es $\theta(in^2)$, donde i, es el número de iteraciones.

3.2.1.5 Reduced Nearest Neighbour (RNN)

Una de las razones principales por las que CNN no obtiene un número mínimo de prototipos, es porque en los primeros pasos del algoritmo se incluyen prototipos que luego serán redundados por nuevas adiciones. Una solución a este problema fue propuesta por Gates [53], llamada los Vecinos Cercanos Reducidos. Este método para obtener un conjunto de clasificación utiliza como punto de partida el subconjunto obtenido de CNN y posteriormente lo reduce [53]. En la **Tabla 4** se muestra el algoritmo RNN. Este enfoque es eficiente, sólo si el conjunto consistente de CNN contiene el conjunto mínimo consistente del dataset de aprendizaje, lo que no siempre es el caso [53]. La complejidad de este algoritmo es de $\theta(n^3)$.

Entradas:	T Set de entrenamiento, con patrones $\mathbf{X_1}, \mathbf{X_2}, \mathbf{X_3}, \ldots, \mathbf{X_n}$ CNN Conjunto de instancias de muestra (Condensed Nearest Neighbour) RNN Conjunto de vecinos más cercanos reducidos, con prototipos $\mathbf{rnn_1}, \mathbf{rnn_2}, \ldots, \mathbf{rnn_n}$ Candidato_RNN Un conjunto de prototipos
Salida: Conjunto de instancias de muestra	
1.	**Inicio**
2.	RNN ← CNN
3.	**Para** i =1 **hasta** |CNN|
4.	Candidato_RNN ← RNN – $\{rnn_i\}$
5.	Clasifique todo T con Candidato_RNN
6.	**Si** todos los patrones en T son correctamente clasificados
7.	RNN ← Candidato_RNN
8.	**Fin Si**
9.	**Fin Para**
10.	**Fin**

Tabla 4. Algoritmo Reduced Nearest Neighbour

3.2.1.6 Variable Similarity Metric (VSM)

Los algoritmos de interpolación del vecino más cercano tienen muchas propiedades útiles para las aplicaciones del aprendizaje de máquina, pero a menudo presentan una generalización deficiente [54]. VSM elimina una instancia si la mayoría de sus vecinos más cercanos (60%) la clasifican de forma correcta o incorrecta [54].

3.2.1.7 Multiedit

Es una modificación del algoritmo ENN que garantiza la independencia estadística en el prototipo seleccionado [54]. El algoritmo Multiedit rompe aleatoriamente el conjunto de entrenamiento inicial en diferentes subconjuntos. En cada subconjunto, cada instancia se clasifica utilizando la regla 1-NN con las instancias en el siguiente subconjunto. Las instancias mal clasificadas se descartan. Las instancias restantes constituyen un nuevo conjunto y el algoritmo se repite iterativamente hasta que no se eliminan más instancias [52].

3.2.1.8 Chang

La idea principal del algoritmo de Chang es generar nuevos prototipos, al fusionar dos prototipos existentes en uno solo, ubicado en su punto medio ponderado. El proceso de fusión se detiene cuando el error de clasificación comienza a aumentar [51].

Si bien se han obtenido buenos resultados de clasificación con los clasificadores del algoritmo de Chang, debe señalarse que, a medida que se generan nuevos prototipos, las distancias de los prototipos a los patrones de entrenamiento deben calcularse muchas veces, lo que impone una pesada carga computacional [51]. La **Tabla 5** presenta el algoritmo de Chang.

3.2.1.9 Shrink

Este algoritmo es similar a RNN, almacena los puntos de borde, pero a diferencia de RNN, este considera si la instancia eliminada se clasificaría correctamente, mientras que RNN

considera si la clasificación de otras instancias se vería afectada por la eliminación de la instancia. Shrink es sensible al ruido [55].

Este algoritmo comienza con S = T y luego elimina las instancias que aún serían clasificadas correctamente por el subconjunto restante. Esto es similar a la regla del Vecino más cercano reducido (RNN), excepto que solo considera si la instancia eliminada se clasificaría correctamente, mientras que RNN considera si la clasificación de otras instancias se vería afectada por la eliminación de la instancia. Al igual que RNN y muchos de los otros algoritmos, retiene puntos de borde, pero a diferencia de RNN, este algoritmo es sensible al ruido [33].

Entradas:	T Set de entrenamiento, con patrones $X_1, X_2, X_3, \dots, X_n$ Chang Conjunto de prototipos Chang $Chang_1, Chang_2, \dots, Chang_n$ $Changw_i$ Peso asociado al prototipo $Chang_i$ en Chang Error Variable para almacenar la tasa de error
Salida: Conjunto de instancias de muestra	
1.	**Inicio**
2.	Chang = T
3.	changw = 1 para todos los prototipos
4.	**Repita**
5.	Encuentra a los dos vecinos más cercanos $Chang_i, Chang_j$ en Chang
6.	Elimine $Chang_i, Chang_j$ en Chang
7.	Crear $Chang_k$ en el punto medio entre $Chang_i, Chang_j$, de modo que $Chang_k \leftarrow (Changw_i * Chang_i + Changw_j * Chang_j)/(Changw_i + Changw_i)$
8.	$Changw_k \leftarrow Changw_i + Changw_j$
9.	Clasifique T con Chang, y calcule la tasa de erorr
10.	**Si** el error es mayor que el error deseado, **entonces** elimine $Chang_k$ de Chang, vuelva a reinsertar $Chang_i$y $Chang_j$y finalice la fusión
11.	**Hasta que** el error exceda la tasa de error deseada
12.	**Fin**

Tabla 5. Algoritmo Chang

3.2.1.10 Instance Based Learning (IBL, IB2, IB3, TIBL, BIBL)

El aprendizaje basado en instancias (IBL) es una extensión de los algoritmos de clasificación del vecino más cercano o kNN. Los algoritmos IBL no mantienen un conjunto de abstracciones del modelo creado a partir de las instancias [56], en lugar de eso, analizan cómo se puede reducir significativamente el requisito de almacenamiento, con solo una pérdida menor en la velocidad de aprendizaje y la precisión. Además, busca que se logre trabajar con instancias ruidosas, ya que muchos conjuntos de datos de la vida real tienen instancias de entrenamiento muy diferentes y kNN no funcionan bien con el ruido [55][56].

El algoritmo IB1 es idéntico al algoritmo del vecino más cercano, sin embargo, normaliza los rangos del atributo, lo que también puede hacer el algoritmo NN, sin embargo, este algoritmo procesa instancias de forma incremental, a diferencia del algoritmo NN tradicional [57].

El algoritmo IB2 mejora a IB1 para que se reduzca el requisito de almacenamiento. Sin embargo, la precisión de la clasificación de IB2 disminuye más rápidamente que la de IB1 a medida que aumenta el nivel de ruido, esto se debe a que es más probable que las

instancias ruidosas se clasifiquen erróneamente y que IB2 guarde solo las instancias mal clasificadas, que a su vez utiliza para generar decisiones de clasificación. De modo que, cuando el conjunto de datos está relativamente libre de ruido, las precisiones de clasificación de IB2 son comparables a las de IB1. Si el conjunto de datos es ruidoso, la precisión de IB2 es un poco menor [24], [57].

IB3 mejora al algoritmo IB2 para manejar instancias ruidosas. Este algoritmo mantiene un registro de clasificación con cada instancia guardada. El registro de clasificación guarda el rendimiento de clasificación de una instancia en las instancias de entrenamiento presentadas posteriormente y sugiere cómo se desempeñará en el futuro. Sin embargo, a medida que aumenta el número de atributos, el rendimiento de IB3 disminuye drásticamente y su requerimiento de espacio también aumenta drásticamente. IB3 no funciona bien con conjuntos de datos de alta dimensión con muchos atributos irrelevantes [24], [57]. La complejidad de este algoritmo es de $\theta(n^2 \log_2 n)$.

En resumen, los algoritmos IBL son algoritmos de aprendizaje robustos, que pueden tolerar el ruido y los atributos irrelevantes y pueden representar conceptos probabilísticos y superpuestos.

3.2.1.11 Model Class Selection (MCS)

El algoritmo MCS (selección de clase de modelo) busca evitar el ruido [58]. Este algoritmo busca reducir el tamaño del conjunto de entrenamiento al hacer un seguimiento de cuántas veces cada instancia fue uno de los k vecinos más cercanos de otra instancia (a medida que las instancias se van agregando a la descripción del concepto), y si su clase coincide con la de la instancia que se está clasificando. Si el número de veces que estuvo mal es mayor que el número de veces que fue correcto, entonces se desecha. Esto tiende a evitar el ruido utilizando un enfoque más simple que IB3.

3.2.1.12 Iterative Case Filtering (ICF)

Busca seleccionar las instancias que clasifican más prototipos correctamente. Este algoritmo utiliza la cobertura y otros conceptos accesibles para llevar a cabo la selección [57]. ICF elimina las instancias cuyo alcance sea mayor a su cobertura, es decir, una instancia I se elimina cuando otras instancias brindan la misma información que I. ICF filtra la muestra empleando ENN. ICF define el conjunto local L(x) que contiene todos los casos dentro de la hiperesfera más grande centrada en x, de modo que la hiperesfera solo contiene casos de la misma clase que la instancia x [29]. Este algoritmo define dos propiedades muy importantes: alcance y cobertura.

En la primera fase, ICF utiliza el algoritmo ENN para eliminar el ruido del conjunto de entrenamiento. En la segunda fase, el algoritmo ICF elimina cada instancia x para la que la Cobertura(x) sea más grande que el Alcance(x). Este procedimiento se repite para cada instancia en T. Luego, el ICF vuelve a calcular las propiedades de cobertura y alcance y reinicia la segunda fase (siempre que se observe algún progreso). La complejidad asintótica de este algoritmo es $\theta(n^2)$ [29].

3.2.1.13 Gabriel Graph (GG)

Se dice que dos puntos A y B son vecinos de Gabriel si su esfera diametral (es decir, la esfera tal que AB es su diámetro) no contiene ningún otro punto [59]. La **Ilustración 12**

muestra que los puntos A y B son vecinos de Gabriel, mientras que B y C no lo son. Una gráfica en la que todos los pares de vecinos de Gabriel están conectados con un borde se denomina gráfica de Gabriel. Para calcular el gráfico de Gabriel se debe seguir los siguientes pasos:

- Visitar cada nodo y calcular si todos sus vecinos de Gabriel son de la misma clase que el nodo actual.
- Eliminar todos los nodos marcados y detenerse con los restantes como el conjunto de entrenamiento editado.

Esta heurística reduce la complejidad media a $\theta(mn^3)$. [60].

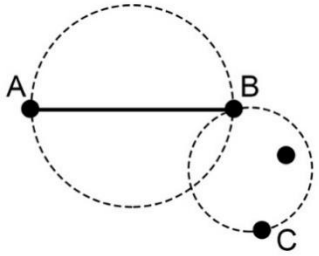


Ilustración 12. Vecinos de Gabriel. Ilustración propia.

3.2.1.14 Prototype Selection using Relative Certainty Gain (PSRCG)

PSRCG utiliza la teoría de la información para construir un criterio estadístico a partir de la topología del vecino más cercano. Este marco estadístico se utiliza en un algoritmo de selección de prototipos hacia atrás, que consiste en identificar y eliminar instancias no informativas y luego reducir la incertidumbre global del conjunto de aprendizaje. Este enfoque proporciona una buena solución para mantener un pequeño número de prototipos, sin comprometer la precisión de la clasificación. El algoritmo PSRCG parece ser robusto en presencia de ruido. Las interpretaciones en varios puntos de referencia tienden a mostrar la relevancia y la eficacia del método en comparación con los algoritmos clásicos de selección de prototipos.

3.2.2 Selección basada en eliminación ordenada

Esta técnica busca la reducción de instancias tolerando el ruido, alta precisión de generalización, insensibilidad al orden de presentación de las instancias y significativa reducción de almacenamiento, que a su vez mejora la velocidad de generalización [61].

A continuación se listan los algoritmos DROP, los cuales se basan en la eliminación ordenada de instancias [62]. Esta colección de heurísticas se utiliza para decidir qué instancias se deben mantener y qué instancias se deben eliminar de un conjunto de entrenamiento. A diferencia de la mayoría de los métodos anteriores, estos algoritmos toman una nota cuidadosa del orden en que se eliminan las instancias [61]. Los tres primeros métodos, DROP1 – DROP3 fueron creados por Wilson et at [61] con los nombres RT1-RT3 respectivamente. La complejidad asintótica en tiempo de estos algoritmos es $\theta(n^3)$ [62].

3.2.2.1 Decremental Reduction Optimization Procedure 1 (Drop1)

Este algoritmo verifica si la eliminación de una instancia degrada la precisión de la generalización de la validación cruzada, que es una estimación de la verdadera capacidad de generalización del clasificador resultante. Se elimina una instancia si las otras instancias similares se pueden clasificar correctamente sin dicha instancia, esto es, la ausencia de la instancia no afecta en la clasificación [62][63].

El algoritmo DROP1 y DROP2 no realizan un filtrado inicial del ruido. Esto conlleva a que las instancias ruidosas tengan un profundo impacto en cómo se ordenan las instancias al comienzo de estos algoritmos. Una instancia ruidosa significa que sus vecinos se consideran parte del límite de la clase, y se pueden mantener en el conjunto seleccionado que se ha filtrado incluso después de que se haya eliminado la instancia ruidosa [64]. En la **Tabla 6** se muestra se muestra el pseudocódigo de DROP1, que es la base de las otras variantes de DROP.

Entradas:	*T* Set de entrenamiento, con patrones $\{(X_1, Y_1), \ldots, (X_n, Y_n\}$
Salida: Conjunto de instancias *S* ⊆*T*	
1.	**Inicio**
2.	S ← T
3.	**Para** cada instancia X ∈ S **haga**
4.	Encuentre X.$N_{1\ldots k+1}$, los k + 1 vecinos más cercanos de X en S
5.	Agregue X a cada una de las listas de asociados de sus vecinos
6.	Fin Para
7.	**Para** cada instancia X ∈ S **haga**
8.	con = # de asociados de X clasificados correctamente con X como vecino
9.	sin = # de asociados de X clasificados correctamente sin X como vecino
10.	Fin Para
11.	**Si** sin ≥ con **entonces** eliminar X de S
12.	**Para** cada asociado a de X **haga**
13.	eliminar X de la lista de vecinas más cercanos de a
14.	buscar un nuevo vecino más cercano para a
15.	Agregue a, a la lista de asociados de su nuevo vecino.
16.	Fin Para
17.	**Para** cada vecino a de X **haga**
18.	eliminar X de la lista de asociados de a
19.	Fin Para
20.	**Fin**

Tabla 6. Decremental Reduction Optimization Procedure 1 (DROP1)

La lista de vecinos y asociados más cercanos se construyen de la misma manera en todas las variantes de DROP. Luego, se procesan las instancias y se calculan los valores de dos variables, ***con*** y ***sin***. Para cada instancia, la variable ***con*** cuenta cuántos de sus asociados se clasifican correctamente cuando la instancia se mantiene en el conjunto de datos, mientras que la variable ***sin*** registra el recuento de asociados clasificados correctamente cuando la instancia se elimina del conjunto de datos. Si el valor de ***sin*** es mayor o igual que el valor de ***con,*** la instancia no está ayudando en la clasificación de sus asociados y, por lo tanto, puede eliminarse del conjunto de datos. Cuando se elimina una instancia, es necesario actualizar la lista de vecinos más cercanos de todos sus asociados (se necesita un nuevo vecino más cercano). Como se muestra en la línea 3, las listas de vecinos más cercanos tienen en realidad k+1 vecinos, lo que hace que el cálculo de la variable sea más

rápido, ya que es posible posponer la búsqueda de un nuevo vecino más cercano hasta que la eliminación de la instancia tenga lugar efectivamente. DROP2 solo difiere de DROP1 en la clasificación inicial ya que elimina las líneas 14 y 15. Esta modificación significa que DROP2 considera el conjunto de datos inicial completo, en lugar de solo el subconjunto seleccionado [65].

3.2.2.2 Decremental Reduction Optimization Procedure 2 (Drop2)

Este algoritmo busca cambiar el orden de eliminación de las instancias. Verificando que causa tiene, si se elimina la instancia en el dataset de entrenamiento E. Se caracteriza porque busca eliminar primero las instancias más alejadas del límite de decisión, lo que a su vez aumenta la posibilidad de retener puntos de borde [62][63].

3.2.2.3 Decremental Reduction Optimization Procedure 3 y 4 (Drop3 y Drop4)

Drop3 y Drop4 utilizan un filtrado de ruido antes de ordenar las instancias en el dataset de entrenamiento [62]. Estos Algoritmos, aplican un filtro de ruido similar a la edición de ENN antes de iniciar el proceso de selección de instancia. El estado de filtrado elimina todas las instancias que no están clasificadas correctamente por sus k vecinos más cercanos [65], la diferencia entre estos dos algoritmos, es que Drop4 antes de realizar el filtrado, evalúa el impacto de la eliminación de la instancia, y con ello determina si se elimina o no [63].

3.2.2.4 Decremental Reduction Optimization Procedure 5 (Drop5)

Este algoritmo es una modificación de Drop2, primero elimina las instancias más parecidas a los enemigos más cercanos, es decir, instancias parecidas pero con distinta clase [63].

3.2.2.5 Decremental encoding length (DEL)

El algoritmo de longitud de codificación decremental (DEL) es el mismo que DROP3, excepto que utiliza la heurística de longitud de codificación, para decidir en cada caso si se puede eliminar una instancia. DEL comienza con S = T, y comienza con un filtrado de ruido en el que se elimina cada instancia, si:

- Está mal clasificada por sus k vecinos más cercanos
- La eliminación de la instancia no aumenta el costo de longitud de codificación.

Las instancias restantes se ordenan por la distancia a su enemigo más cercano, y siempre que se realice alguna mejora, las instancias restantes se eliminan (comenzando con la instancia más alejada de su enemigo más cercano) si eso no aumenta el costo de la longitud de codificación. La complejidad asintótica en tiempo de este algoritmo es $\theta(n^2)$ [61].

3.2.2.6 Patterns by Ordered Projections (POP)

Los algoritmos de patrones por proyecciones ordenadas (POP), presentan un nuevo enfoque para encontrar patrones representativos para la edición de conjuntos de datos. Se destaca por algunas características interesantes [48]:

- Importante reducción del número de instancias del conjunto de datos.
- Los algoritmos de Gabriel y SNN [47], [60], necesitan calcular las distancias entre los ejemplos, lo que consume bastante tiempo. Si se consideran n instancias con m atributos. El algoritmo SNN consume $\theta(mn^2 + n^3)$; El algoritmo de Gabriel $\theta(mn^3)$.

Mientras que el algoritmo POP tiene un menor costo computacional $\theta(mn \log n)$ con respecto a otros algoritmos típicos debido a la ausencia de cálculos de distancia [48].
- Conservación de los límites de decisión, especialmente desde el punto de vista de la aplicación de clasificadores de eje paralelo.

POP se ha comparado con IB2, ENN y SHRINK en relación con el porcentaje de reducción y el costo computacional. POP tiene un costo más bajo que los otros métodos. IB2 y SHRINK alcanzan un gran porcentaje de reducción, aunque SHRINK tiene una tasa de error muy alta, como en NN como C4.5. Al aplicar C4.5 a la base de datos reducida generada por IB2, el árbol de decisión produce una tasa de error mayor que la de POP. Además, se ha analizado la precisión de kNN y C4.5 después de aplicar las técnicas de reducción, dando resultados prometedores [48].

3.2.2.7 C-Pruner

El algoritmo C-Pruner, identifica los candidatos de poda y calcula el orden en que se eliminarán estos candidatos. El orden de las instancias es muy importante, ya que la eliminación de un candidato puede descalificar a otros candidatos. El método C-Pruner elimina instancias superfluas de una manera más cuidadosa que la familia DROP y la familia IB para lograr una mayor precisión de clasificación. Su objetivo es eliminar las instancias internas y conservar las instancias de borde [10].
Una instancia ruidosa es aquella que es clasificada incorrectamente por sus vecinos. La razón es porque: una instancia ruidosa está muy lejos de las instancias de su propia clase, por lo tanto, la clase que se deduce tiende a ser diferente a la clase a la que verdaderamente pertenece.

El orden de eliminación de las instancias es muy importante en el proceso de eliminación, ya que la eliminación de una instancia afectará la decisión sobre si otras instancias pueden descartarse. Las instancias internas deben eliminarse primero, para que las instancias de borde se puedan mantener tanto como sea posible. De lo contrario, puede causar el efecto dominó de la eliminación de las instancias de borde, por lo que se eliminan muchas instancias de borde, mientras que las instancias más reservadas son instancias internas. Una instancia interna está más cerca de los centros de su clase y más lejos de sus instancias enemigas más cercanas.

Zhao et al [10], verificaron los beneficios de C-Pruner, llevando a cabo experimentos en más de 15 dataset, comparando los resultados con los algoritmos IB3, DROP3 y KNN. Dicho estudio concluye que, el algoritmo KNN básico tiene la mayor precisión de clasificación promedio, sin embargo, almacena todas las instancias de entrenamiento. Mientras que C-Pruner alcanza una precisión de clasificación promedio considerablemente alta, que es aproximadamente un 2% inferior al método kNN. Además, C-Pruner supera a DROP3 en la precisión de clasificación promedio en casi un 1,4% y gana a IB3 en más de un 2%. De hecho, de los 15 dataset probados, C-Pruner obtiene una mayor precisión que DROP3 en 12 conjuntos de datos. Como se muestra en la **Tabla 7**, el requisito de almacenamiento de C-Pruner es un poco más alto que IB3 y DROP3.

3.2.3 Selección basada en Algoritmos Evolutivos

Como la reducción de datos se puede ver como un problema de búsqueda, se puede resolver utilizando algoritmos evolutivos [3]. Los algoritmos evolutivos, también conocidos como algoritmos genéticos en una terminología anterior, son algoritmos probabilísticos para

la optimización, que imitan a los operadores de la selección natural y genética, para hacer evolucionar una población de individuos codificados en cromosomas creando nuevas generaciones de descendientes a través de un proceso iterativo hasta que se cumplan algunos criterios o condiciones de convergencia [66]. La idea básica es mantener una población de cromosomas (individuos), que representan soluciones plausibles al problema que evolucionan con el tiempo a través de un proceso de competencia y variación controlada [67]. Los algoritmos evolutivos se han utilizado para resolver el problema de selección de instancias con resultados prometedores [68][69].

Datasets	**KNN**		**IB3**		**DROP3**		**C-Pruner**	
	Acc. (%)	**Stor. (%)**	**Acc. (%)**	**Stor. (%)**	**Acc. (%)**	**Stor. (%)**	**Acc. (%)**	**Stor. (%)**
Anneal	93.11	100,00	91.23	10.07	93,99	8,63	94,12	11,79
Australian	84.78	100,00	85.65	4.98	84,20	5,94	85,07	10,90
Breast Cancer (WI)	96.28	100,00	97.00	3.39	96,14	3,61	95,86	4,42
Bridge	44.00	100,00	44.00	10.05	44,00	4,28	44,00	8,06
Glass	73.83	100,00	62.19	33.18	64,55	23,68	68,74	31,62
Heart (Long Beach)	74.50	100,00	70.00	4.89	74,50	1,11	74,50	1,72
Heart (Swiss)	93.46	100,00	93.26	3.70	93,46	1,81	93,46	1,81
Image Segmentation	93.10	100,00	91.43	15.87	92,62	11,11	94,05	12,67
LED Creator	73.40	100,00	70.90	22.87	71,40	11,87	72,80	38,48
Liver (Bupa)	65.57	100,00	57.93	10,66	60,56	25,16	65,50	45,70
Sonar	87.55	100,00	71.24	12,98	78,00	26,82	79,90	35,79
Soybean (large)	88.59	100,00	85.65	30,15	84,65	25,12	86,60	27,87
Vehicle	71.76	100,00	66.09	28,41	65,38	23,09	67,62	34,13
Vowel	96.57	100,00	88.43	36,89	89,56	44,82	91,07	46,53
Zoo	94.44	100,00	93.33	30,99	88,89	20,86	88,89	24,20
Promedio	**82.06**	**100,00**	**77.89**	**17,27**	**78,79**	**15,86**	**80,15**	**22,37**

Tabla 7. Exactitud (Acc.) y almacenamiento (Stor.) de KNN, IB3, DROP 3 y C-Pruner

El principal inconveniente de los algoritmos evolutivos es la convergencia prematura, es decir, que después de algunas generaciones, la población actual alcanza un estado en el que la función objetivo no es óptima y ya no mejora más [70]. Por esta razón, su diseño debe tener en cuenta dos conceptos claves, la exploración del espacio de búsqueda y la explotación de las zonas más promisorias, evitando que el algoritmo quede atrapado en óptimos locales, es decir, que el algoritmo cuente con estrategias para escapar de esos óptimos locales. A continuación, se introducen algunos tres algoritmos que muestran las principales características usadas en selección de instancias:

3.2.3.1 Generational Genetic Algorithm (GGA)

Es un wrapper basado en un método hibrido para la selección de instancias [71], la idea básica en GGA [47][48] es mantener una población de cromosomas, que representan soluciones plausibles al problema particular que evoluciona a lo largo de sucesivas iteraciones (generaciones) a través de un proceso de competencia y variación controlada. Este algoritmo genético generacional (algoritmo genético estándar) inicia con una población generada aleatoriamente y luego crea nuevos descendientes a partir de los miembros de esa población utilizando operadores genéticos (selección, cruce y mutación) y coloca a

estos nuevos individuos en una nueva población que se convierte en la población de la siguiente generación (iteración) [74]. En este caso, cada individuo vive durante una única generación.

Cada cromosoma en la población tiene una aptitud asociada para determinar qué cromosomas se usarán para formar otros nuevos en el proceso de competencia. Esto se llama selección. Los nuevos se crean utilizando operadores genéticos como el cruce y la mutación. El modelo clásico de las algoritmos evolutivos es el GGA, que consta de tres operaciones [3]:

- Evaluación de la aptitud individual.
- Formación de un acervo genético (población intermedia), a través del mecanismo de selección.
- La recombinación a través de operadores de cruce y mutación.

El algoritmo genético modela la selección natural y el proceso de evolución y ha tenido éxito en áreas donde los métodos tradicionales no son suficientes. GA tiene varias ventajas sobre los métodos de optimización tradicionales: puede optimizar funciones continuas y discretas; no requiere ecuaciones diferenciales complicadas o una función objetivo suave; puede ser altamente paralelizado para mejorar el rendimiento de cómputo; busca más espacio de solución y, por lo tanto, aumenta la probabilidad de encontrar el óptimo global [75].

3.2.3.2 Steady-State Genetic Algorithm (SSGA)

Es un wrapper basado en un método hibrido para la selección de instancias [71]. El algoritmo genético incremental o de estado estacionario, es un genético que en cada generación crea un único descendiente, que reemplaza a un miembro de la población inicial. este modelo tiene una correspondencia muy cercana con el modelo generacional, ya que puede usar los mismos operadores de selección, cruce y mutación [66], pero se diferencia al modelo generacional, en que típicamente hay un solo miembro nuevo insertado en la nueva población en cualquier momento. Un operador de reemplazo o eliminación define qué miembro de la población será reemplazado por el nuevo individuo (descendencia) [74]. Muchos profesionales de minería de datos apoyan el uso de algoritmos genéticos de estado estable en los que se reemplaza a un solo individuo en cada paso [66].

3.2.3.3 Population based incremental learning (PBIL)

Es un wrapper basado en un método hibrido para la selección de instancias [71]. El algoritmo de aprendizaje incremental basado en población (PBIL) fue introducido por primera vez por Shumeet Baluja en 1994 [76] y se consideró una mejora del algoritmo genético estándar. Este algoritmo es una combinación de AG y aprendizaje competitivo y tiene muchas aplicaciones [76][77][78]. A diferencia de GA, PBIL mantiene un vector de probabilidad con valores reales, en lugar de una población, por lo que produce un requisito de memoria mucho menor. Además, sin un operador genético complejo como el crossover, PBIL tiene un costo computacional mucho menor [79].

3.2.3.4 CHC

Cross generational elitist selection Heterogeneous recombination Cataclysmic mutation algorithm (CHC) es un algoritmo evolutivo no tradicional que combina una estrategia de selección conservadora (que siempre preserva a los mejores individuos encontrados hasta la generación actual) con una recombinación (cruce) altamente disruptiva (HUX) que

produce descendientes que son muy diferentes de sus dos padres [80][81]. Este algoritmo tiene codificación binaria que trata de evitar la convergencia prematura obteniendo un equilibrio adecuado entre la capacidad de explorar (diversidad de las soluciones) el espacio de búsqueda y la capacidad de explotar las propiedades locales de las mejores soluciones. En lugar de la operación de mutación de un GA, en CHC se incluye un proceso de reinicio que ofrece muchos de los beneficios de una población de gran tamaño sin el costo de una búsqueda más lenta.

El algoritmo CHC funciona con una población de individuos o un conjunto de soluciones. El tamaño de la población (denotado por N) es una constante, generalmente pequeña. Esta población se actualiza en cada paso de tiempo o generación mediante la aplicación de los siguientes operadores [80]:

- Operador de cruce: En cada generación, se produce una nueva descendencia al elegir pares aleatorios de individuos de la población parental y recombinarlos con HUX [81].
- Mecanismo de prevención de incesto: La recombinación de parejas solo se realiza si el par de individuos no son muy similares [81].
- Reemplazo y selección elitista: La siguiente población se crea de acuerdo con un criterio elitista, basado en la selección de los mejores individuos en la población de padres y su descendencia [81].

Rathee et al [82], han usado este algoritmo en investigaciones para reducción de instancias, utilizando conjuntos de datos disponibles en el repositorio de aprendizaje automático de la UCI (University of California at Irvine). Cano et al [3]. presentaron un estudio experimental de diferentes algoritmos evolutivos (GGA, SGA, CHC, PBIL) y de acuerdo con sus resultados, el enfoque de CHC logra el mejor rendimiento en precisión y retención. Además, CHC fue el método que requirió menos tiempo de ejecución.

3.2.3.5 Encoding Length

Un grupo de tres algoritmos se inspiró en el principio de codificación de longitud de Cameron-Jones [83], el cual determina qué tan bueno es el subconjunto S al describir T. El algoritmo básico comienza con una fase de crecimiento que toma cada instancia i en T y la agrega a S si eso da lugar a un costo menor que al no agregarla. Al igual que con IB3, la fase de crecimiento puede verse afectada por el orden de presentación de las instancias [61].

3.2.3.6 Encoding Length Heuristic (ELH)

El algoritmo ELH es una implementación del algoritmo de selección de instancias propuesto por Cameron-Jones [83]. El algoritmo se basa en la heurística Encoding Length, evaluando la influencia de reflexión de una instancia. El algoritmo comienza rechazando una instancia y luego evalúa la influencia del rechazo en la heurística ELH. Si el rechazo no disminuye el valor de la heurística, se elimina el vector y se repite todo el procedimiento. Este algoritmo tiene muy buenas propiedades, como baja compresión, generalmente alta precisión, y es insensible a los valores atípicos. La complejidad de este algoritmo es $\theta(n^2)$ [37].

3.2.3.7 ELGrow / Explore

Este algoritmo realiza la reducción de las instancias, donde se elimina cada instancia i en S si al hacerlo se reduce el costo del clasificador. Cameron-Jones llama a este método el método "Pre/All", ya que no es realmente incremental. Este método es llamado Encoding Length Grow (ELGrow).

El método Explorer [83], comienza con el crecimiento y la reducción de S utilizando el método ElGrow, y luego realiza 1000 mutaciones para tratar de mejorar el clasificador. Cada mutación intenta agregar una instancia a S, eliminando una de S o intercambiando una en S con una en T - S, y mantiene el cambio si no aumenta el costo del clasificador. La precisión de generalización del método Explore es bastante buena empíricamente, y su reducción de almacenamiento es mucho mejor que la mayoría de los otros algoritmos.

Según [61], este algoritmo retiene en promedio 2.01% del conjunto de datos original. En experimentos de [84], retuvo en promedio el 2.26% de los conjuntos de datos considerados. Es, por lo tanto, uno de los algoritmos de reducción de instancias más efectivos. La complejidad de este algoritmo es $\theta(n^2)$ [61].

3.3 Resumen clasificación de algoritmos

Los algoritmos de selección de instancias funcionan de diferentes maneras, en la **Tabla 8** se resumen los principales algoritmos citados en este trabajo.

Estrategia	**Clasificación**	**Algoritmo**	**Complejidad**	
Muestreo	Muestreo Aleatorio	Random Mutation Hill Climbing (RMHC)	$i * \log_2 n$	n, numero de registros en el conjunto de entrenamiento, i la cantidad de prototipos.
Selección de prototipos	Selección basada en reglas del vecino más cercano (NN)	Condensed Nearest Neighbour (CNN)	$\theta(n^3)$	
		Selective Nearest Neighbour (SNN)	$\theta(mn^2 + n^3)$	
		Edited Nearest Neighbour (ENN)	$\theta(n^2)$	
		Repeated Edited Nearest Neighbour (RENN)	$\theta(in^2)$	i, número de iteraciones
		Reduced Nearest Neighbour (RNN)	$\theta(n^3)$	
		IB3	$\theta(n^2 \log_2 n)$	
		Iterative Case Filtering (ICF)	$\theta(n^2)$	
		Gabriel Graph (GG)	$\theta(mn^3)$.	m, es el número de atributos
	Selección basada en eliminación ordenada	Decremental Reduction Optimization Procedure 1 (Drop1)	$\theta(n^3)$	
		Decremental Reduction Optimization Procedure 2 (Drop2)	$\theta(n^3)$	
		Decremental Reduction Optimization Procedure 3 y 4	$\theta(n^3)$	
		Decremental Reduction Optimization Procedure 5 (Drop5)	$\theta(n^3)$	

Estrategia	Clasificación	Algoritmo	Complejidad	
		Decremental encoding length (DEL)	$\theta(n^2)$	
		Patterns by Ordered Projections (POP)	$\theta(mn \log n)$	m, es el número de atributos
	Selección basada en Algoritmos Evolutivos	Population based incremental learning (PBIL)		
	Encoding Length	Encoding Length Heuristic (ELH)	$\theta(n^2)$	
		ELGrow / Explore	$\theta(n^2)$	

Tabla 8. Clasificación algoritmos de reducción de instancias

Para probar la confiabilidad de los algoritmos de selección de instancias, Grochowski et al [85] verificaron el rendimiento de cada uno de estos en varios conjuntos de datos. Casi todas las pruebas se basaron en conjuntos de datos del repositorio de aprendizaje automático de la UCI [86]. Los resultados de la validación cruzada se repitieron y promediaron más de 10 veces (aunque lo normal deben ser 30 veces para que se cumpla con el teorema del límite central para los valores promedios). La estandarización de todos los datos la realizaron antes del aprendizaje. Los siguientes conjuntos de datos del repositorio de la UCI se utilizaron en las pruebas:

- Wisconsin breast cancer (699 casos, 9 atributos y dos clases)
- Cleveland heart disease (303 casos, 13 atributos, 2 clases)
- Appendicitis (106 instancias, 8 atributos, dos clases)
- Iris (150 instancias, 4 atributos, dos clases)
- Wine (178 instancias, 13 atributos, 3 clases)
- Pima indians diabetes (768 instancias, 8 atributos, dos clases)

La **Ilustración 13** presenta la precisión en los datos y en la compresión proporcionada por los algoritmos de selección. La ilustración corresponde al algoritmo de clasificación kNN; los resultados se promediaron en todos los puntos de referencia. El eje horizontal muestra la compresión del conjunto de entrenamiento en porcentajes (100% = el conjunto de entrenamiento completo). El eje vertical corresponde a los cambios de precisión en el conjunto de pruebas para un algoritmo de selección de instancia. El nivel cero se define por la precisión obtenida por el algoritmo de clasificación determinado entrenado en todo el conjunto de entrenamiento. Por ejemplo, "+2" en el eje vertical significa que la precisión promedio de la prueba es un 2% mejor que el algoritmo base entrenado con el conjunto de datos completo.

Como puede verse en la **Ilustración 13**, el rendimiento de ELH o ELGrow es menor que el de RMHC, y el rendimiento del algoritmo DEL se encuentra entre ellos. Junto a los algoritmos MC1 y DEL están los algoritmos ICF, DROP3 y DROP5. Algoritmos como LVQ, RMHC, Explore, MC1, DEL y DROP2-4 tienen el mejor rendimiento basado en una mayor precisión y reducción del conjunto de datos (las partes superiores izquierda de la ilustración para el clasificador kNN).

En la mayoría de los casos, Explore extrae un conjunto extremadamente pequeño (unos pocos) de instancias y son muy efectivos en los casos en los que Explore no tiene la máxima precisión en los datos invisibles, puede ser sustituido por los algoritmos RMHC, MC1 o

DROP2-4. Por ejemplo, en el dataset Appendicitis, la precisión de kNN con el algoritmo de selección Explore fue de 82.7%, y con MC1 fue de 86.7% (el rendimiento básico de kNN fue de 86.4%).

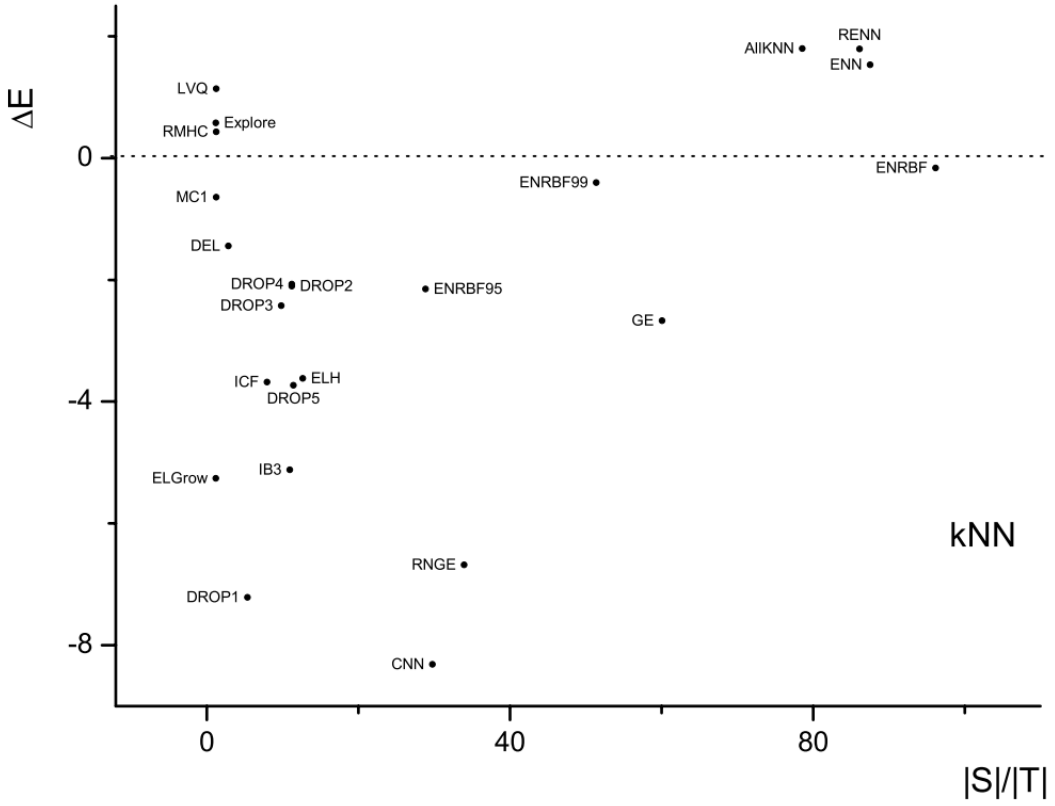


Ilustración 13. Clasificador: kNN (ΔE = 0 corresponde a una precisión de 85.77%)

En [85] concluyen que los algoritmos de selección por prototipos Explore, RMHC, LVQ, MC1, DROP2-4 y DEL son los más efectivos. Estiman automáticamente el número de instancias para la compresión óptima del conjunto de entrenamiento y alcanzan una alta precisión en los datos. Los algoritmos de RMHC y LVQ también terminaron las pruebas con un alto nivel de precisión en los datos. En el grupo de filtros de ruido, el algoritmo ENN y RENN se ubicaron en la parte superior, pero con poca reducción del conjunto de datos.

De acuerdo a los estudios realizados en el estado del arte, Zhao et al [10], concluyen que el algoritmo C-Pruner obtiene mejores resultados entre los algoritmos IB3, DROP3 y KNN. Siendo ligeramente menor que KNN (2%), pero lo supera en más del 88% en almacenamiento. Por otro lado, el estudio de Grochowski et al [85] muestra que el algoritmo Explore, y RMHC sobresale entre los algoritmos de selección de prototipos; Y el algoritmo ENN se destaca entre los algoritmos con filtro de ruido.

Con el fin de verificar los beneficios del algoritmo propuesto, se llevaron a cabo experimentos con mayor número de datasets (en este caso 26), extraídos del Repositorio de Conjuntos de Datos de Aprendizaje Automático de la Universidad de California en Irvine [86], los resultados se compararán con algoritmos del estado del arte: Random Mutation Hill Climbing (RMHC), Encoding Length Heuristic (ELH), Condensed Nearest Neighbour (CNN) y Edited Nearest Neighbour (ENN). Se usaron dos métricas de efectividad, la primera basada en el porcentaje de instancias correctamente clasificadas para evaluar la calidad de la clasificación, y la segunda, es la tasa de reducción de instancias.

Página intencionalmente dejada en blanco

4. SELECCIÓN DE INSTANCIAS BASADO EN CAs –versión 1

4.1 Algoritmo propuesto

La versión 1 del algoritmo propuesto toma el dataset T, calcula el número de filas, para posteriormente buscar en un repositorio (base de datos) el Covering Array (CA) que tenga cubrimiento para este dataset con base en la fuerza f que el usuario defina. Después recorre una a una, las filas del CA, creando un dataset S basado en la configuración de cada fila del CA (este define qué instancias se incluyen o no en el nuevo datasets). Sobre cada dataset se ejecuta el algoritmo de clasificación o MAS seleccionado (kNN, C4.5, Random Forest o cualquier otro), guardando el resultado de la ejecución y pasa a la siguiente fila del CA. Al terminar de recorrer todas las filas del CA, ordena los resultados utilizando el frente de Pareto para seleccionar las mejores soluciones y deja al usuario la decisión de escoger cual considera la mejor.

En la experimentación se asume que el usuario escoge aquella solución que tiene el menor porcentaje de instancias incorrectamente clasificadas (pct) y el mejor número de registros (mayor reducción del dataset), teniendo en cuenta el mismo peso para ambos criterios, aunque el usuario en la vida real puede determinar dar mayor peso a uno de estos dos criterios. La **Tabla 9** resume el algoritmo en su primera versión, para la selección de instancias basado en los arreglos de cobertura.

Entradas:	T	Dataset de entrenamiento con todos los datos
	f	Fuerza del covering array
	MAS	Algoritmo de clasificación a ejecutar
Salida:	Un dataset S con el dataset de muestra seleccionado y el porcentaje de instancias correctamente clasificadas (%pct), y el número de instancias seleccionadas en S.	

```
Inicio
    listaResultados = Vacío
    coveringArray = ObtenerCoveringArray (T, f)
    Para cada fila ∈ coveringArray haga
        S = crearCasoPrueba(T, fila)
        pct = ejecutarAlgoritmoClasificacion(S, MAS)
        Agregar pct, S y |S| a listaResultados
    Fin Para
    mejoresResultados = seleccionarResultados(listaResultados)
Fin
```

Tabla 9. Algoritmo de reducción de instancias basado en Covering Arrays

La función *ObtenerCoveringArray (T, f),* recibe como parámetro el dataset completo y la fuerza del covering array. Esta función permite identificar cual covering array del repositorio (actualmente este repositorio cuenta con CAs binarios de fuerza 2 a 6 hasta con 24310 filas para fuerza 2, hasta 10648 filas para fuerza 3 y así sucesivamente hasta 1373 filas para fuerza 6) se selecciona teniendo en cuenta la fuerza seleccionada y el número de registros

del dataset. Si un dataset tiene un tamaño que supera el covering array que provee el máximo número de casos de prueba, se debe dividir dicho dataset y buscar el CA ideal para cada partición, lo que implica ejecutar el mismo proceso de selección de instancias por cada partición (esto a la fecha no ha sido evaluado con la experimentación realizada y se establece como trabajo futuro).

La función *crearCasoPrueba(T, fila),* recibe como parámetro el dataset completo y una fila del CA, dicha fila está compuesta por columnas binarias, las cuales indican presencia o ausencia de una instancia. La función lee cada columna de la fila actual del CA, y si el valor es 1, indica que la instancia de la misma posición debe estar presente en el nuevo caso de prueba o dataset, como se ilustra paso a paso, desde la **Ilustración 14** hasta **Ilustración 17**.

En la **Ilustración 14**, el algoritmo toma la primera fila del CA, la cual puede tener valores 0 ó 1 en cada celda (CA binario), donde el 1, indica la presencia de la instancia en el nuevo dataset y 0 su ausencia. Si la fila no tiene ningún 1, como es el caso de esta ilustración, el nuevo dataset queda vacío, razón por la cual no se le aplica el MAS y básicamente se ignora.

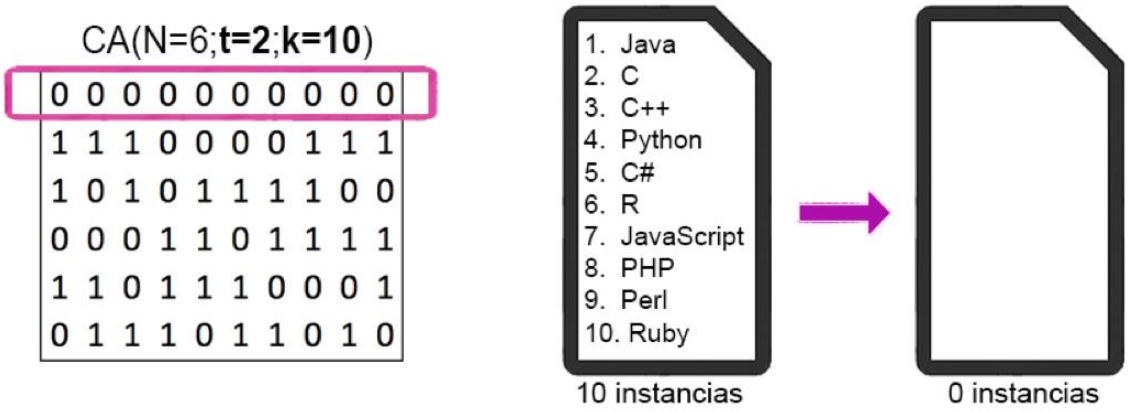


Ilustración 14. Función CrearCasoPrueba, primera fila del CA. Ilustración propia

En la **Ilustración 15**, el algoritmo ha avanzado a la segunda fila del CA. la cual contiene 6 unos, en las posiciones 1, 2, 3, 8, 9, 10, respectivamente, lo cual genera un nuevo dataset con las instancias de las posiciones marcadas por el CA en el dataset original.

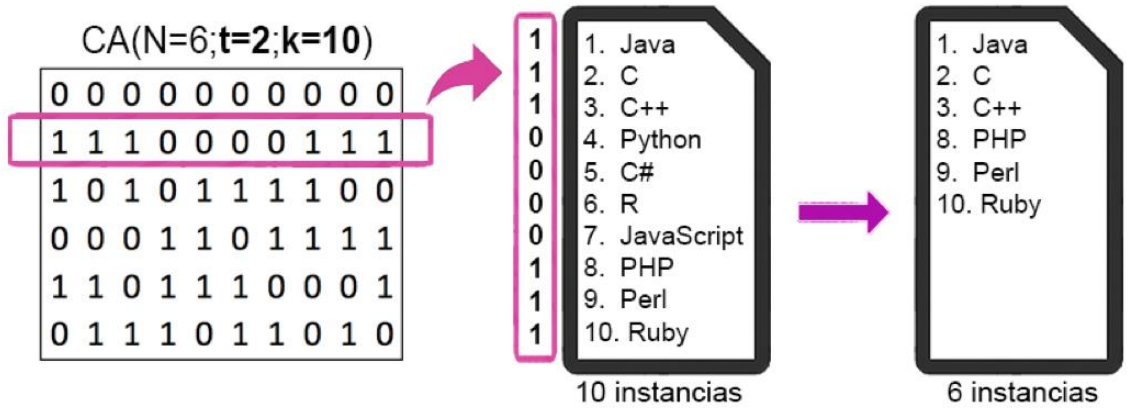


Ilustración 15. Función CrearCasoPrueba, segunda fila del CA. Ilustración propia

Esto ocurre con cada fila del CA, generando un nuevo caso de prueba o dataset, por cada fila. Para este ejemplo, se generan 6 datasets (técnicamente 5 porque el primero quedo vacío). Como se puede ver en la **Ilustración 16** y la **Ilustración 17**.

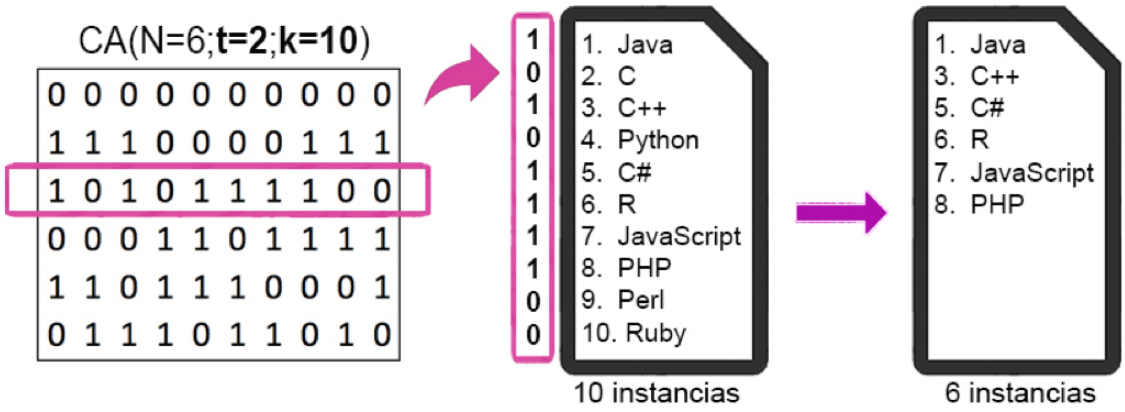


Ilustración 16. Función CrearCasoPrueba, tercera fila del CA. Ilustración propia

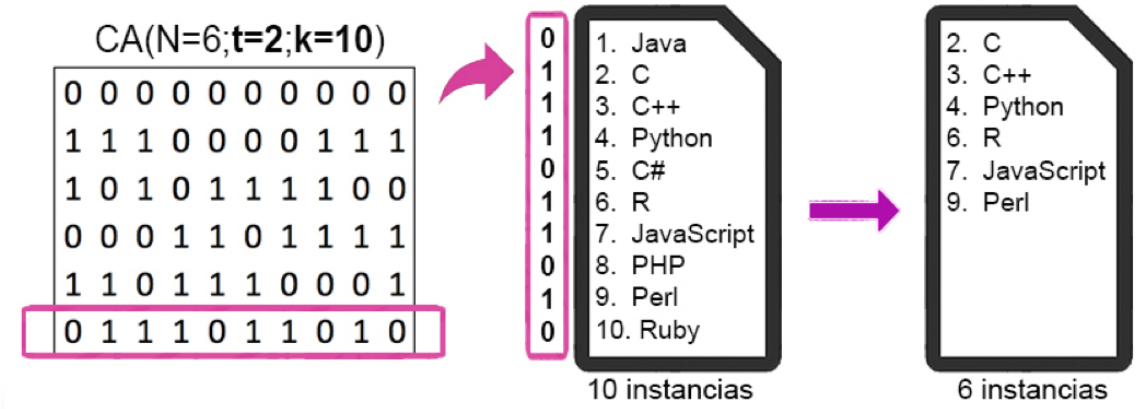


Ilustración 17. Función CrearCasoPrueba, sexta fila del CA. Ilustración propia

Otra función importante en el algoritmo se denomina seleccionarResultados(listaResultados). Esta rutina permite identificar cual es la mejor solución obtenida entre todos los casos de prueba, para ello utiliza un frente de Pareto como se muestra en **Ilustración 18**. En donde el eje Y, corresponde al porcentaje de error (instancias incorrectamente clasificadas), y el eje X, muestra el porcentaje del tamaño original del dataset, es decir, si el dataset original tiene 1000 instancias, y se genera un dataset de 350 instancias, este tendrá un 35% del tamaño original. El algoritmo retorna la lista de las mejores soluciones (frente de Pareto), aunque es una decisión que se le puede dejar al usuario, como se mencionó previamente, para efectos de dar una única respuesta en la experimentación, en este trabajo se seleccionó como mejor, la que tenga menor distancia euclidiana al origen [87], con lo que se da igual ponderación a los dos criterios.

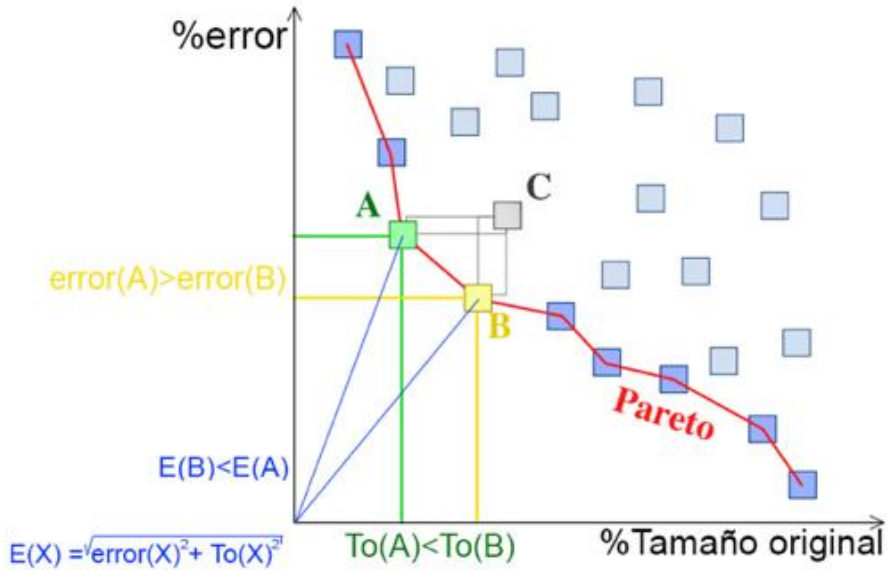


Ilustración 18. Frente de Pareto. Ilustración propia

En la **Ilustración 18**, los puntos representan los casos de prueba (datasets reducidos) que se generaron a partir del CA, el punto C no está en el Frente de Pareto dado que es dominado por A y B, los cuales a su vez no son dominados por ningún otro punto, por lo tanto, están en el Frente. Para los experimentos, el punto seleccionado como el mejor, será aquel que tenga la menor distancia Euclidiana, en este caso es el punto B.

El algoritmo propuesto en su versión 1, se diferencia principalmente frente a la versión final de la tesis, en que, en la versión final el algoritmo se ejecuta iterativamente hasta que se supera un umbral máximo de error tolerado, este umbral es especificado por el usuario. Además, el usuario puede especificar el peso de las dos variables: porcentaje de instancias correctamente clasificadas, y el tamaño del dataset, con ello se elige automáticamente la mejor opción en el frente de Pareto.

4.2 Evaluación del algoritmo

4.2.1 Conjunto de datos para la validación

La experimentación se realizó utilizando 26 datasets reconocidos como conjuntos de prueba en varios estudios, a saber: Banknote, Blood, Car, Climate, Contraceptive, Dermatology, Diabetes, Ecoli, Fertility, Glass, Haberman, Ionosphere, Iris, Leaf, Libras, Planning, Qsarbiodegradation, Seeds, Segment, Sonar, Soybean, Spectf, Vowel, Wine, Yeast y Zoo, como se muestra en **Tabla 10**. Estos datasets se seleccionaron porque han sido utilizados ampliamente en las investigaciones sobre reducción de instancias, además el acceso a ellos es libre. La descripción y una información más detallada de cada uno de los dataset se puede encontrar en el Repositorio de la Universidad de California en Irvine (UCI) [86].

4.2.2 Covering Arrays utilizados

Los Covering Array que se utilizaron para los experimentos fueron suministrados por el Laboratorio de Tecnologías de la Información del CINVESTAV (Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional de Tamaulipas, México). La

evaluación y análisis de los resultados utilizaron CA de fuerza 2,3,4,5, con alfabeto binario. La comparación de los resultados buscaba identificar cual es la fuerza que arroja mejores resultados en cuanto a clasificación y tamaño del dataset, utilizando el algoritmo propuesto. Los dataset utilizados en este trabajo tienen diferentes números de instancias, por ello, cada uno requiere un CA especifico (valor k del CA).

Dataset	#Instancias	#Características	#Clases
Banknote	1372	4	2
Blood	748	4	2
Car	1728	6	4
Climate	900	18	2
Contraceptive	1473	9	3
Dermatology	366	34	6
Diabetes	768	8	2
Ecoli	336	7	8
Fertility	100	9	2
Glass	214	9	7
Haberman	306	3	2
Ionosphere	351	34	2
Iris	150	4	3
Leaf	340	15	36
Libras	360	90	15
Planning	304	12	2
QSARBiodegradation	1055	41	2
Seeds	210	7	3
Segment	2310	19	7
Sonar/Connectionist	208	60	2
Soy Bean	683	35	19
Spectf	267	44	2
Vowel	990	10	11
Wine	178	13	3
Yeast	1484	8	10
Zoo	101	17	7

Tabla 10. Datasets de prueba

La cantidad de pruebas necesarias para la evaluación del algoritmo propuesto depende del tamaño del dataset (número de instancias), y de la fuerza del CA que se utilizará. En la **Ilustración 19** y la **Ilustración 20**, se muestra en el eje horizontal la fuerza de los CA y en el eje vertical el número de pruebas realizadas para cada dataset.

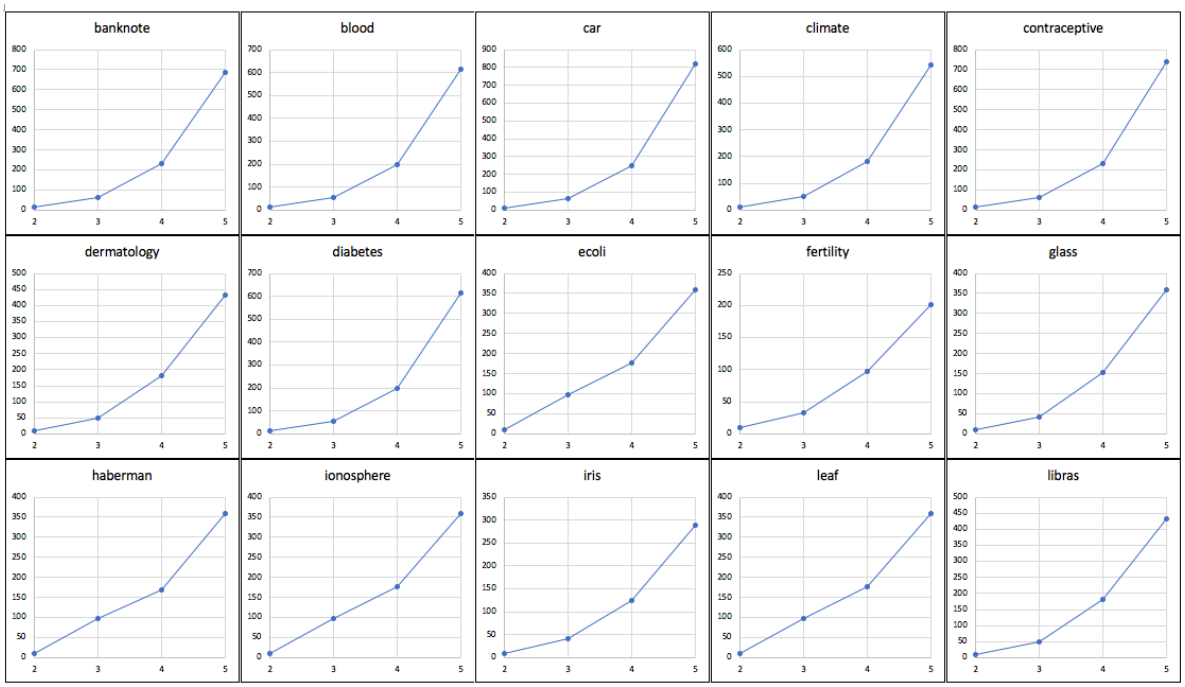


Ilustración 19. Número de pruebas requeridas por nivel de fuerza en cada dataset

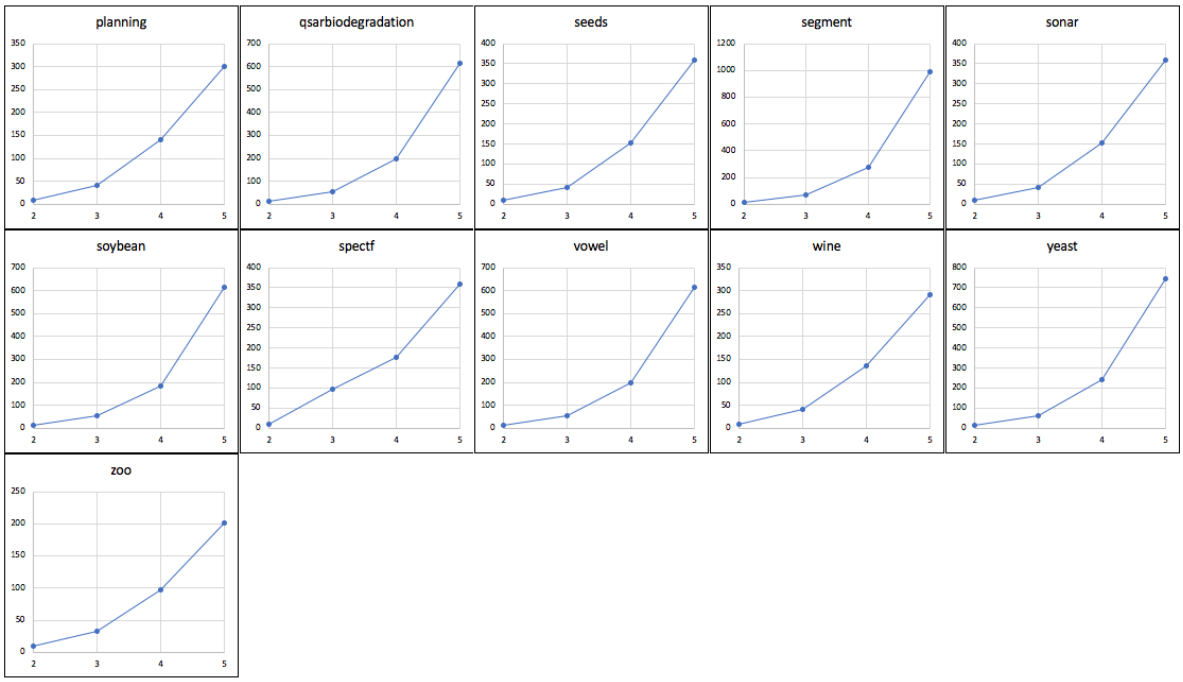


Ilustración 20. Número de pruebas requeridas por nivel de fuerza en cada dataset

4.2.3 Comparación de los resultados experimentales

Para generar los casos de prueba basados en los CA, se usó una fuerza de 2, 3, 4 y 5, ya que estas fuerzas han demostrado resultados satisfactorios en términos de efectividad, en experimentos similares en otras áreas [88][6][89]. En la **Ilustración 21** el eje Y representa

la cantidad de casos de prueba, por lo que se puede evidenciar que a medida que incrementa la fuerza, también lo hacen los casos de prueba.

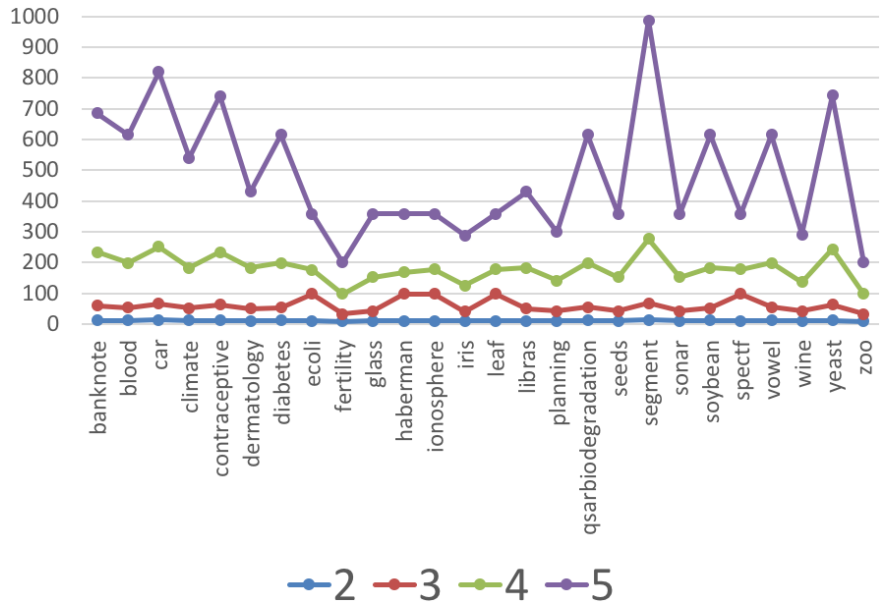


Ilustración 21. Número de casos de prueba según la fuerza

En la **Tabla 11** y en la **Tabla 12** se presenta el resumen de los resultados obtenidos al ejecutar el algoritmo en su primera versión, utilizando los 26 datasets, cada uno evaluado con fuerza 2,3,4,5. Además, se muestra la evaluación utilizando los clasificadores C4.5 (J48 de Weka) y 1NN (kNN con k=1 en Weka), y el valor obtenido por los clasificadores con el dataset completo. Los experimentos se repitieron 31 veces con diferentes semillas para la generación de valores aleatorios, los cuales se usaron para desordenar el dataset y de esa forma estudiar el efecto del orden de las filas en el dataset sobre el algoritmo propuesto.

El menor promedio del porcentaje de error para cada dataset se indica con negrita. Los promedios de error logrados con el algoritmo propuesto que son menores al error del dataset completo se resaltan en amarillo, es decir, que la propuesta aquí planteada mejoró la calidad de los resultados, disminuyendo el error, y además el número de instancias. Por ejemplo, los resultados del dataset Blood al aplicar la propuesta aquí planteada con CA de fuerza 2, 3, 4, 5, y ejecutando el algoritmo C4.5 obtiene los promedios de error 19,230%, 19,076%,17,7419% y 17,846% respectivamente, estos valores son menores que el error obtenido al ejecutar el algoritmo C4.5 sobre el dataset completo (22,727%), en donde el mejor resultado se logra con f4 (17,7419%).

En la **Tabla 11** se muestra que el algoritmo aquí propuesto obtuvo para varios datasets un promedio de porcentaje de error igual o menor que el porcentaje de error utilizando el algoritmo de clasificación directamente sobre el dataset completo. Se generaron 208 resultados del promedio de error, para los 26 datasets (26 por cada fuerza y por cada clasificador), con los algoritmos C4.5, y 1NN y CA de fuerza 2, 3, 4 y 5, de los cuales 148 son menores o iguales (resaltados en amarillo) que el error obtenido del dataset completo; 38 corresponden a fuerza 2 (18 de C4.5 y 20 de 1NN), 34 son de fuerza 3 (16 de C4.5 y 18 de 1NN), 37 son de fuerza 4 (21 de C4.5 y 16 de 1NN) y 39 son de fuerza 5 (19 de C4.5 y 20 de 1NN). También se puede observar que el promedio de los errores obtenidos con los

CA de f2, f3, f4 y f5 para el algoritmo C4.5 son de 18,039%, 17,583%, 16,891%, 16,738% respectivamente; de igual manera para el algoritmo 1NN son de 18.349%, 17,995%, 17,458% y 17,554%. Se puede observar que los resultados del promedio de error, con los CA de f5 y el clasificador C4.5 son ligeramente mejores frente a los demás; Además, los CA de f4 y el clasificador 1NN mejora ligeramente los resultamos frente a las otras fuerzas.

	Promedio %error									
	C4.5					**1NN**				
	f2	**f3**	**f4**	**f5**	**Error dataset completo**	**f2**	**f3**	**f4**	**f5**	**Error dataset completo**
Banknote	1,614	2,390	2,2917	**0,954**	1,385	**0,000**	0,176	**0,000**	**0,000**	0,146
Blood	19,230	19,076	**17,7419**	17,846	22,727	23,441	23,234	**21,369**	22,044	28,342
Car	11,463	**10,737**	12,257	10,921	8,449	11,225	**11,114**	12,802	11,769	7,581
Climate	5,974	5,779	4,4041	**3,676**	7,593	8,656	8,149	7,316	**6,617**	11,852
Contraceptive	45,955	46,717	46,063	**45,387**	48,540	52,434	52,563	**52,005**	52,376	55,533
Dermatology	4,646	3,498	3,6142	**3,526**	5,738	3,965	3,479	**3,047**	3,829	6,284
Diabetes	23,311	23,004	**21,0999**	22,482	28,776	25,893	25,700	**23,720**	24,713	29,688
Ecoli	15,085	15,092	16,5615	**13,538**	17,560	13,965	14,118	16,856	**13,730**	19,048
Fertility	7,022	6,098	7,433	**3,830**	16,000	9,384	7,597	7,408	**3,302**	18,000
Glass	29,831	**24,677**	31,0787	24,733	34,579	27,377	**24,696**	29,281	24,168	30,374
Haberman	23,170	22,484	**20,502**	20,846	29,739	26,553	26,171	**25,182**	25,929	31,699
Ionosphere	8,122	8,047	**5,6471**	7,010	10,256	11,098	11,021	**8,476**	10,064	13,960
Iris	2,915	5,033	**1,0768**	6,267	4,000	1,810	1,552	**0,273**	2,537	6,000
Leaf	42,608	42,268	37,1702	**39,312**	37,941	**85,537**	88,216	87,131	88,297	96,177
Libras	41,525	40,992	39,1734	**38,727**	36,667	19,076	18,885	**17,932**	18,322	15,000
Planning	23,366	21,840	**20,9098**	26,201	28,571	29,891	28,392	**25,306**	33,355	32,967
Qsarbiodegradation	15,137	14,219	**13,779**	14,729	15,640	15,423	14,456	**13,806**	14,286	15,829
Seeds	6,605	**5,270**	9,2063	5,602	10,952	3,610	**2,575**	4,667	3,031	7,619
Segment	4,360	4,218	3,81655	**3,526**	3,550	3,994	3,631	3,429	**3,108**	3,074
Sonar	22,817	**20,316**	24,4406	21,071	28,846	12,746	**11,022**	15,565	12,177	12,981
Soybean	10,804	10,634	8,994	**8,883**	9,224	8,953	8,813	**7,686**	7,764	8,492
Spectf	16,320	16,102	**12,7877**	15,089	13,467	19,411	19,906	**16,944**	18,711	17,765
Vowel	30,836	30,000	**27,4695**	29,005	22,626	11,213	10,337	**5,933**	8,668	0,808
Wine	6,642	10,301	**4,2381**	8,883	8,427	1,892	4,068	**0,527**	3,194	3,933
Yeast	43,049	42,883	42,633	**40,728**	44,609	46,646	45,661	45,780	**44,324**	49,124
Zoo	6,616	5,484	4,7871	**2,414**	7,921	2,730	2,333	1,462	**0,102**	5,941
Promedio	18,039	17,583	16,891	**16,738**	19,376	18,343	17,995	**17,458**	17,554	20,316

Tabla 11. Resultados promedio de error f2-f5

La **Tabla 12** muestra el promedio del porcentaje de instancias que se logra obtener ejecutando el algoritmo propuesto. Se marca en negrilla, los mejores resultados en cuanto a reducción de instancias. Por ejemplo, para el dataset Car, el dataset obtenido utilizando f4 y el clasificador C4.5 tiene un tamaño que corresponde al 27,54% del dataset original.

Teniendo en cuenta los resultados en la columna “promedio % error” y “Promedio % instancias” de la **Tabla 11** y de la **Tabla 12**, para el método propuesto con fuerza 2-5 y sin aplicar el método (i.e. con el dataset completo), en los 26 datasets se realizó la prueba estadística no paramétrico de Friedman, los resultados de esta prueba se muestran en la **Tabla 13**.

De acuerdo a la **Tabla 13**, con el algoritmo C4.5, para la columna “Promedio % error” el ranking que se obtuvo, ubicó al método propuesto con fuerza 5 en la primera posición, luego el de fuerza 4, además, este ranking fue estadísticamente significativo ya que el valor p de la prueba es menor 0,05, en este caso fue de 4,9952E-5. Para este clasificador se toma como mejor opción al método propuesto con fuerza 4, ya que con este se obtiene un error similar al de fuerza 5 y adicionalmente el número de instancias que genera con respecto al dataset original es menor, quedando de segundo en el ranking siendo superado solo por fuerza 2.

	Promedio %Instancias								
	C4.5				1NN				
	f2	f3	f4	f5	f2	f3	f4	f5	Instancias dataset completo
Banknote	46,50%	35,22%	**29,52%**	49,93%	39,33%	**21,43%**	23,41%	49,93%	1372
Blood	**41,44%**	45,44%	45,45%	47,88%	**40,77%**	45,39%	45,75%	47,97%	748
Car	43,03%	45,47%	**27,54%**	28,21%	43,07%	45,45%	33,50%	**28,85%**	1728
Climate	**31,85%**	41,59%	35,84%	49,81%	**29,47%**	42,07%	36,56%	49,81%	540
Contraceptive	44,11%	28,14%	**24,81%**	49,71%	44,29%	**24,79%**	25,40%	49,72%	1473
Dermatology	42,55%	45,34%	**37,60%**	47,47%	40,91%	45,14%	**35,90%**	47,10%	366
Diabetes	**41,11%**	45,37%	46,04%	47,92%	**41,42%**	45,42%	46,36%	47,98%	768
Ecoli	41,90%	45,23%	**22,36%**	48,36%	37,57%	44,90%	**21,84%**	48,45%	336
Fertility	**36,71%**	45,58%	47,74%	48,73%	**40,19%**	45,39%	48,03%	48,70%	100
Glass	48,01%	47,63%	**34,31%**	45,46%	47,47%	47,83%	**30,25%**	45,65%	214
Haberman	31,25%	44,88%	**24,04%**	47,17%	33,74%	45,19%	**19,01%**	46,76%	306
Ionosphere	**42,10%**	45,46%	49,66%	49,02%	**41,32%**	45,45%	49,56%	49,13%	351
Iris	35,20%	44,37%	48,30%	**20,13%**	30,28%	33,03%	48,06%	**14,27%**	150
Leaf	**44,73%**	45,47%	49,42%	48,61%	**29,24%**	45,26%	49,45%	48,62%	340
Libras	46,36%	45,44%	**42,65%**	47,42%	47,69%	45,37%	**42,29%**	47,52%	360
Planning	37,47%	46,01%	48,35%	**33,39%**	38,59%	46,67%	48,67%	**35,82%**	182
Qsarbiodegradation	**40,56%**	45,46%	45,75%	48,84%	**40,01%**	45,43%	45,45%	48,91%	1055
Seeds	40,00%	46,71%	**30,55%**	45,10%	40,00%	47,76%	**31,84%**	44,95%	210
Segment	44,70%	**45,45%**	43,80%	49,86%	**44,09%**	45,45%	45,49%	49,59%	2310
Sonar	39,95%	47,12%	**35,64%**	44,82%	40,12%	48,19%	**31,93%**	45,05%	208
Soybean	**42,97%**	45,44%	44,82%	47,83%	**42,03%**	45,50%	41,60%	47,91%	683
Spectf	**41,48%**	45,47%	49,61%	49,00%	**43,18%**	45,62%	49,53%	48,93%	349
Vowel	46,82%	**45,45%**	48,39%	48,64%	**45,44%**	45,45%	54,25%	48,72%	990
Wine	**38,96%**	41,75%	48,46%	39,11%	**36,23%**	38,89%	48,66%	38,71%	178
Yeast	45,29%	31,55%	**24,01%**	49,87%	43,85%	29,66%	**23,10%**	49,87%	1484
Zoo	**43,92%**	45,42%	48,20%	49,31%	**42,96%**	45,26%	48,26%	48,86%	101
Promedio	41,50%	43,71%	**39,73%**	45,45%	40,13%	42,54%	**39,39%**	45,30%	

Tabla 12. Resultados promedio tamaño de instancias f2-f5

Datos evaluados	Algoritmo de clasificación	Fuerza	Ranking	Valor P
Promedio % error	1NN	2	3,5385 (4)	1.1947E-5
		3	3,0385 (3)	
		4	**2,0769 (1)**	
		5	2,3077 (2)	
		Dataset completo	4,0385 (5)	
	C4.5	2	3,6923 (4)	4.9952E-5
		3	3,0000 (3)	
		4	2,3462 (2)	
		5	**2,1154 (1)**	
		Dataset completo	3,8462 (5)	
Promedio % instancias	1NN	2	**1,8077 (1)**	0.001175914
		3	2,5385 (3)	
		4	2,4231 (2)	
		5	3,2308 (4)	
	C4.5	2	**1,8462 (1)**	2.65504E-4
		3	2,5769 (3)	
		4	2,2308 (2)	
		5	3,3462 (4)	

Tabla 13. Clasificaciones promedio de la prueba de Friedman

La misma prueba estadística fue usada sobre los resultados de 1NN, el ranking en relación con el "promedio % error" fue: 1-método propuesto fuerza 4, 2-método propuesto fuerza 5, 3-método propuesto fuerza 3, 4-método propuesto fuerza 2 y finalmente 5-dataset

completo, con un valor p de 1,1947E-5 que lo hace estadísticamente valido. Como además la fuerza 4 también obtiene el segundo puesto en reducción de instancias, se toma también la fuerza 4 como el valor apropiado para el algoritmo, igual que con C4.5.

Teniendo en cuenta que las pruebas de Friedman fueron significativas, se procedió a realizar el post hoc de Holm con el objetivo de determinar si existen verdaderas relaciones de dominancia entre los resultados obtenidos por los algoritmos.

La **Tabla 14** representa la dominancia de los datos, a partir de los resultados del post hoc de Holm. En donde los datos por debajo de la diagonal azul tienen un 95% de confianza, y los que se encuentran por encima de la diagonal, poseen un 90% de confianza. El símbolo '•' representa, que el método de la fila mejora el método de la columna y el símbolo 'o' representa, que el método de la columna mejora el método de la fila.

Por ejemplo, con los resultados de 'Promedio de %error' en la **Tabla 14** se puede concluir que:

- Los resultados de fuerza 5 dominan a los de fuerza 2 con una confianza de 95% (obviamente con 90% también).
- Los resultados de fuerza 4 dominan a los de fuerza 2 con una confianza de 95% (obviamente con 90% también).
- Los resultados de fuerza 4 y fuerza 5 dominan a los del clasificador con el dataset completo con una confianza de 95% (obviamente con 90% también).

		Promedio % error					**Promedio % instancias**			
		f2	**f3**	**f4**	**f5**	**Completo**	**f2**	**f3**	**f4**	**f5**
1NN	**f2**			o	o					o
	f3									
	f4	•				•				
	f5	•				•	•			
	Completo			o	o					
C4.5	**f2**			o	o					•
	f3									
	f4	•				•				•
	f5	•				•	o		o	
	Completo			o	o					

Tabla 14. Resultados post hoc de Holm

Finalmente se realizó la prueba no paramétrica de rangos con signo de Wilcoxon. Con el fin de comparar los resultados y examinar si con esta prueba se aceptan otras relaciones de dominancia que no acepta Holm. Los resultados se muestran en la **Tabla 15**.

Como resultado se obtuvo, que tanto los resultados del método en fuerza 4, como en fuerza 5, dominan los resultados obtenidos con el dataset completo, con un 95% de significancia y que entre ellos (fuerza 4 y fuerza 5) no se puede establecer relación de dominancia. Teniendo en cuenta esto último, también se seleccionó el método propuesto con fuerza 4 por las mismas razones expresadas en la prueba estadística de C4.5, usa menos tiempo de ejecución y reduce más filas del dataset original.

		Promedio % error					Promedio % instancias			
		f2	f3	f4	f5	Completo	f2	f3	f4	f5
1NN	f2		o	o	o	•		•		•
	f3	•		o	o	•	o			•
	f4					•				•
	f5	•	•			•	o	o	o	
	Completo	o	o	o	o					
C4.5	f2		o	o	o	•		•		•
	f3	•		o	o	•	o		o	•
	f4	•				•				•
	f5	•	•			•	o	o	o	
	Completo			o	o					

Tabla 15. Resultados prueba de los rangos con signo de Wilcoxon

4.3 Análisis de resultados

A continuación, en la **Ilustración 22**, se muestra un diagrama de cajas y alambres del 'Porcentaje % error', para los resultados de 31 repeticiones del mismo experimento de la propuesta sobre el dataset Haberman, utilizando CA de fuerza 2 y el algoritmo de clasificación 1NN. Este diagrama consiste en un rectángulo, el cual está dividido por una línea vertical indicando la posición de la mediana, a la derecha el tercer cuartil y a la izquierda el primer cuartil, el segundo cuartil coincide con la mediana. El diagrama tiene como extremos el valor mínimo (izquierda) y máximo (derecha) de los resultados. Las líneas que se extienden del rectángulo, se denominan alambres o bigotes, los cuales tienen un límite de prolongación. Cada dato que no se encuentre en ese rango, se dibuja individualmente, como un valor atípico. En la **Ilustración 22**, la parte izquierda de la caja es mayor que la ubicada a la derecha, esto implica que los porcentajes de error tienden a estar más entre 24,9699% y 27,0968%. El alambre de la izquierda muestra el resultado menor esperado 22,2222%, sin embargo, existe un resultado atípico, que se aleja de los valores esperados, 9.2593%, que es mucho menor a la mayoría de los resultados.

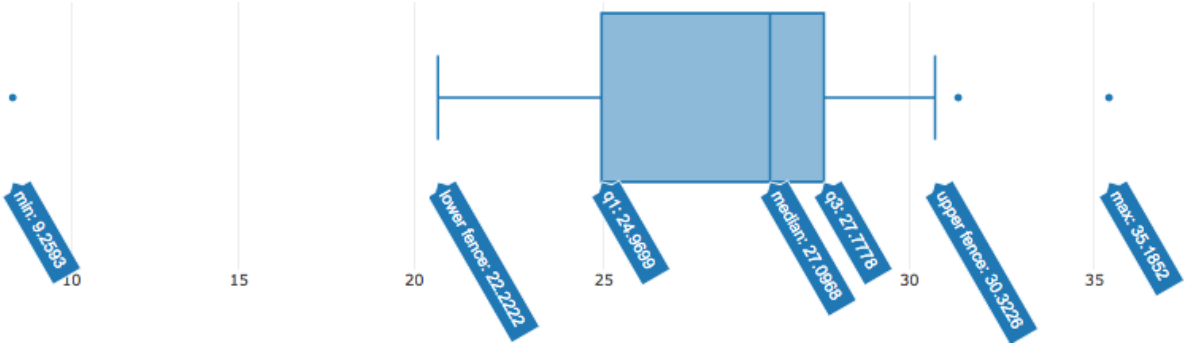


Ilustración 22. Diagrama de cajas y alambres resultado dataset Haberman

La **Ilustración 23**, la **Ilustración 24**, la **Ilustración 25** y la **Ilustración 26**, representan el resumen del porcentaje de error, después de las 31 repeticiones de la propuesta planteada, utilizando CA de fuerza 2 y fuerza 4, con los algoritmos C4.5 y 1NN. En las cuales se puede observar que, la mayoría de los resultados del porcentaje de error tienden a estar entre 0% y 30%. Al comparar las ilustraciones por dataset sin importar el clasificador son bastante similares.

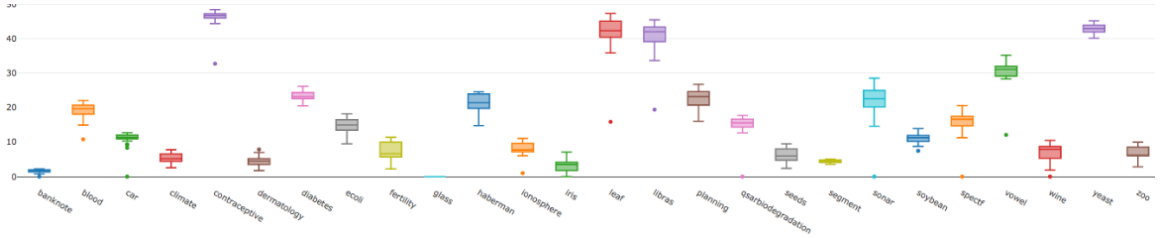


Ilustración 23. Porcentaje de error con C4.5 usando CAs de fuerza 2

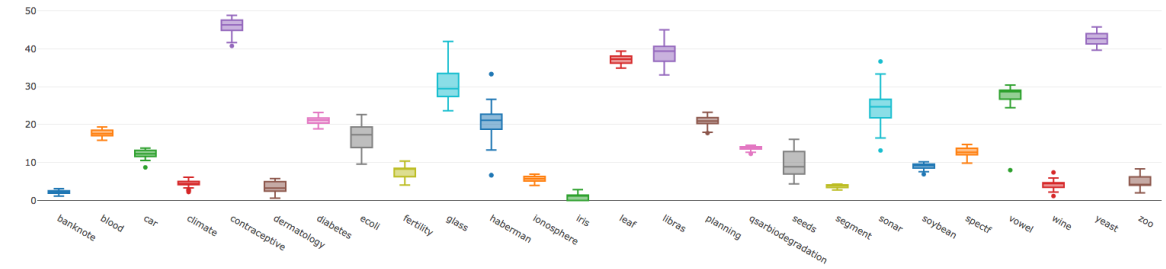


Ilustración 24. Porcentaje de error con C4.5 usando CAs de fuerza 4

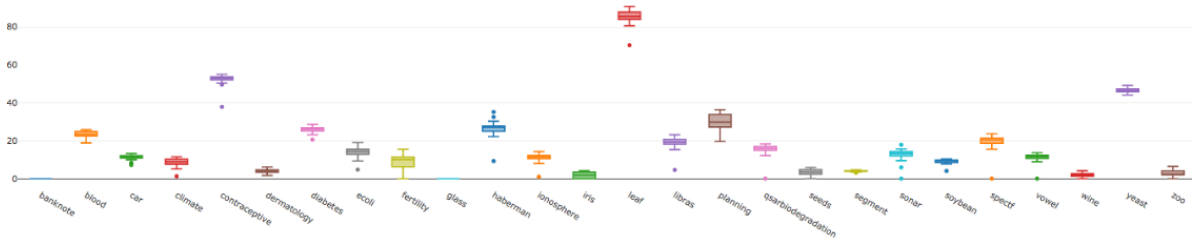


Ilustración 25. Porcentaje de error con 1NN usando CAs de fuerza 2

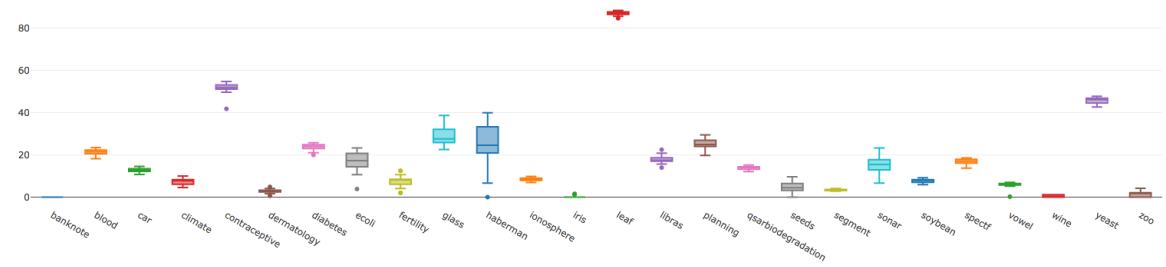


Ilustración 26. Porcentaje de error con 1NN usando CAs de fuerza 4

En la **Ilustración 27**, la **Ilustración 28**, la **Ilustración 29**, y la **Ilustración 29** se muestra el porcentaje de instancias con respecto al dataset original (a qué porcentaje queda reducido el dataset original?) de cada uno de los 26 datasets. En ellas se puede observar que el tamaño de cada dataset resultante tiende al mismo tamaño en todos. Esto se debe, a que en la mayoría de los casos se usaron CAs óptimos, por lo tanto, la cantidad de filas seleccionadas en cada registro del CA en general tiende a ser el mismo valor. Por el ello el "Promedio %Instancias" es muy similar, dando como resultado, datos que tienen medianas muy similares, y muy pocos diagramas tienen alambres. Los casos con varianza alta son el resultado de usar CAs más grandes de lo requerido en el dataset a los cuales se les quitan algunas columnas y por dicha operación siguen siendo CAs, pero ya no son óptimos. Para evitar esto se puede considerar un paso de optimización de este CA con un algoritmo voraz.

En estas ilustraciones se evidencia que el comportamiento es un tanto diferente de acuerdo con el clasificador que se use.

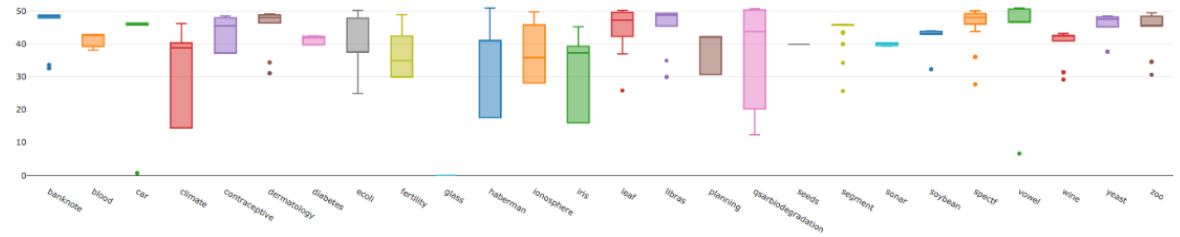


Ilustración 27. Porcentaje de instancias con C4.5 usando CAs de fuerza 2

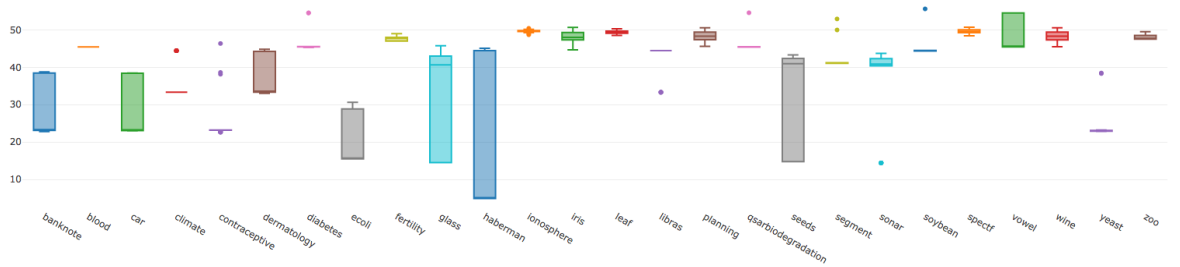


Ilustración 28. Porcentaje de instancias con C4.5 usando CAs de fuerza 4

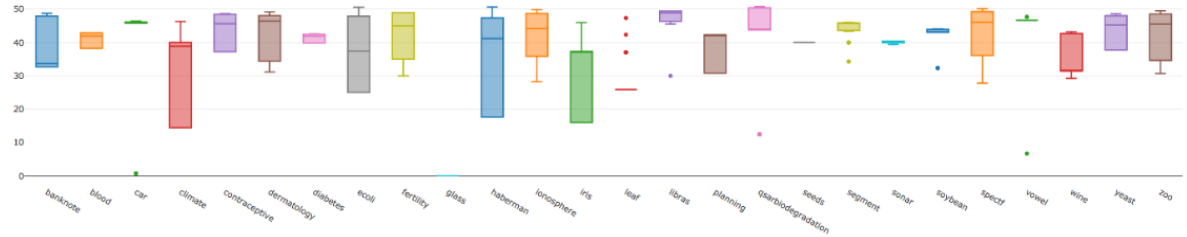


Ilustración 29. Porcentaje de instancias con 1NN usando CAs de fuerza 2

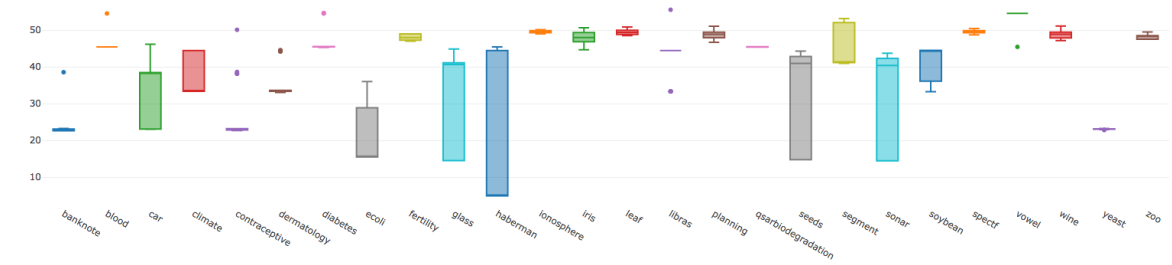


Ilustración 30. Porcentaje de instancias con 1NN usando CAs de fuerza 4

Los resultados de este capítulo permiten inferir que el algoritmo propuesto logra mejores resultados (en promedio de 2,5%) de calidad (exactitud) cuando se ejecuta el clasificador 1NN o C4.5 con el dataset reducido con fuerza 4 en relación con el dataset completo. Además, que el tamaño del dataset reducido llega a ser en promedio un 39,5% del dataset original. Con esto se concluye que la hipótesis h1 se acepta.

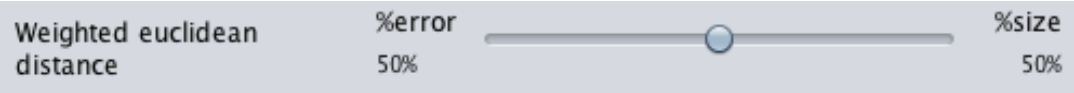
Página intencionalmente dejada en blanco

5. SELECCIÓN DE INSTANCIAS BASADO EN CAs –versión 2

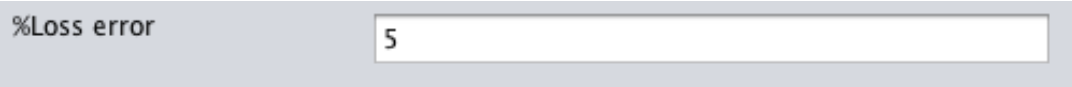
5.1 Evaluación del algoritmo

Para obtener la segunda versión del algoritmo se realizaron dos modificaciones, con el fin de buscar una mejor solución en cuanto a reducción del error y del tamaño del dataset. Las adecuaciones fueron las siguientes:

1. Se permite al usuario asignar directamente un peso (prioridad) al porcentaje de error y el porcentaje del tamaño del dataset, con el fin de calcular la distancia euclidiana ponderada en el frente de Pareto y seleccionar la mejor solución.



2. El usuario puede seleccionar un umbral de pérdida de error, esto, para realizar recursivamente el algoritmo, buscando reducir el dataset y sin superar dicho umbral de error.



   Por ejemplo, si el error que se obtiene al ejecutar el clasificador sobre el dataset completo es 15%, y al ejecutar el algoritmo el usuario digita 5, en '%Loss error'; el algoritmo se ejecutará recursivamente hasta que la solución obtenida no supere el error del 20% (15% + 5).

En la **Tabla 16** se describe el algoritmo en su segunda versión.

Con el llamado recursivo de la primera versión de la propuesta, en esta segunda versión se busca reducir tanto como sea posible el tamaño del dataset manteniendo una perdida máxima de error. Esta segunda versión se denominó "Instance Selection by Covering Array" (ISCA).

La función **seleccionarMejorResultado,** recibe como parámetro, una lista con todos los resultados de un CA, y el peso del porcentaje de error. Cada resultado de la lista corresponde a la evaluación del algoritmo de clasificación sobre un dataset, el cual es generado por cada fila del CA. Se selecciona el mejor resultado, el cual se obtiene, calculando la distancia euclidiana del porcentaje de error (pct) y el porcentaje de instancias (|S|), esta distancia tiene en cuenta el peso del porcentaje de error, como se muestra en la **Ecuación 1**.

Otra función importante es, **validarSolución,** la cual recibe tres parámetros: resultado, el error original (dataset completo) y el umbral de perdida de error. Esta función se encarga de verificar si el resultado obtenido ha superado el umbral de perdida de error, de no ser

así, se ejecuta recursivamente el algoritmo con este nuevo dataset, hasta que se supere el umbral. Al finalizar las iteraciones, la función **seleccionarSolucion,** se encarga de elegir la solución más pequeña y que no haya superado el umbral del error. Dando como resultado un dataset mucho mas pequeño que el obtenido en la primera ejecución y con un error entre el umbral seleccionado por el usuario.

Entradas:	T	Dataset de entrenamiento con todos los datos
	f	Fuerza del covering array
	MAS	Algoritmo de clasificación a ejecutar
	wError	Prioridad o peso al porcentaje de error [0, 1]
	lossError	Umbral de pérdida de error
Salida:	Un dataset S con el dataset de muestra seleccionado y el porcentaje de instancias correctamente clasificadas (%pct), y el número de instancias seleccionadas en S.	

```
Inicio
    T2 = T
    pctOriginal = ejecutarAlgoritmoClasificacion (T, MAS)
    superaUmbral = false
    listaMejoresResultados = Vacío

    Mientras no superaUmbral
        listaResultadosCA = Vacío
        coveringArray = ObtenerCoveringArray (T2, f)

        Para cada fila ∈ coveringArray haga
            S = crearCasoPrueba(T, fila)
            pct = ejecutarAlgoritmoClasificacion (S, MAS)
            Agregar pct, S y |S| a listaResultadosCA
        Fin Para

        mejorResultadoCA = seleccionarMejorResultado (listaResultados, wError)
        superaUmbral = validarSolución (mejorResultado,pctOriginal,lossError)

        T2 = mejorResultadoCA
        Agregar T2 a listaMejoresResultados
    Fin mientras

    seleccionarSolucion(listaMejoresResultados)
Fin
```

Tabla 16. Algoritmo de reducción de instancias basado en Covering Arrays v2

$$Distancia\ euclidiana\ = \sqrt[2]{(pct^2 * wError) + (|S|^2 * (1 - wError))}$$

Ecuación 1 Calculo de distancia Euclidiana ponderada con los pesos definidos por el usuario

5.2 Evaluación del parámetro '%Loss error'

Como se presentó previamente, en ISCA se incluyó el parámetro '%Loss error'. De acuerdo con la **Ilustración 31** (que se presenta más adelante), se logra evidenciar, que los algoritmos del estado del arte pierden calidad en la clasificación, entre un 0-10%. Por ello se realizan dos ejecuciones del algoritmo, fijando el valor de '%Loss error' a 5% y a 10%.

Lo anterior, con el fin de analizar los resultados y poder determinar la mejor configuración, y con ello poder realizar la comparación de la propuesta con los algoritmos del estado del arte.

Las siguientes tablas muestran los resultados obtenidos al ejecutar el algoritmo sobre los 26 datasets (31 repeticiones en cada uno), con la configuración de perdida de error máxima del 5% y 10%, y con fuerza 2 y fuerza 4. En la **Tabla 17** se utilizó el clasificador 1NN, y en la **Tabla 18** el clasificador C4.5 (J48 en Weka).

Los resultados que se resaltan en amarillo indican que el valor obtenido, tiene una calidad de la clasificación igual o mejor, frente al resultado obtenido utilizando el clasificador sobre el dataset completo. El menor promedio del porcentaje de error, y el menor promedio del tamaño para cada dataset se indica con negrita.

En la **Tabla 17** se puede apreciar en la última columna 'Loss error', los puntos de error que se perdieron frente al error del dataset completo. En promedio al utilizar las configuraciones 5% y 10%, se obtiene una pérdida de 0,55 y 3,181 respectivamente, mientras que el promedio del tamaño del dataset se reduce al 9,93% y 6,51% del tamaño original.

		%Error Promedio		Tamaño Promedio		Tiempo Promedio		Loss error	
Dataset	**Error dataset completo**	**Loss error 5%**	**Loss error 10%**	**5%**	**10%**	**5%**	**10%**	**5%**	**10%**
banknote	0,146	**3,191**	6,684	1,51%	**0,76%**	2,161	**2,124**	**3,045**	6,539
blood	28,877	23,797	23,797	0,13%	0,13%	1,007	**0,878**	-5,080	-5,080
car	6,308	**9,761**	14,477	27,00%	**14,78%**	**2,688**	3,355	**3,454**	8,169
climate	11,481	8,519	8,519	0,19%	0,19%	1,451	**1,411**	-2,963	-2,963
contraceptive	55,872	57,324	**57,298**	0,09%	**0,07%**	3,926	**2,924**	1,452	**1,426**
dermatology	5,738	**8,840**	12,013	6,43%	**4,09%**	0,976	**0,874**	**3,102**	6,275
diabetes	29,427	**29,078**	34,715	0,40%	**0,15%**	**1,354**	1,401	**-0,349**	5,288
ecoli	19,048	**20,584**	23,454	2,88%	**1,96%**	0,426	**0,339**	**1,536**	4,407
fertility	17,000	12,000	12,000	1,00%	1,00%	0,151	**0,108**	-5,000	-5,000
glass	29,907	**32,092**	36,057	10,21%	**6,62%**	0,217	**0,211**	**2,186**	6,150
haberman	32,353	26,471	26,471	0,33%	0,33%	0,283	**0,278**	-5,882	-5,882
ionosphere	13,105	**15,440**	18,252	3,04%	**1,76%**	1,182	**1,102**	**2,334**	5,147
iris	4,667	**5,462**	8,151	4,65%	**2,97%**	0,124	**0,116**	**0,796**	3,484
leaf	97,059	95,294	95,294	0,31%	0,31%	0,692	**0,587**	-1,765	-1,765
libras	13,333	**14,686**	17,625	35,73%	**30,13%**	1,291	**1,187**	**1,353**	4,292
planning	36,813	28,571	28,571	0,55%	0,55%	0,187	**0,177**	-8,242	-8,242
qsarbiodegradation	15,924	**19,774**	23,314	3,39%	**0,68%**	**4,198**	5,446	**3,850**	7,390
seeds	5,714	**9,002**	11,260	3,64%	**1,94%**	0,210	**0,204**	**3,287**	5,545
segment	2,597	**6,548**	10,543	10,43%	**4,15%**	7,266	**7,062**	**3,951**	7,946
sonar	12,500	**12,593**	18,657	26,44%	**15,52%**	0,703	**0,539**	**0,093**	6,157
soybean	8,638	**11,330**	13,947	17,86%	**14,37%**	1,763	**1,537**	**2,692**	5,309
spectf	13,754	**15,769**	22,257	29,61%	**8,74%**	0,982	**2,289**	**2,015**	8,504
vowel	0,606	**1,730**	3,522	51,24%	**46,19%**	**0,879**	1,350	**1,124**	2,916
wine	5,056	**6,814**	8,065	3,12%	**2,21%**	0,235	**0,211**	**1,758**	3,008
yeast	47,035	**50,667**	54,295	1,08%	**0,42%**	3,524	**3,217**	**3,632**	7,260
zoo	3,960	**5,877**	10,380	16,90%	**9,13%**	0,238	**0,109**	**1,916**	6,420
Promedio:	19,882	**20,431**	23,062	9,93%	**6,51%**	**1,466**	1,501	**0,550**	3,181

Tabla 17. Comparación perdida error 5% y 10%, fuerza 2 - 1NN

Por otro lado, al utilizar el clasificador C4.5, el promedio de pérdida de calidad para las configuraciones 5% y 10%, son de 0,857 y 4,582 respectivamente. Además, la reducción del dataset llega a ser sólo el 13,87% y 8,92% del tamaño original. En ambos casos (1NN y C4.5) a pesar de que se configura una perdida alta, el resultado promedio entregado por el algoritmo sobre todos los dataset no alcanza a llegar a la mitad del valor configurado.

		%Error Promedio		Tamaño Promedio		Tiempo Promedio		Loss error	
Dataset	**Error dataset completo**	**Loss error 5%**	**Loss error 10%**	**5%**	**10%**	**5%**	**10%**	**5%**	**10%**
banknote	1,603	**4,759**	10,329	7,15%	**2,16%**	1,256	**1,068**	**3,155**	8,725
blood	22,193	23,797	23,797	0,19%	**0,13%**	**0,361**	0,416	1,604	1,604
car	7,639	**11,238**	15,532	29,52%	**14,40%**	**0,520**	0,728	**3,599**	7,893
climate	6,111	8,530	**8,519**	0,61%	**0,19%**	1,057	**0,855**	2,419	**2,407**
contraceptive	50,849	**53,463**	57,359	1,46%	**0,12%**	1,852	**1,785**	2,615	**6,511**
dermatology	6,557	**7,130**	11,678	15,55%	**10,65%**	**0,312**	0,369	**0,573**	5,121
diabetes	26,172	**26,453**	34,896	0,95%	**0,13%**	0,927	**0,803**	**0,281**	8,724
ecoli	17,560	**19,374**	23,416	8,92%	**6,00%**	0,261	**0,209**	**1,815**	5,856
fertility	14,000	12,000	12,000	1,00%	1,00%	0,093	**0,091**	-2,000	-2,000
glass	33,178	**34,579**	38,197	13,01%	**9,60%**	0,302	**0,186**	**1,402**	5,020
haberman	24,510	26,407	26,407	0,66%	0,66%	**0,149**	0,170	1,898	1,898
ionosphere	12,536	**14,374**	17,572	4,02%	**2,81%**	0,960	**0,761**	**1,838**	5,036
iris	4,667	**5,548**	6,839	9,57%	**8,19%**	0,069	**0,059**	**0,882**	2,172
leaf	40,000	**38,567**	39,042	31,43%	**30,90%**	**0,787**	0,918	**-1,433**	-0,958
libras	30,556	**24,014**	36,228	48,13%	**31,24%**	**2,874**	3,007	**-6,541**	5,672
planning	28,571	**28,554**	28,571	3,85%	**1,13%**	0,199	**0,169**	**-0,018**	0,000
qsarbiodegradation	17,346	**19,954**	24,143	4,57%	**1,73%**	**4,588**	5,186	**2,608**	6,797
seeds	11,905	**13,164**	15,008	4,95%	**3,87%**	**0,109**	0,130	**1,260**	3,103
segment	3,290	**7,033**	10,765	11,16%	**4,20%**	**9,996**	10,036	**3,742**	7,475
sonar	31,731	**28,691**	32,987	4,37%	**2,88%**	0,625	**0,619**	**-3,040**	1,256
soybean	7,321	**9,838**	13,111	33,63%	**24,15%**	0,906	**0,753**	**2,517**	5,790
spectf	12,321	**14,364**	19,549	26,16%	**12,97%**	0,706	**1,650**	**2,043**	7,228
vowel	18,788	**14,972**	26,061	52,76%	**31,92%**	2,750	**2,408**	**-3,816**	7,273
wine	6,742	**9,079**	11,743	19,34%	**12,07%**	0,146	**0,128**	**2,338**	5,002
yeast	44,609	**46,915**	51,259	3,78%	**1,65%**	**3,373**	3,709	**2,306**	6,649
zoo	7,921	**8,144**	12,807	23,95%	**17,06%**	**0,083**	0,144	**0,224**	4,887
Promedio:	18,795	**19,652**	23,377	13,87%	**8,92%**	**1,356**	1,398	**0,857**	4,582

Tabla 18. Comparación perdida error 5% y 10%, fuerza 2 - C4.5

En la **Tabla 19** y la **Tabla 20**, se muestran los resumes de los datos obtenidos ejecutando 31 veces ISCA sobre cada dataset, en este caso utilizando CA de fuerza 4, y los algoritmos de clasificación 1NN y C4.5. Se logra evidenciar que configurando el algoritmo con una perdida máxima de error del 5% y del 10%, se pierde calidad de 0,002 y 2,834 respectivamente con 1NN y la reducción del dataset llega a 9,48% y 5,54% del dataset original, que es una reducción muy significativa.

Para el clasificador C4.5, el promedio de pérdida de calidad para las configuraciones 5% y 10%, son de 0,242 y 4,338 respectivamente. Además, la reducción del dataset llega a ser en promedio del 13,50% y 7,91% del tamaño original. En ambos casos (1NN y C4.5) a pesar de que se configura una perdida alta, el resultado promedio entregado por el algoritmo sobre todos los dataset no alcanza a llegar a la mitad del valor configurado.

Para un mejor entendimiento de los datos, los promedios se resumen en la **Tabla 21**, en donde se puede evidenciar que la mejor configuración se logra con la Perdida máxima del 5% de error, ya que con esta se obtiene en promedio una Perdida del 0,413 en la calidad de los datos. Además, se puede observar que la menor pérdida de calidad para los dos clasificadores se obtiene con fuerza 4. En relación con la reducción, es claro que con una mayor perdida se puede reducir más el tamaño del dataset, pero con Loss error de 5% se obtiene un dataset reducido al 11,695% (0,117) del tamaño original, y en este caso el mejor resultado también se obtiene con fuerza 4, por un estrecho margen.

		Error Promedio		Tamaño Promedio		Tiempo Promedio		Pérdida promedio	
Dataset	**Error original**	**Loss error 5%**	**Loss error 10%**	**Loss error 5%**	**Loss error 10%**	**Loss error 5%**	**Loss error 10%**	**Loss error 5%**	**Loss error 10%**
banknote	0,146	2,490	7,595	1,56%	0,61%	5,508	19,855	2,344	6,757
blood	28,877	23,797	23,797	0,13%	0,13%	1,961	9,494	-5,080	-5,080
car	6,308	9,261	14,742	27,68%	12,65%	25,809	39,931	2,953	8,249
climate	11,481	8,519	8,519	0,19%	0,19%	3,217	11,378	-2,963	-2,963
contraceptive	55,872	57,298	57,298	0,07%	0,07%	8,018	39,332	1,426	1,426
dermatology	5,738	8,355	11,593	7,14%	6,03%	3,126	9,218	2,618	5,936
diabetes	29,427	28,121	34,896	0,49%	0,13%	2,679	13,065	-1,306	5,469
ecoli	19,048	19,902	22,991	1,96%	2,19%	0,669	2,032	0,854	2,679
fertility	17,000	12,000	12,000	1%	1%	0,165	0,450	-5,000	-5,000
glass	29,907	31,504	37,972	10,87%	5,24%	0,530	1,344	1,598	6,372
haberman	32,353	26,471	26,471	0,33%	0,33%	0,350	1,486	-5,882	-5,882
ionosphere	13,105	14,806	15,941	2,23%	1,57%	2,294	7,032	1,700	2,512
iris	4,667	5,570	8,779	3,08%	3,32%	0,258	0,587	0,903	2,182
leaf	97,059	95,436	95,294	0,29%	0,29%	0,832	3,266	-1,622	-1,765
libras	13,333	10,054	20,051	43,70%	24,55%	8,895	14,391	-3,280	6,717
planning	36,813	28,571	28,571	0,55%	0,55%	0,355	1,321	-8,242	-8,242
qsarbiodegradation	15,924	18,383	19,767	0,79%	0,47%	2,611	62,803	2,239	3,843
seeds	5,714	5,813	11,039	3,98%	1,47%	0,120	1,442	2,765	5,325
segment	2,597	14,252	10,866	9,85%	3,93%	0,613	97,027	3,229	8,268
sonar	12,500	10,128	15,721	19,84%	17,50%	6,621	4,370	1,752	3,221
soybean	8,638	15,815	13,704	18,28%	11,58%	2,043	20,101	1,742	5,066
spectf	13,754	0,701	22,092	23,67%	7,16%	1,906	9,440	2,061	8,338
vowel	0,606	6,089	8,246	53,96%	31,68%	1,413	14,001	0,094	7,640
wine	5,056	49,465	8,427	2,65%	1,97%	4,325	1,253	1,033	3,371
yeast	47,035	5,653	52,520	0,85%	0,38%	0,415	36,949	2,430	5,485
zoo	3,960	8,479	7,723	11,40%	9,11%	1,413	0,576	1,693	3,762
Promedio:	19,882	**19,882**	22,947	9,48%	**5,54%**	**3,313**	16,236	**0,002**	2,834

Tabla 19. Comparación perdida error 5% y 10%, fuerza 4 - 1NN

		Error Promedio		Tamaño Promedio		Tiempo Promedio		Pérdida promedio	
dDataset	**Error original**	**Loss error 5%**	**Loss error 10%**	**Loss error 5%**	**Loss error 10%**	**Loss error 5%**	**Loss error 10%**	**Loss error 5%**	**Loss error 10%**
banknote	1,603	4,778	9,561	4,77%	1,46%	6,639	5,187	3,174	7,958
blood	22,193	23,797	23,797	0,13%	0,13%	4,741	2,475	1,604	1,604
car	7,639	10,788	14,936	28,60%	13,38%	5,354	4,623	3,149	7,297
climate	6,111	8,530	8,519	0,45%	0,19%	4,820	3,860	2,419	2,407
contraceptive	50,849	53,258	57,298	0,67%	0,07%	11,424	9,902	2,409	6,449
dermatology	6,557	5,658	11,078	15,24%	9,86%	3,325	2,067	-0,899	4,521
diabetes	26,172	26,168	34,896	1,34%	0,13%	5,873	4,504	-0,004	8,724
ecoli	17,560	19,355	24,351	8,35%	3,54%	2,094	1,231	1,795	6,791
fertility	14,000	12,000	12,000	1%	1%	1,247	0,329	-2,000	-2,000
glass	33,178	33,765	39,766	12,54%	4,72%	1,804	1,243	0,588	6,589
haberman	24,510	26,471	26,471	0,33%	0,33%	1,021	0,519	1,961	1,961
ionosphere	12,536	15,504	17,068	3,30%	1,94%	6,026	4,710	2,968	4,533
iris	4,667	5,118	10,848	9,91%	5,15%	0,443	0,515	0,452	6,182
leaf	40,000	40,598	40,374	25,39%	25,05%	6,070	5,066	0,598	0,374
libras	30,556	17,760	35,152	54,48%	30,86%	26,263	24,880	-12,796	4,596
planning	28,571	28,571	28,571	1,93%	0,55%	0,991	0,752	0,000	0,000
qsarbiodegradation	17,346	19,486	23,817	3,19%	1,29%	27,404	26,108	2,156	6,471
seeds	11,905	12,903	13,117	5,82%	3,94%	1,111	0,994	0,998	1,212
segment	3,290	6,294	7,766	9,55%	7,46%	45,322	70,571	2,991	4,476
sonar	31,731	28,148	31,827	2,82%	2,02%	3,717	3,417	-3,583	0,096
soybean	7,321	6,622	14,362	42,82%	21,83%	5,010	5,426	-0,699	7,041
spectf	12,321	13,014	18,739	23,11%	12,35%	4,766	4,914	0,693	6,418
vowel	18,788	13,535	24,545	51,91%	30,95%	13,098	15,543	-5,253	5,758
wine	6,742	7,340	12,416	19,81%	7,19%	1,357	0,857	0,598	5,674
yeast	44,609	47,428	50,000	2,25%	1,35%	17,629	27,097	2,819	5,391
zoo	7,921	8,080	10,198	21,34%	18,81%	0,696	0,509	0,160	2,277
Promedio:	18,795	**19,037**	23,134	13,50%	**7,91%**	**8,009**	8,742	**0,242**	4,338

Tabla 20. Comparación perdida error 5% y 10%, fuerza 4 - C4.5

Algoritmo de clasificación	Fuerza	Error original	Error Promedio		Tamaño Promedio		Tiempo Promedio		Pérdida promedio	
			Loss error 5%	Loss error 10%	Loss error 5%	Loss error 10%	Loss error 5%	Loss error 10%	Loss error 5%	Loss error 10%
1NN	2	19,882	20,431	23,062	9,93%	6,51%	1,466	1,501	0,550	3,181
C4.5	2	18,795	19,652	23,377	13,87%	8,92%	1,356	1,398	0,857	4,582
1NN	4	19,882	19,882	22,947	9,48%	5,54%	3,313	16,236	0,002	2,834
C4.5	4	18,795	19,037	23,134	13,50%	7,91%	8,009	8,742	0,242	4,338
Promedio:		19,338	**19,751**	23,130	11,695%	**7,22%**	**3,536**	6,969	**0,413**	3,734

Tabla 21. Comparación perdida error 5% y 10%

5.3 Comparación de los resultados frente al estado del arte

De acuerdo con el análisis realizado en la **sección 4.2.3** y la sección previa se logra determinar que las mejores soluciones al aplicar el algoritmo propuesto se obtienen al utilizar ISCA con un CA de fuerza 4. Por ello en esta sección se realiza la comparación de los resultados obtenidos, frente a los principales algoritmos del estado del arte de reducción de instancias.

Por otro lado, previamente en la **sección 3.3** se realizó un resumen de los algoritmos del estado del arte en selección de instancias. En este apartado se decidió seleccionar un algoritmo de cada grupo en la gráfica, esto con el fin de obtener una visión general del resultado de la propuesta frente a dichos algoritmos.

La **Ilustración 31** muestran los grupos más representativos de los algoritmos del estado del arte. El eje horizontal representa el tamaño del dataset (100% = el conjunto de entrenamiento completo). El eje vertical corresponde a los cambios de precisión en el conjunto de pruebas para un algoritmo de selección de instancia. El grupo 1, son algoritmos que lograron reducir el dataset a un conjunto muy pequeño, y el porcentaje de error es muy similar al obtenido por el clasificador kNN. El grupo 2, son algoritmos que logran reducir el dataset a un porcentaje muy pequeño, pero el error es mayor (4% aproximadamente). El grupo 3, está compuesto por el algoritmo CNN, que es el algoritmo que más calidad pierde. Y el grupo 4, está compuesto por algoritmos que mejoran la calidad de los datos, pero no logran reducir significativamente el tamaño del dataset.

Para realizar la comparación del algoritmo propuesto, se seleccionó un algoritmo de cada grupo, a saber:

- Grupo 1: Random Mutation Hill Climbing (RMHC)
- Grupo 2: Encoding Length Heuristic (ELH)
- Grupo 3: Condensed Nearest Neighbour (CNN)
- Grupo 4: Edited Nearest Neighbour (ENN)

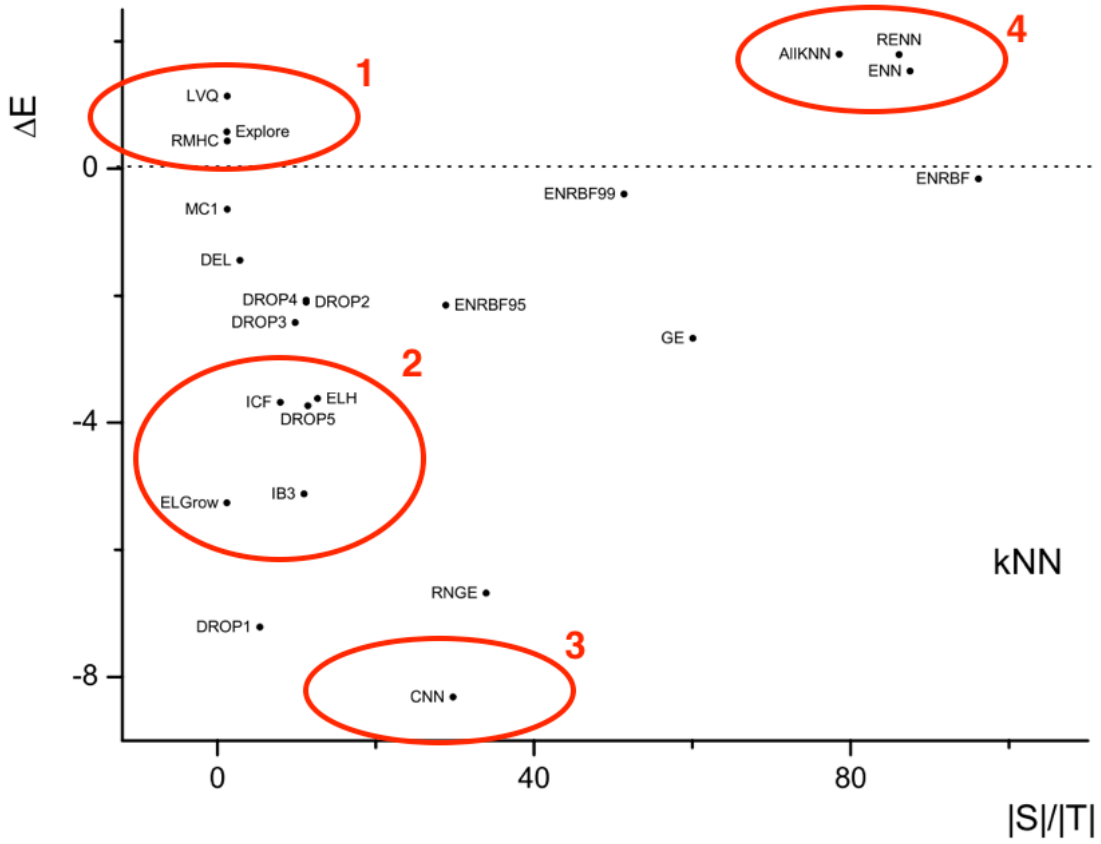


Ilustración 31. Selección de los algoritmos del estado del arte. Adaptada de [85]

De acuerdo con los resultados presentados en la sección 4, en donde se comparó la efectividad de los CA de fuerza 2, 3, 4 y 5 usando ISCA versión 1, se pudo afirmar, que los CA de fuerza 4 infieren los mejores resultados (en promedio de 2,5%) de calidad (exactitud) cuando se ejecuta el clasificador 1NN o C4.5 con el dataset reducido con fuerza 4 en relación con el dataset completo. Además, que el tamaño del dataset reducido llega a ser en promedio un 39,5% del dataset original. En cuanto al valor del umbral de perdida de error, se compararon los resultados de 5% y 10%, como se muestra en la **Tabla 21,** de ellos se seleccionó el umbral de 5%, ya que con este se obtiene en promedio una Perdida del 0,413 en la calidad de los datos, frente a 3,734 (logrado con el 10%). Además, en relación con la reducción, con 'Loss error' de 5% y 10% se obtiene un dataset reducido al 11,695% y 7,22%, respectivamente del tamaño original. Siendo un margen pequeño, y con CA f4 la perdida de error es mucho más pequeña frente a f5, se concluye que la mejor configuración para el algoritmo ISCA es CA de fuerza 4 y 'Loss error' 5%. Esta configuración, mejora significativamente el desempeño del algoritmo, reduciendo el número de instancias, y obteniendo una perdida máxima de error hasta de 5 (en comparación con el error obtenido con el dataset completo). Esta configuración es la utilizada en esta sección del documento, para realizar la comparación de los resultados obtenidos frente a los algoritmos del estado del arte.

Los algoritmos CNN, ELH, ENN, RMHC, ISCA se ejecutaron 31 veces sobre cada uno de los 26 datasets, con el fin de poder identificar el desempeño promedio de estos y comparar a ISCA frente a los algoritmos de selección de instancias del estado del arte. Los resultados se resumen en la **Tabla 22** y la **Tabla 23**, en donde se marcan en negrilla los menores (mejores) resultados.

	1NN					C4.5				
	CNN	**ELH**	**ENN**	**RMHC**	**ISCA**	**CNN**	**ELH**	**ENN**	**RMHC**	**ISCA**
Banknote	2,791	3,880	**0,070**	0,650	2,490	17,391	18,804	**0,440**	6,798	4,778
Blood	**16,141**	28,971	18,048	24,361	23,797	26,388	31,892	**19,027**	23,622	23,797
Car	2,662	**0,060**	12,861	21,554	9,261	9,997	10,379	**9,447**	20,259	10,788
Climate	9,935	13,470	**8,330**	10,250	8,519	**5,084**	6,828	8,520	9,983	8,530
Contraceptive	**18,360**	51,344	33,340	52,763	57,298	**35,675**	51,442	38,134	50,037	53,258
Dermatology	**2,133**	4,019	2,337	11,917	8,355	**3,393**	5,394	3,774	24,503	5,658
Diabetes	**11,728**	15,839	17,286	27,638	28,121	22,725	27,798	**18,880**	27,696	26,168
Ecoli	**1,038**	2,181	10,626	11,914	19,902	15,313	16,973	**9,936**	21,611	19,355
fertility	**3,000**	5,774	10,000	12,806	12,000	13,387	18,871	**12,000**	12,226	**12,000**
glass	**9,150**	11,984	20,923	32,213	31,504	23,245	26,289	**22,745**	48,146	33,765
haberman	**16,909**	30,844	16,066	32,183	26,471	26,079	34,334	**24,480**	26,607	26,471
ionosphere	**0,119**	1,073	12,540	8,243	14,806	19,318	33,755	**8,550**	21,808	15,504
iris	**2,796**	5,700	4,000	3,958	5,570	38,557	40,086	**2,670**	13,592	5,118
leaf	**8,881**	10,246	77,334	68,273	95,436	**13,397**	14,421	76,243	84,279	40,598
libras	**0,905**	1,774	16,039	31,084	10,054	35,277	39,687	**17,241**	63,782	17,760
planning	**2,696**	5,742	22,743	27,277	28,571	**28,570**	**28,570**	**28,570**	32,613	28,571
qsarbiodegradation	**7,167**	10,016	8,719	18,383	18,383	11,903	14,969	**11,370**	19,844	19,486
seeds	**5,792**	7,818	6,020	8,418	5,813	9,970	11,766	**7,620**	17,005	12,903
segment	4,832	6,330	**1,945**	7,942	14,252	10,164	14,709	**1,995**	9,332	6,294
sonar	**2,962**	4,576	14,033	19,108	10,128	22,519	24,721	**11,540**	33,917	28,148
soybean	**1,165**	11,075	18,098	15,497	15,815	**5,809**	26,312	23,945	28,295	6,622
spectf	5,351	9,798	17,714	20,612	**0,701**	**11,443**	16,850	12,890	26,787	13,014
vowel	4,201	6,862	**1,065**	49,378	6,089	36,485	37,876	**4,340**	51,312	13,535
wine	2,502	3,298	**2,196**	8,155	49,465	**5,599**	5,873	8,265	22,889	7,340
yeast	**2,470**	4,708	32,233	44,131	5,653	**26,602**	29,721	33,309	46,342	47,428
zoo	**3,545**	4,279	3,736	5,812	8,479	32,798	36,151	**4,950**	33,788	8,080
Promedio:	**5,740**	10,064	14,935	22,097	19,882	19,503	24,018	**16,188**	29,887	19,037

Tabla 22. Comparación algoritmos - Promedio % error

	1NN					C4.5				
	CNN	**ELH**	**ENN**	**RMHC**	**ISCA**	**CNN**	**ELH**	**ENN**	**RMHC**	**ISCA**
banknote	0,018	**0,012**	0,999	0,100	0,016	0,018	**0,012**	0,999	0,100	0,048
Blood	0,481	0,020	0,728	0,098	**0,001**	0,481	0,020	0,728	0,098	**0,001**
Car	0,291	0,269	0,832	**0,099**	0,277	0,291	0,269	0,832	**0,099**	0,286
climate	0,338	0,266	0,898	0,099	**0,002**	0,338	0,266	0,898	0,099	**0,005**
contraceptive	0,675	0,065	0,490	0,099	**0,001**	0,675	0,065	0,490	0,099	**0,007**
dermatology	0,422	0,273	0,857	0,098	**0,071**	0,422	0,273	0,857	**0,098**	0,152
diabetes	0,511	0,425	0,694	0,098	**0,005**	0,511	0,425	0,694	0,098	**0,013**
Ecoli	0,366	0,296	0,846	0,097	**0,020**	0,366	0,296	0,846	0,097	**0,084**
fertility	0,382	0,275	0,874	0,100	**0,010**	0,382	0,275	0,874	0,100	**0,010**
glass	0,457	0,393	0,682	0,098	**0,109**	0,457	0,393	0,682	**0,098**	0,125
haberman	0,495	0,016	0,739	0,097	**0,003**	0,495	0,016	0,739	0,097	**0,003**
ionosphere	0,228	0,143	0,849	0,099	**0,022**	0,228	0,143	0,849	0,099	**0,033**
iris	0,126	0,087	0,960	0,100	**0,031**	0,126	**0,087**	0,960	0,100	0,099
leaf	0,883	0,863	0,180	0,099	**0,003**	0,883	0,863	0,180	**0,099**	0,254
libras	0,356	0,307	0,809	**0,100**	0,437	0,356	0,307	0,809	**0,100**	0,545
planning	0,576	0,483	0,648	0,097	**0,005**	0,576	0,483	0,648	0,097	**0,019**
qsarbiodegradation	0,372	0,297	0,826	0,098	**0,008**	0,372	0,297	0,826	0,098	**0,032**
seeds	0,201	0,163	0,881	0,099	**0,040**	0,201	0,163	0,881	0,099	**0,058**
segment	0,124	**0,099**	0,958	0,100	**0,099**	0,124	0,099	0,958	0,100	**0,095**
sonar	0,344	0,269	0,817	**0,096**	0,198	0,344	0,269	0,817	0,096	**0,028**
soybean	0,575	0,192	0,529	**0,099**	0,183	0,575	0,192	0,529	**0,099**	0,428
spectf	0,339	0,246	0,765	**0,097**	0,237	0,339	0,246	0,765	**0,097**	0,231
vowel	0,201	0,178	0,983	**0,100**	0,540	0,201	0,178	0,983	**0,100**	0,519
wine	0,404	0,343	0,731	0,095	**0,026**	0,404	0,343	0,731	**0,095**	0,198
yeast	0,661	0,594	0,550	0,099	**0,008**	0,661	0,594	0,550	0,099	**0,022**

zoo	0,142	0,114	0,944	**0,099**	0,114	0,142	0,114	0,944	**0,099**	0,213
Promedio:	0,383	0,257	0,772	0,098	**0,095**	0,383	0,257	0,772	**0,098**	0,135

Tabla 23. Comparación algoritmos - Promedio % instancias

A continuación, en la **Tabla 24** y la **Tabla 25** se resumen los datos de las tablas previas, para tener una visión general más completa de la comparación de los algoritmos, basados en el promedio del error, y el promedio del tamaño del dataset.

Clasificador	CNN	ELH	ENN	RMHC	ISCA
1NN	5,740	10,064	14,935	22,097	19,882
J48	19,503	24,018	16,188	29,887	19,037
Promedio:	**12,622**	17,041	15,562	25,992	19,460

Tabla 24. Resumen comparación algoritmos - Promedio % error

En la **Tabla 24**, se evidencia que, el algoritmo que en promedio tiene menor error es CNN (12,622), en segundo lugar, ENN (15,562), seguido de ELH (17,041), posteriormente ISCA (19,460) y por último RMHC (25,992). Así mismo, en la **Tabla 25**, relacionada con la reducción del tamaño del dataset original, se puede ver que el algoritmo ISCA se encuentra en segunda posición (0,115), siendo superado por RMHC.

Clasificador	CNN	ELH	ENN	RMHC	ISCA
1NN	0,383	0,257	0,772	0,098	0,095
J48	0,383	0,257	0,772	0,098	0,135
Promedio:	0,383	0,257	0,772	**0,098**	0,115

Tabla 25. Resumen comparación algoritmos - Promedio % instancias

La comparación realizada con estos 4 algoritmos del estado del arte indica que, el algoritmo ISCA escoge una cantidad de instancias mucho menor (11,5%), en comparación a los algoritmos CNN (38,3%), ELH (25,7%) y ENN (77,2%), por otro lado, RMHC es el algoritmo que en promedio logra reducir más el dataset (9,8%). En cuanto al promedio del % error, ISCA mejora los resultados en comparación con RMHC, situándose ligeramente cerca de ELH, siendo superado por los demás algoritmos. En la **Tabla 24** se puede evidenciar que los resultados de ISCA son muy similares para 1NN (19,882) y J48 (19,037), lo que tiende a inferir que ISCA no es sensible al cambio de clasificador, sin embargo, es necesario realizar más experimentos para llegar a esta conclusión.

La **Ilustración 32** muestra en el eje X el promedio del porcentaje de error obtenido en cada algoritmo, sin importar el clasificador, y en el eje Y, muestra el promedio del porcentaje de instancias con respecto al dataset original, en donde el valor 1, representa el dataset completo, tal y como se aprecia con los puntos etiquetados como C4.5 y 1NN que usaron el dataset completo. En color rojo están definidos los rankings de Pareto siendo el más cercano al origen el Frente de Pareto. Este Frente de Pareto está conformado por los algoritmos CNN, ELH, ISCA versión 2 con fuerza 4 y RMHC. Entre estos algoritmos no se puede establecer que uno le gana al otro, ya que, si en un objetivo gana, por ejemplo, reducción de instancias, en el otro pierde, por ejemplo, porcentaje de error y viceversa. Además, se puede establecer que en el siguiente ranking esta ENN, seguido por C4.5 y en el último lugar 1NN.

En este sentido, el algoritmo propuesto, ISCA versión 2.0 es competitivo con el estado del arte y presenta una solución en el Frente de Pareto que dista de las ya existentes, lo cual puede ser de gran utilidad para el proceso de selección de instancias.

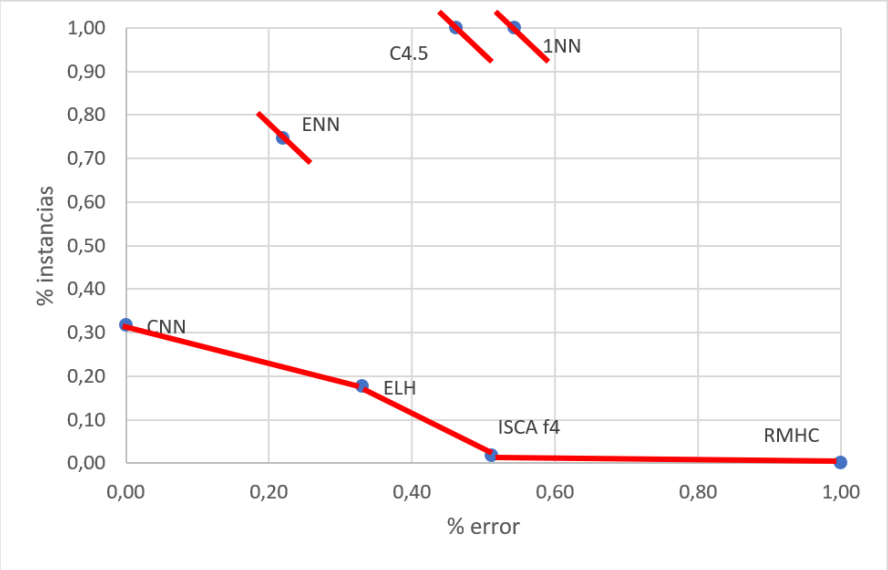


Ilustración 32. Resumen comparación algoritmos

La **Ilustración 33** muestra individualmente los resultados de todos los algoritmos, pero con el clasificador con el que fueron ejecutados. Al realizar los rankings de dominancia se encuentra que en el Frente de Pareto están CNN para reducir y luego 1NN para clasificar (CNN-1NN), ELH-1NN e ISCA tanto con C4.5 como con 1NN. Esto es muy importante, ya que ISCA versión 2 funciona bien con los dos clasificadores, mientras que CNN y ELN sólo obtienen resultados competitivos cuando el clasificador que se usa es 1NN. En el segundo ranking están ENN-1NN, CNN-C4.5 y RMHC-1NN, mostrando que CNN es más consistente que ELH. En el tercer ranking están ENN-C4.5, ELH-C45 y RMHC-C4.5, luego sigue C4.5 con todo el dataset y finalmente 1NN con todo el dataset.

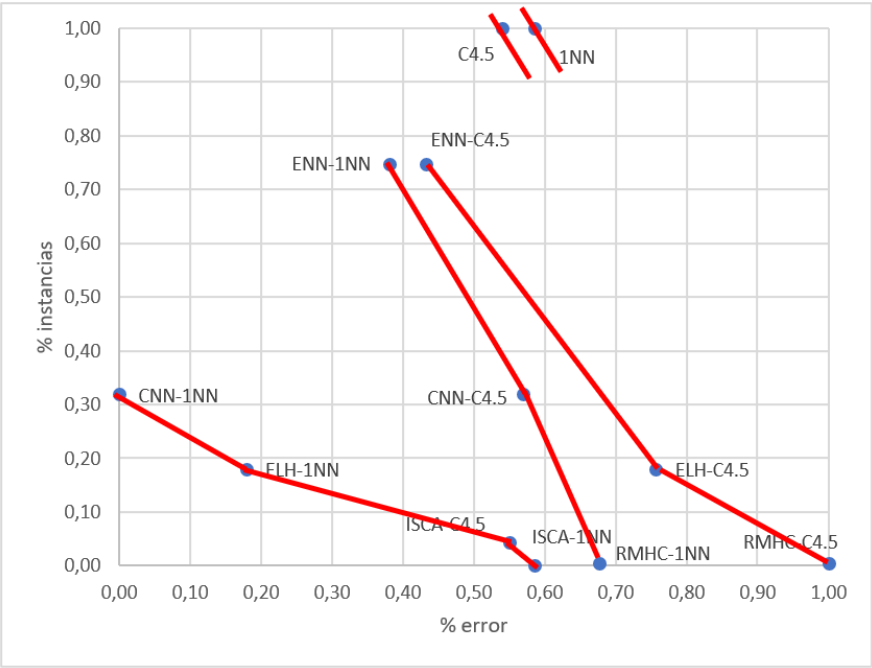


Ilustración 33. Frente de Pareto - comparación de algoritmos

6. CONCLUSIONES Y TRABAJO FUTURO

En una era en que aumenta exponencialmente los volúmenes de datos, es vital mejorar las técnicas de reducción de datos o selección de instancias, con el objetivo de hacer más eficiente el preprocesamiento de estos datos en procesos de KDD. En esta tesis se exploró una nueva forma de reducir o eliminar instancias de conjuntos de datos, con la propuesta de un algoritmo basado en Arreglos de Cobertura (Covering Arrays).

El algoritmo propuesto se denomina ISCA, usa Covering Arrays binarios, los cuales permiten generar diferentes muestras del dataset original y de menor tamaño, de los cuales, se selecciona el mejor muestreo, basado en los pesos asignados para el porcentaje de error de la clasificación (% error), y el porcentaje de instancias que quedan en la muestra en relación con el tamaño del dataset original (% instancias). El algoritmo ISCA en su primera versión se evaluó utilizando CAs de fuerza 2, 3, 4 y 5. La evaluación se realizó con 26 datasets del repositorio de aprendizaje de máquina de la Universidad de California en Irvine, usando dos medidas de comparación: la calidad de la clasificación (menor porcentaje de error en la clasificación) y el porcentaje de reducción de instancias. Los datasets reducidos con ISCA se usaron con dos clasificadores (1NN y C4.5) y la calidad de los modelos se comparó frente al uso del dataset completo (original).

En los experimentos con la primer versión de ISCA se pudo observar que el rendimiento de los clasificadores C4.5 y 1NN, en términos de precisión de clasificación y tasas de reducción de datos, usando el algoritmo propuesto con CAs de fuerza 2, 3, 4, 5 lograron disminuir el error de clasificación de 19,376% a 18,039% (f2), 17,583% (f3), 18,188% (f4) y 16,738% (f5) en los 26 dataset, logrando además reducir el porcentaje de instancias en promedio en un 41,50% (f2), 43,71% (f3), 39,11% (f4) y 45,45% (f5). Por otro lado, en el algoritmo 1NN, se logró disminuir el error de clasificación de 20,316% a 18,346% (f2), 17,995% (f3), 17,292 (f4) y 17,554 (f5), además los datasets se lograron reducir en un 40,13% (f2), 42,54% (f3), 39,12 (f4) y 45,30% (f5).

Sobre los resultados de la experimentación con ISCA versión 1, se realizó la prueba estadística no paramétrica de Friedman y un post hoc de Holm, con el objetivo de determinar cuál fuerza del CA tenía dominancia sobre las demás, la conclusión a la que se llegó, fue que la fuerza 4 es la que mejores resultados reportaba en comparación con las otras. No obstante, en algunos casos los resultados con fuerza 5 fueron mejores, pero estos requieren mayor cantidad de recursos computacionales para su procesamiento.

La segunda versión de ISCA, se puede resumir como un llamado iterativo de la versión 1 hasta que se supere un máximo nivel de tolerancia en la pérdida de calidad de la clasificación (Loss error). Para evaluar el efecto de este nivel de tolerancia se compararon los resultados de ISCA con dos diferentes valores para 'Loss error': 5% y 10%, de ellos se seleccionó el umbral de 5%, ya que con este se obtiene en promedio una Perdida del 0,413 en la calidad de los datos, frente a 3,734 (logrado con el 10%). Además, en relación con la reducción, con 'Loss error' de 5% y 10% se obtiene un dataset reducido al 11,695% y 7,22%, respectivamente del tamaño original. Siendo un margen pequeño, y con CA f4 la perdida de error es mucho más pequeña frente a f5, se concluye que la mejor configuración

para el algoritmo ISCA es CA de fuerza 4 y 'Loss error' 5%. Esta configuración, mejora significativamente el desempeño del algoritmo, reduciendo el número de instancias, y obteniendo una perdida máxima de error hasta de 5 (en comparación del error obtenido con el dataset completo). Dicha configuración fue la utilizada para obtener los resultados, con los cuales se realiza la comparación frente a los algoritmos del estado del arte.

ISCA versión 2 se comparó con 4 algoritmos del estado del arte, a saber: CNN, ELH, ENN, RMHC. El diagrama de Pareto se usó para para comparar las soluciones que reportan al mismo tiempo dos objetivos separados de calidad (% error y %instancias). En dicha comparación se encontró que ISCA es competitivo frente al estado del arte, ya que se ubicó en el frente de Pareto junto con CNN, ELH y RMHC, pero además se encontró que si el análisis se hace incluyendo el clasificador que se usa, ISCA se encuentra presente en ese Frente de Pareto con los dos clasificadores evaluados (C4.5 y 1NN), mientras que los otros (CNN y ELH) solo se reportan en el frente cuando son usados con 1NN.

Estos resultados son muy prometedores, ya que la propuesta que se presenta en este trabajo logra reducir el porcentaje de error y además reducir las instancias de los datasets, y al comparar con los clasificadores C4.5 y 1NN, se logran mejorar los resultados, reduciendo el porcentaje de error y a su vez el tamaño del dataset.

Además, al comparar el algoritmo propuesto, con algoritmos del estado del arte como RMHC, ELH, CNN, ENN, se obtiene que los algoritmos RMHC (9,8%) e ISCA f4 (11,5%), son los mejores algoritmos para reducir instancias, logrando en promedio, reducir los datasets hasta el 10% aproximadamente. En cuanto a reducción de error, los mejores algoritmos son CNN (12,622) y ENN (15,562), seguidos de ELH (17,041), ISCA (19,460) y por último RMHC (25,992). Los resultados de ISCA varían, de acuerdo con las necesidades del usuario, ya que puede configurar los parámetros: 'Loss error', fuerza del CA, y el peso o prioridad en cuanto al %error o al %instancias.

Con este proyecto de investigación se estudia las cualidades de los Covering Array binarios, en un campo que no se habían aplicado, como lo es la reducción o eliminación de instancias. El Algoritmo ISCA identifica el CA apropiado para un determinado dataset, generando múltiples nuevos datasets (muestras), y selecciona la mejor solución, de acuerdo con los parámetros establecidos por el usuario.

El algoritmo propuesto para reducción de instancias proporciona un valor agregado al minero de datos, ya que él puede decidir el peso entre %error y %tamaño, y a su vez, decidir cuanto error estaría dispuesto a perder, durante la ejecución de múltiples iteraciones del algoritmo. Además, exporta todos los resultados, para que el usuario pueda seleccionar, cual solución se adapta más a sus necesidades.

Cabe destacar, que esta propuesta es sensible al orden de los datos, si estos se alteran, los resultados cambian, pero como se muestra en los experimentos la variabilidad de estos no es sustancial.

Como trabajo futuro el Grupo de I+D em Tecnologías de la Información (GTI) espera llevar a cabo las siguientes actividades:

- Incorporar más dataset de diferentes dominios de aplicación y en especial que tengan grandes volúmenes de datos para aplicar el método con particiones del dataset original.

- Comparar la propuesta con otros algoritmos de clasificación disponibles en el estado del arte (Random Forest, Naïve Bayes, entre otros) y otros algoritmos de reducción de instancias (Explore, LVQ, MC1, DEL o C-Pruner).
- Realizar experimentos usando CAs de fuerza 6, 7, 8 y analizar los resultados en conjunto con los presentados en este trabajo.
- Evaluar el uso de arreglos de cobertura incrementales (incremental covering arrays) binarios, los cuales incluyen en un solo objeto diferentes niveles de fuerza.

Página intencionalmente dejada en blanco

7. REFERENCIAS BIBLIOGRÁFICAS

[1] B. Gu, F. Hu, y H. Liu, «Sampling: Knowing Whole from Its Part», *Instance Sel. Constr. Data Min.*, pp. 21-38, 2013.

[2] L. Huan y H. Motoda, «Instance Selection and Construction for Data Mining», *KLUWER Int. Ser. Eng. Comput. Sci.*, p. 448, 2001.

[3] J. R. Cano, F. Herrera, y M. Lozano, «Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study», *IEEE Trans. Evol. Comput.*, vol. 7, n.º 6, pp. 561-575, 2003.

[4] F. Herrera y J. R. Cano, «Técnicas de reducción de datos en KDD. El uso de Algoritmos Evolutivos para la Selección de Instancias», *Actas I Semin. Sobre Sist. Intel.*, pp. 165-181, 2006.

[5] J. A. Timaná Peña, C. A. Cobos Lozada, y J. Torres Jimenez, «Metaheuristic algorithms for building Covering Arrays: A review», *Rev. Fac. Ing.*, vol. 25, n.º 43, pp. 31-45, sep. 2016.

[6] J. Torres Jimenez y J. C. Perez Torres, «A Branch & Bound Algorithm to Derive a Direct Construction for Binary Covering Arrays», *Adv. Artif. Intell. Soft Comput.*, vol. 9413, pp. 158-177, 2015.

[7] J. Han, M. Kamber, y J. Pei, «Data Mining: Concepts and Techniques», *Morgan Kaufman Publ.*, 2012.

[8] P. Gopinadh y K. Palanivel, «Intelligent Report and Predicting Student's Performance using Classification via clustering in e-learning System», 2014.

[9] N. García Pedrajas y A. De Haro García, «Boosting instance selection algorithms», *Knowledge-Based Syst.*, vol. 67, n.º May, pp. 342-360, 2014.

[10] K. P. Zhao, S. G. Zhou, J. H. Guan, y A. Zhou, «C-pruner: an improved instance pruning algorithm», en *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, 2003, vol. 1, pp. 94-99 Vol.1.

[11] C. García, «Selección de Instancias y Atributos en Conjuntos de Datos mediante Algoritmos sobre Grafos», 2012.

[12] T. Ravindra y M.Ñarasimha, «Comparison of genetic algorithm based prototype selection schemes», *Pattern Recognit.*, vol. 34, pp. 523? - 525, 2001.

[13] H. Dorado, C. Cobos, J. Torres Jimenez, D. Jimenez, y M. Mendoza, «A Proposal to Estimate the Variable Importance Measures in Predictive Models Using Results from a Wrapper», en *Mining Intelligence and Knowledge Exploration*, 2018, pp. 369-383.

[14] S. A. Bestoun y K. Z. Zamli, «A review of covering arrays and their application to software testing», *J. Comput. Sci.*, vol. 7, pp. 1375-1385, 2011.

[15] C. E. Serrano, «Modelo Integral para el profesional en ingeniería», *Editor. Univ. del Cauca*, 2005.

[16] O. F.Y, A. J.E.T, A. O, H. J. O, O. O, y A. J, «Supervised Machine Learning Algorithms: Classification and Comparison», *Int. J. Comput. Trends Technol.*, vol. 48, n.º 3, pp. 128-138, 2017.

[17] A. Ahlemeyer Stubbe, S. Coleman, y S. Coleman, «A practical guide to data mining for business and industry», p. 325, 2014.

[18] P. Cichosz, «Data Mining Algorithms: Explained Using R», 2015.

[19] C. Bishop y N. Nasrabadi, «Pattern recognition and machine learning», *Pattern Recognit.*, vol. 4, n.º 4, p. 738, 2006.

[20] M. Kuhn y K. Johnson, *Applied Predictive Modeling [Hardcover]*. 2013.
[21] S. Suthaharan, «Supervised Learning Models», en *Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning*, Boston, MA: Springer US, 2016, pp. 145-181.
[22] I. H. Witten, E. Frank, y M. a. Hall, *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition*, vol. 54, n.º 2. 2011.
[23] D. Pyle., «Data preparation for data mining», *Morgan Kaufmann.*, 1999.
[24] S. Zhang, C. Zhang, y Q. Yang, «Data preparation for data mining. Applied Arti?cial Intelligence», vol. 17, p. 375?381, 2003.
[25] M. Castejón, J. B. Ordieres, F. J. Martínez, y E. P. Vergara, «Outlier detection and data cleaning in multivariate non-normal samples: The PAELLA algorithm», *Data Min. Knowl. Discov.*, n.º 9, p. 171? 187, 2004.
[26] V. Detours, J. E. Dumont, H. Bersini, y C. Maenhaut, «Integration and cross-validation of hihg-throughtput gene expression data: comparing heterogeneous data sets», *FEBS Lett.*, pp. 546(1):98-?102, 2003.
[27] E. Schallehn, K. Sattler, y G. Saake, «Efficient similarity based operations for data integration», *Data Knowl. Eng.*, p. (48):361?-387, 2004.
[28] H. Liu y H. Motoda, «On issues of instance selection», *Data Min. Knowl. Discov.*, pp. 6:115-130, 2002.
[29] H. Brighton y C. Mellish, «Advances in instance selection for instance- based learning algorithms», *Data Min. Knowl. Discov.*, pp. 6:153-172, 2002.
[30] B. S. Ahmed y K. Z. Zamli, «A Review of Covering Arrays and Their Application to Software Testing», *J. Comput. Sci.*, vol. 7, n.º 9, pp. 1375-1385, 2011.
[31] G. Tzanakis, L. Moura, D. Panario, y B. Stevens, «Constructing new covering arrays from LFSR sequences over finite fields», *Discrete Math.*, vol. 339, n.º 3, pp. 1158-1171, 2016.
[32] J. Timaná, C. A. Cobos lozada, y J. Torres jimenez, «Algoritmos metaheurísticos para construir Covering Arrays : Revisión», *Rev. Fac. Ing.*, vol. 25, n.º 43, pp. 31-45, 2016.
[33] D. B. Skalak, «Reduction Techniques for Instance Based Learning Algorithms», *Int. Conf. Mach. Learn.*, pp. 293-301, 1994.
[34] L. M. Webb y Y. Wang, «Techniques for Sampling Online Text Based Data Sets», *Big Data*, n.º May 2015, pp. 655-675, 2016.
[35] P. E. N. Lutu, «Database Sampling for Data Mining», *Encycl. Data Warehous. Mining, Second Ed.*, pp. 604-609, 2011.
[36] Mitchell, Melanie, y J. H. Holland, «When will a genetic algorithm outperform hill-climbing?», 1993.
[37] M. Hofmann y R. Klinkenberg, *Data Mining and Knowledge Discovery Series : RAPID MINER Data Mining Use Cases and Business Analytics Applications*. 2014.
[38] M. C. Liu, G. Wang, K. Josien, T. W. Liao, y E. Triantaphyllou, «An Evaluation of Sampling Methods for Data Mining with Fuzzy C-Means», pp. 355-369, 2013.
[39] R. L. Scheaffer, W. Mendenhall, y L. Ott, «Elementary survey sampling», *SERBIULA (sistema Libr. 2.0)*, 2012.
[40] S. K. Thompson, *Sampling Third Edition STEVEN*. A JOHN WILEY & SONS, INC., 2012.
[41] P. Krishnaiah y C. Rao, «HandBook of statistics 6: sampling», *North- Holl.*, 1988.
[42] G. Kalton, «Introduction to survey sampling», *Newbury Park*, 1983.
[43] R. Scheaffer, W. Mendenhall, y L. Ott, «Elementary survey sampling», *Duxbury Press*, 1996.
[44] B. Chen, H. Way, S. S. Francisco, P. Haas, S. Jose, y P. Scheuermann, «A New Two-Phase Sampling Based Algorithm for Discovering Association Rules».
[45] F. Provost, D. Jensen, y T. Oates, «Efficient progressive sampling», *Knowl. Discov.*

Data Min., pp. 23-32, 2004.
[46] M. Grochowski, «Simple Incremental Instance Selection Wrapper for Classification BT - Artificial Intelligence and Soft Computing», 2012, pp. 64-72.
[47] G. Ritter, H. Woodruff, S. Lowry, y T. Isenhour, «An algorithm for a selective nearest neighbor decision rule», *IEEE Trans. Inf. Theory*, vol. 21, n.º 6, pp. 665-669, 1975.
[48] J. C. Riquelme, J. S. Aguilar Ruiz, y M. Toro, «Finding representative patterns with ordered projections», *Pattern Recognit.*, vol. 36, n.º 4, pp. 1009-1018, 2002.
[49] D. L. Wilson, «Asymptotic properties of nearest neighbor rules using edited data», *IEEE Trans. Syst. Man, Cybern.*, vol. SMC–2, pp. 408–421, 1972.
[50] P. E. Hart, «The condensed nearest neighbor rule», *IEEE Trans. Inform. Theory*, vol. IT–14, pp. 515–516, 1968.
[51] V. J. De-Almeida, S. Lobo, y V. Lobo, «Ship Noise Classification», *Lisbon New Univ. Lisbon Coll. Sci. Technol.*, 2002.
[52] S. Kanj, F. Abdallah, T. Denœux, y K. Tout, «Editing training data for multi-label classification with the k-nearest neighbor rule», *Pattern Anal. Appl.*, vol. 19, n.º 1, pp. 145-161, 2016.
[53] G. W. Gates, «The reduced nearest neighbor rule», *IEEE Trans. Inform. Theory*, vol. IT–14, pp. 431–433, 1972.
[54] D. G. Lowe y D. G. Lowe, «Similarity metric learning for a variable-kernel classifier», *Neural Comput.*, vol. 7, n.º 1, pp. 72-85, 2008.
[55] D. Kibbler y D. W. Aha, «Learning representative exemplars of concepts: An initial case of study», *Proc. 4 Int. Work. Mach. Learn.*, pp. 24-30, 1987.
[56] D. W. Aha, D. Kibbler, y M. K. Albert, «Instance based learning algorithms», *Mach. Learn.*, vol. 6, pp. 7–66, 1991.
[57] H. Brighton, C. Mellish, y C. Mellish, «Advances in instance selection for instance-based learning algorithms», *Data Min. Knowl. Discov*, vol. 6, pp. 6:153-172, 2002.
[58] C. E. Broadley, «Addressing the selective superiority problem: Automatic algorithm/model class selection», *Procc. 10th Int. Mach. Learn. Conf., Amherst, MA*, pp. 17–24, 1993.
[59] Bhattacharya R., «Application of Proximity Graphs to Editing Nearest Neighbor Decision Rule», *Int. Symp. Inf. Theory, St. Monica,* 1981.
[60] B. Aronov, M. Dulieu, y F. Hurtado, «Witness Gabriel graphs», *Comput. Geom. Theory Appl.*, vol. 46, n.º 7, pp. 894-908, 2013.
[61] D. R. Wilson y T. R. Martinez, «Reduction Techniques for Instance Based Learning Algorithms», pp. 257-286, 2000.
[62] D. R. Wilson y T. R. Martinez, «Improved Heterogeneous Distance Functions», vol. 6, pp. 1-34, 1997.
[63] D. R. Wilson y T. R. Martinez, «Instance Pruning Techniques», en *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997, pp. 403-411.
[64] C. F. Tsai y C.-W. Chang, «SVOIS: Support Vector Oriented Instance Selection for text classification», *Inf. Syst.*, vol. 38, n.º 8, pp. 1070-1083, nov. 2013.
[65] G. Arnaiz, Á. D. Pastor, J. F. Rodríguez, J. J. García, y C. Osorio, «Instance selection for regression: Adapting DROP», *Neurocomputing*, vol. 201, pp. 66-81, 2016.
[66] A. Agapie y A. H. Wright, «Theoretical analysis of steady state genetic algorithms», *Appl. Math.*, vol. 59, n.º 5, pp. 509-525, 2014.
[67] T. Back, D. Fogel, y Z. Michalewicz, «Handbook of Evolutionary Computation», *London, U.K. Oxford Univ. Press*, 1997.
[68] L. Kuncheva, «Editing for the k-Nearest neighbors rule by a genetic algorithm», *Pattern Recognit. Lett.*, vol. 16, pp. 809–814, 1995.
[69] C. R. Reeves y D. R. Bush, «Using Genetic Algorithms for Training Data Selection in RBF Networks», en *Instance Selection and Construction for Data Mining*, 2001, pp.

339-356.
[70] M. V Luz, E. Barreiro, y E. Yeguas, «GA and CHC . Two Evolutionary Algorithms to Solve the Root Identification Problem in Geometric Constraint Solving», *Comput. Sci. 2004*, pp. 139-146, 2004.
[71] R. Silhavy, R. Senkerik, Z. Kominkova, O. Petr, S. Zdenka, y C. S. O. Conference, *Artificial Intelligence Perspectives in Intelligent Systems*, vol. 464. 2016.
[72] D. E. Goldberg y J. H. Holland, «Genetic Algorithms and Machine Learning», *Mach. Learn.*, vol. 3, n.º 2, pp. 95-99, oct. 1988.
[73] J. H. Holland, «ADAPTATION IN NATURAL An Introductory Analysis with Applications to Biology », p. 183 pages, 1975.
[74] F. Vavak y T. C. Fogarty, «Comparison of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments».
[75] H. Xu, «Comparison of genetic operators on a general genetic algorithm package», *Shanghai Jiao Tong Univ.*, vol. 48, n.º 9, pp. 1-16, 2005.
[76] S. Baluja, «Population based incremental learning: a method for integrating genetic search based function optimization and competitive learning», *Carnegie Mellon Univ.*, pp. 94-163, 1994.
[77] H. Pang, K. Hu, y Z. Hong, «Adaptive PBIL algorithm and its application to solve scheduling problems», *Proc. ISIC 2006*, pp. 784– 789.
[78] H. Xing y R. Qu, «A population based incremental learning for delay constrained network coding resource minimization», *in Proc. EvoAppli- cations*, pp. 51–60, 2011.
[79] H. Xing y R. Qu, «A population based incremental learning for network coding resources minimization», *IEEE Commun. Lett.*, vol. 15, n.º 7, pp. 698-700, 2011.
[80] J. Marín, D. Molina, y F. Herrera, «Modeling dynamics of a real-coded CHC algorithm in terms of dynamical probability distributions», *Soft Comput.*, vol. 16, n.º 2, pp. 331-351, 2012.
[81] L. M, H. F, K. N, y M. D, «Real coded memetic algorithms with crossover hill climbing», *Evol Comput*, vol. 12, n.º 3, pp. 273–302, 2004.
[82] S. Rathee, S. Ratnoo, y J. Ahuja, «Instance Selection Using Multi-objective CHC Evolutionary Algorithm BT - Information and Communication Technology for Competitive Strategies», 2019, pp. 475-484.
[83] R. . Cameron-Jones, «Instance selection by encoding length heuristic with random mutation hill climbing», *Proc. Eighth Aust. Jt. Conf. Artif. Intell.*, pp. 99–106, 1995.
[84] K. el Hindi y M. AL-Akhras, «Smoothing decision boundaries to avoid overfitting in neural network training». 2011.
[85] M. Grochowski y N. Jankowski, «Comparison of Instance Selection Algorithms II. Results and Comments», n.º June, pp. 580-585, 2010.
[86] C. J. Merz y P. . Murphy, «UCI repository of machine learning databases», 1998.
[87] H. Zille, A. Kottenhahn, y S. Mostaghim, «Dynamic Distance Minimization Problems for dynamic multi-objective optimization», *2017 IEEE Congr. Evol. Comput. CEC 2017 - Proc.*, pp. 952-959, 2017.
[88] E. Lopez, J. Daniel, Torres, T. Jose, Rodriguez, y N. Eduardo, Rangel, Valdez, «Strength two covering arrays construction using a SAT representation», *Lect. Notes Comput. Sci.*, vol. 5317 LNAI, pp. 44-53, 2008.
[89] E. Rodriguez Tello y J. Torres Jimenez, «Memetic Algorithms for Constructing Binary Covering Arrays of Strength Three», *Artifical Evol.*, vol. 5975, pp. 86-97, 2010.

8. ANEXOS

Los anexos fueron almacenados en forma digital y se encuentran en el CD incluido en la monografía. Se encuentra estructurado de la siguiente forma:

- **Anexo 1**: Carpeta Anexo 1, Código fuente del algoritmo ISCA, en sus dos versiones, además de un corto manual de funcionamiento.
- **Anexo 2**: Artículo "Algoritmo para la selección de instancias en problemas de clasificación basado en arreglos de cobertura" publicado en evento cuyas memorias se encuentran en proceso de publicación en la revista RISTI.
- **Anexo 5**: Carpeta Resultados_ISCA con los resultados de la comparación del algoritmo propuesto y los algoritmos del estado del arte, además de los dataset utilizados.