ARREGLOS DE CUBRIMIENTO PARA SOPORTAR EL PROCESO DE SELECCIÓN DE CARACTERÍSTICAS EN EL CLASIFICADOR RANDOM FOREST

Juan Sebastián Vivas Méndez

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

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de I+D en Tecnologías de la Información (GTI)
Línea Investigación en Gestión de la Información -> Minería de Datos
Popayán, abril de 2018

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE SIGLAS

RF	Random Forest (Bosques aleatorios)
CA	Covering Arrays (Arreglos de Cobertura)
TCA	Towers of Covering Arrays (Torres de Arreglos de Cobertura)
RFCA	Random Forest que integra Covering Arrays
RFTCA	Random Forest que integra Towers of Covering Arrays
UCI	Universidad de California en Irvine

LISTA DE ANEXOS

Anexo 1: Implementación de un paquete Weka en el que se incluye algoritmo Random Forest que integra los Covering Arrays (RFCA) y el algoritmo Random Forest que integra a las torres de Covering Arrays(RFTCA), con la documentación y código fuente respectivo.

Anexo 2: Artículo “Covering arrays to support the process of feature selection in the Random Forest classifier”.

Anexo 3: Propuesta y análisis de resultados de Random Forest basado en Torres de arreglos de Cobertura(RFTCA)

Anexo 4: Resultados obtenidos en la experimentación para exactitud en RFCA y RFTCA.

Anexo 5: Resultados obtenidos en la experimentación para medida F en RFCA y RFTCA.

Anexo 6: Resultados obtenidos en la experimentación para tiempo de ejecución en RFCA y RFTCA.

Anexo 7: Vista Minable de cada fuerza para los resultados en términos de exactitud en la propuesta RFCA.

RESUMEN

El algoritmo Random Forest (RF) es actualmente uno de los más usados en minería de datos para resolver problemas de clasificación. La literatura señala dos limitaciones importantes de RF: 1) la cantidad de tiempo que toma la fijación manual de los híper parámetros (número de árboles en el bosque y numero de características) y la falta de un proceso más apropiado de selección de características que la sencilla selección aleatoria. En este trabajo se proponen y evalúan diversas variaciones del algoritmo RF en los que se integran arreglos de cubrimiento (Covering Arrays) (CA) de fuerza dos a siete, y Torres de arreglos de cubrimiento (Towers of Covering Arrays) (TCA) binarios de fuerza dos a seis como mecanismo de selección de características, donde, el número de filas del CA o del TCA permite definir el número de árboles a generar. Cada renglón del CA o el TCA define las características que utiliza cada subconjunto bootstrap (muestreo aleatorio con reemplazo del conjunto de datos) en la creación de cada árbol base. Para comparar el desempeño de los algoritmos propuestos, Random Forest con Covering Arrays (RFCA) y Random Forest con Torres de Covering Arrays (RFTCA), se definieron 33 conjuntos de datos (datasets) que representan problemas de clasificación con diferentes niveles de complejidad obtenidos del repositorio de la Universidad de California en Irvine (UCI). Los algoritmos fueron evaluados usando validación cruzada(cross-validation) de 10 folders y se evidencia que RFCA y RFTCA logran obtener el mejor desempeño basado en los resultados de los test estadísticos no paramétricos de Friedman y Wilcoxon. Con los resultados de exactitud del algoritmo RFCA con respecto al algoritmo de referencia originalmente propuesto por Breiman se obtuvieron seis modelos de árboles de decisión (uno para cada fuerza) que facilitaron la identificación de los tipos de conjunto de datos donde la propuesta RFCA tiene una mayor probabilidad de obtener mejores resultados. Los resultados muestran que en general RFCA obtiene mejores resultados que el estado del arte con una mejora entre 0.5% y 2%.

CAPÍTULO 1

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

Hoy día, gracias a los avances tecnológicos tanto en recolección como almacenamiento de datos, la humanidad está viviendo en la era de la información en la cual los volúmenes de datos superan nuestra capacidad de trabajar eficazmente con los mismos. Por esto, surge la necesidad de establecer relaciones entre las múltiples características de los datos y con ello extraer la mayor cantidad de conocimiento útil. En este contexto, la minería de datos es un proceso que mediante una variedad de métodos de análisis permite descubrir patrones y relaciones desconocidas, inesperadas, interesantes y relevantes que se pueden utilizar en las diferentes áreas de desarrollo humano, haciendo uso de técnicas que permiten resolver problemas de clasificación, estimación, predicción, agrupación por similitud, asociación o análisis de series temporales [1][2][3].

En minería de datos, una gran variedad de los problemas que se resuelven, están relacionados con la tarea de clasificación, y en esta tarea se usan diversas técnicas basadas en estadística, aprendizaje de máquina, entre otras. La clasificación consiste en tomar un elemento y definir a cuál clase pertenece de un conjunto previamente definido de clases, por ejemplo, definir si un solicitante de un préstamo estará en la clase de los que van a pagar o de los que no lo van a hacer [1][2][3].

Entre los principales algoritmos de clasificación están: Naive Bayes, Redes neuronales, k-vecinos más cercanos, Máquinas de Soporte Vectorial (Support Vector Machines), árboles de decisión (Random Forest, C4.5, C5.0, CART) y algoritmos de cobertura (Ripper). La calidad de los resultados del clasificador depende de diversas circunstancias, entre ellas: la naturaleza de los datos (continuos, categóricos, binarios), la cantidad y calidad de los datos, la presencia de características (atributos, variables o columnas) redundantes, correlacionados o inservibles y el ruido en los datos.

En cuanto a los algoritmos de clasificación basados en árboles de decisión se señala en la literatura las siguientes limitaciones:

- C4.5: En [4] y [5] señalan que frecuentemente en la construcción del árbol de decisión se generan nodos innecesarios (generados a partir de características irrelevantes), los cuales son utilizados para la generación de las reglas que componen el árbol, esto trae como consecuencia la disminución en la eficiencia

del clasificador, aumento en la complejidad del modelo, problemas de sobreajuste y la demanda de una mayor cantidad de datos para cubrir las deficiencias en el modelo.

- CART: La principal limitación del algoritmo CART es su sensibilidad a pequeñas perturbaciones del conjunto de aprendizaje, es decir, pequeñas alteraciones sobre los datos de entrenamiento modifican ampliamente el árbol resultante [6][7].
- C5.0: Conforme a [8] la estrategia utilizada para dividir características categóricas puede conducir a que se forme un árbol con muchos nodos hoja que contienen pocos registros.

Teniendo en cuenta las limitaciones presentes en algunos de los algoritmos de clasificación basados en árboles de decisión es preciso destacar a Random Forest (RF) [9][7] por su robustez, poca sensibilidad al ruido y su poco riesgo al sobre entrenamiento (overfitting). Random Forest en resumen, consiste en un ensamble de árboles de decisión construidos a partir de subconjuntos de datos distintos muestreados desde el conjunto de datos de entrenamiento (usando el concepto de muestreo con remplazo, el número de instancias de cada subconjunto es aproximadamente 2/3 partes de las instancias del conjunto de entrenamiento) y subconjuntos de características seleccionadas aleatoriamente con respecto a todas las características presentes en el conjunto de entrenamiento. En RF no se aplica poda y cada uno de los árboles generados funciona como un clasificador base. Para clasificar un nuevo dato se toma una decisión con base en la opinión de los árboles previamente construidos en el entrenamiento (voto mayoritario en problemas de clasificación y el valor promedio en problemas de regresión) [9][7].

En [9] los autores de RF definen dos formas de construir el bosque de árboles basándose en enfoques diferentes del proceso de selección de características: La primera forma denotada como Forest-RI emplea en la división de cada nodo, pequeños grupos de características seleccionadas aleatoriamente, el tamaño F de los grupos es fijo y generalmente toma el valor de 1 (utilizar sólo una característica seleccionada aleatoriamente), o el primer entero menor que $\log_2 P + 1$, donde P es el número de características de entrada. La segunda forma denotada como Forest-RC se emplea cuando se tienen pocas características de entrada, y consiste en realizar combinaciones lineales de las características existentes y generar cada árbol mediante las combinaciones de estas características o variables.

En cuanto a las limitaciones de RF, en [10] señalan la cantidad de tiempo que toma la fijación manual de los híper parámetros (número de árboles en el bosque, número de características y profundidad de cada árbol) y la falta de un proceso más apropiado de selección de características que la sencilla selección aleatoria. Conforme a [11] el valor de K (número de características seleccionadas aleatoriamente para cada árbol) se establece de manera arbitraria o empírica, y muchas veces no tiene una justificación teórica o experimental. En la presente

investigación se propone la integración de los arreglos de cubrimiento (covering arrays) y torres de arreglos de cubrimiento (towers of covering arrays) como mecanismo para mejorar el proceso de selección de características, eliminando además la necesidad de fijar y afinar los híper parámetros K y número de árboles.

Un Covering Array (CA) es un objeto matemático que puede aplicarse en cualquier tarea donde se requiera realizar un gran número de pruebas o donde intervengan también un gran número de parámetros que interactúan entre sí, como por ejemplo en las pruebas de software, las pruebas de hardware, el diseño experimental [12] y más recientemente en el clustering [13] y en la definición de un wrapper para selección de características y estimación de la importancia de las variables en modelos de clasificación y regresión [14].

En este trabajo de grado se usaron CAs y TCAs binarios, permitiendo que el número N de filas del CA o TCA defina el menor número de árboles a generar en RF y cada renglón del CA o TCA las características que utiliza cada subconjunto de datos que se usa para crear cada árbol. En este sentido, en un renglón las características que tienen el valor de 1 indican ser seleccionadas para la construcción del árbol y las que tengan 0, el no ser seleccionadas. El valor de t, la fuerza en un CA y TCA es un híper parámetro del algoritmo Random Forest basado en arreglos de cubrimiento (RFCA) y del Random Forest basado en Torres de arreglos de cubrimiento(RFTCA).

Teniendo en cuenta lo anterior, en la propuesta se buscó resolver la siguiente pregunta de investigación ¿Cuál es la mejor manera de integrar los covering arrays como mecanismo de selección de características en Random Forest para obtener un algoritmo que busque una mayor calidad en la clasificación y un menor tiempo de ejecución con respecto a los reportados por el estado del arte?

1.2 APORTES DEL PROYECTO

Desde la perspectiva de la investigación las contribuciones del presente trabajo de grado se centran en generar nuevo conocimiento, dirigido a la definición de algoritmos basados en Random Forest donde el valor de t, la fuerza de un CA o TCA es un nuevo híper parámetro. El número N de filas de un CA o TCA binario define el número de árboles a generar en RF y en el que cada renglón del CA o TCA define las características que utiliza cada subconjunto de datos que se usa para crear cada árbol base. En este sentido, en un renglón las características que tienen el valor de 1, indica que se seleccionan para la construcción de la muestra y las que tienen 0 no son seleccionadas. Además, con el uso del test estadístico no paramétrico de Friedman y una relación de dominancia con el test estadístico no paramétrico de Wilcoxon se estableció un ranking con los algoritmos que integran los Covering Arrays y otro para los que integran las Torres de Covering Arrays, denominando a los mejores RFCA y RFTCA respectivamente. A la fecha, este enfoque no ha sido propuesto ni evaluado en trabajos previos.

Desde la perspectiva de innovación, esta investigación propone un nuevo algoritmo RF, en el cual el uso de CAs Binarios y TCAs binarios suprime la fijación manual y/o empírica del híper parámetro número de árboles y el mecanismo de selección aleatoria de características que constituyen un árbol base. El híper parámetro (número de árboles) aumenta linealmente la exactitud del modelo, es decir, entre más grande sea este valor mejor será la exactitud, no obstante, en cierto valor, dicha exactitud deja de crecer, y las características seleccionadas para la construcción de un árbol base pueden incidir en la exactitud del RF. Con lo cual al soportar la propuesta en el uso de CAs y TCAs se suprime la elección del híper parámetro número de árboles, y se propone un nuevo mecanismo de selección de características.

En cuanto al desarrollo, el algoritmo propuesto esta implementado en java e integrando a la herramienta de minería de datos Weka [2] [15] versión 3.8, siguiendo y respetando la arquitectura y organización de la misma. Con el fin de evaluar la calidad de los resultados y la eficiencia del nuevo clasificador se utilizaron conjuntos de datos reconocidos por el estado del arte y disponibles en el repositorio de la UCI (University of California).

1.3 OBJETIVOS

A continuación, se describen los objetivos del proyecto, conforme fueron aprobados por el Consejo de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca en el documento del anteproyecto.

1.3.1 Objetivo General

Proponer una estrategia para la selección de características en Random Forest basado en el uso de Arreglos de cubrimiento buscando mejorar la calidad en la clasificación, facilitar la afinación del clasificador y disminuir el número de árboles requeridos por el mismo.

1.3.2 Objetivos Específicos

- Definir un nuevo clasificador basado en Random Forest modificando el proceso de selección de características a través de arreglos de cubrimiento uniformes y torres de arreglos de cubrimiento (también conocidos como arreglos de cubrimiento incrementales) con alfabeto binario y diferentes niveles de fuerza.
- Implementar la nueva versión del clasificador e incorporarlo en la herramienta de minería de datos Weka versión 3.8, siguiendo y respetando la arquitectura y organización de la misma.
- Evaluar la calidad de los resultados y la eficiencia de la nueva versión del clasificador y compararlo con resultados de otros métodos en el estado del arte, usando el error de clasificación, la precisión, el recuerdo, la medida f y el tiempo de ejecución sobre los datasets de Glass, Ionosphera, Sonar, Vehicle, Wdbc,

Wine y Zoo, reconocidos por el estado del arte y disponibles en el repositorio de la UCI (University of California Irvine).

1.4 RESULTADOS OBTENIDOS

A continuación, se resumen los principales resultados del presente trabajo de grado:

- Monografía del trabajo de grado, corresponde al presente documento que contiene el estado del arte sobre el problema. Luego se presentan los algoritmos Random Forest que integran arreglos de cubrimiento (Covering Arrays) (RFCA) binarios y Torres de arreglos de cubrimiento (Towers of Covering Arrays) (RFTCA) (ver **Anexo 3**) binarios como mecanismo de selección de características. Después se presenta el análisis de los resultados obtenidos por los algoritmos propuestos y su comparación con los algoritmos del estado del arte. Finalmente, se muestran las conclusiones y el trabajo futuro que el grupo de investigación espera desarrollar en el corto plazo.
- Aplicación software con la implementación de los algoritmos RFCA y RFTCA en java e integrados a la herramienta de minería de datos Weka versión 3.8, junto a la documentación y código fuente respectivo.
- Artículo “Covering arrays to support the process of feature selection in the Random Forest classifier” que resume parte de la investigación y los resultados obtenidos, el cual se encuentra en proceso de evaluación en LOD 2018 - The Fourth International Conference on Machine Learning, Optimization, and Data Science, September 13-16, 2018 – Volterra, Tuscany, Italy, evento que genera memorias en la revista Lecture Notes in Computer Science clasificada A2 por el PUBLINDEX de COLCIENCIAS en 2017.

1.5 ESTRUCTURA DE LA MONOGRAFIA

A continuación, se describe de manera general el contendido y organización de la presente monografía:

CAPITULO 1: INTRODUCCIÓN: Hace referencia al presente capitulo que introduce el tema de investigación, presenta la pregunta de investigación que origino el trabajo, los aportes al problema, también los objetivos (general y específicos) definidos en el anteproyecto, un breve resumen de los resultados obtenidos y finalmente la organización de la monografía.

CAPITULO 2: CONTEXTO TEÓRICO Y ESTADO DEL ARTE: En el segundo capítulo se resumen el contexto teórico de las temáticas integradas en la investigación (Clasificación, Random Forest, Covering Arrays, Torres de Covering Arrays), luego se describe de manera concisa cada temática, ejemplificando algunas de las mismas. Finalmente se presentan algunos trabajos relacionados con análisis y mejoras teóricas al algoritmo original de RF, junto a mejoras prácticas de su aplicación en diferentes campos.

CAPITULO 3: PROPUESTA: Este capítulo presenta inicialmente una descripción del proceso de investigación realizado en el transcurso de los 9 meses del trabajo de grado, en el resto del capítulo se describe en detalle la propuesta de Random Forest basada en Covering Array(RFCA).

CAPITULO 4: IMPLEMENTACION DE RFCA Y RFTCA EN WEKA: Este capítulo presenta una arquitectura general de los algoritmos: Random Forest, la propuesta RFCA y RFTCA, también se indica de manera general la interacción entre las distintas clases y al final del capítulo se presenta un corto manual de instalación de un paquete weka.

CAPITULO 5: EXPERIMENTACIÓN: En este capítulo se presenta el análisis de los resultados de la experimentación sobre la exactitud con respecto al número de árboles en el Random Forest de Breiman, luego se detalla el proceso realizado para la evaluación del algoritmo propuesto (RFCA) y su comparación con los algoritmos del estado del arte. Finalmente, con la aplicación de minería de datos se realiza un análisis de los resultados obtenidos en la exactitud sobre el algoritmo RFCA.

CAPITULO 6: CONCLUSIONES Y TRABAJOS FUTUROS: En este capítulo se presentan las conclusiones obtenidas al finalizar el trabajo de grado e ideas que el grupo de investigación espera realizar un trabajo futuro.

CAPITULO 7: BIBLIOGRAFIA: Este último capítulo contiene las referencias bibliográficas de los artículos y libros consultados para la realización del proyecto.

CAPÍTULO 2

2 CONTEXTO TEÓRICO Y ESTADO DEL ARTE

2.1 CONTEXTO TEÓRICO

En esta sección se presentan los conceptos más relevantes para el desarrollo de este trabajo de grado, buscando de esta forma dar mayor claridad en la lectura del presente documento. Además, los conceptos aquí presentados van acompañados de referencias bibliográficas para que el lector pueda profundizar y con ello obtener un mayor detalle de los temas.

2.1.1 Clasificación

La clasificación es un proceso cognitivo utilizado para organizar y aplicar el conocimiento sobre el mundo. Es común tanto en la vida cotidiana como en los negocios, donde se clasifican clientes, empleados, transacciones, tiendas, fábricas, dispositivos, documentos o cualquier otro tipo de instancia en un conjunto de clases o categorías significativas previamente definidas.

En este sentido, la construcción de modelos de clasificación realizados mediante el análisis de datos disponibles es una de las tareas centrales de la minería de datos. La tarea de clasificación consiste en asignar instancias de un dominio dado, descrito por un conjunto de características con valores discretos o continuos, a un conjunto de clases, que pueden considerarse valores de una característica objetivo (etiqueta de clase). Las etiquetas de clase correctas generalmente son desconocidas, pero se proporcionan para un subconjunto del dominio (datos de entrenamiento). A partir de esta información es posible crear un modelo de clasificación, que es una representación del conocimiento necesaria para clasificar nuevas instancias del mismo dominio, descritas por el mismo conjunto de características. Esta forma de asignación de la etiqueta de clase es viable para la mayoría de las aplicaciones prácticas de clasificación, donde la clase representa alguna propiedad de instancias clasificadas que es difícil o costosa de determinar, o (más típicamente) que se conoce más tarde de lo necesario [1][2][3].

La aplicación de un modelo de clasificación para asignar etiquetas de clase a las instancias se conoce comúnmente como predicción [16]. Formalmente, un clasificador es un modelo o función MODEL que predice la etiqueta de clase $\hat{y}$ para una entrada x dada (ver **ecuación (1)**) [17].

$$\hat{y} = MODEL(x) \quad (1)$$

Donde $\hat{y} \in \{c_1, c_2, \ldots, c_k\}$, cada c_i es una etiqueta de clase y x el conjunto de características de entrada.

2.1.2 Random Forest

El algoritmo Random Forest (RF) fue desarrollado por Leo Breiman y Adele Cutler en 2001 [9]. Esta compuesto por una colección de M arboles independientes a los que se les pasa una entrada y cada uno emite un voto unitario, luego RF selecciona la clase más popular de todos los votos recibidos (voto mayoritario). Inicialmente en el proceso de construcción de un RF se aplica el método de bootstrapping (muestreo aleatorio con reemplazo) sobre el conjunto de datos de entrenamiento para producir varios subconjuntos de datos distintos [9][18]. Luego cada subconjunto se usa para construir un árbol de decisión.

Cada nodo de un árbol de decisión se construye a partir de un número K de características seleccionadas aleatoriamente con respecto a todas los características presentes en el conjunto de datos [11]. El número de características (K) es fijo y la definición de su valor se describe en la **ecuación (2).** En el RF no se aplica poda y cada uno de los árboles generados funciona como un clasificador Base. Para definir la clase de una instancia, se recibe el voto de cada uno de los clasificadores base y se realiza una ponderación que determina la clase respectiva [7][19].

$$K = \lfloor log_2(P) + 1 \rfloor \quad (2)$$

Donde P es el número de características del conjunto de datos de entrada sin contar la característica objetivo o de clase.

EJEMPLO: Con el fin de ilustrar el comportamiento del algoritmo Random Forest se presenta el siguiente ejemplo, en el cual se utiliza una versión adaptada de uno de los Conjuntos de Datos (Iris) del Repositorio de la UCI (Universidad de California en Irvine) que contiene 3 clases (setosa, versicolor, virginica) de 50 instancias cada una (en total 150 instancias o registros), 4 características (longitud del sépalo, ancho del sépalo, longitud del pétalo y ancho del pétalo), cada clase se refiere a una especie de la planta Iris [20]. En este ejemplo solo se usan 20 instancias en total (ver **Tabla 1**).

En este ejemplo se utiliza el 80% del conjunto de datos para entrenamiento y el 20% restante se usa como conjunto de prueba del RF. Los parámetros se definen así: 1) El número M de Árboles se asignará a tres (3), y 2) El Número de Características (K) según la propuesta de Breiman (ver **ecuación (2)**) [9] queda igual a $\mathrm{K} = \lfloor \log_2(4) + 1 \rfloor = 3$. En este caso M y K coincidieron, pero en la práctica son distintos.

A1	A2	A3	A4	Clase
Longitud Sépalo	**Ancho Sépalo**	**Longitud Pétalo**	**Ancho Pétalo**	**Especie de planta Iris**
Conjunto de Datos de Entrenamiento				
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4.0	1.3	versicolor
6.5	2.8	4.6	1.5	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica
6.3	2.9	5.6	1.8	virginica
6.5	3.0	5.8	2.2	virginica
7.6	3.0	6.6	2.1	virginica
Conjunto de Datos de Prueba				
5.0	3.3	1.4	0.2	setosa
5.7	2.8	4.1	1.3	versicolor
6.2	3.4	5.4	2.3	virginica
5.9	3.0	5.1	1.8	virginica

Tabla 1. Descripción del conjunto de Datos Iris adaptado

En la construcción de un árbol en el Random Forest se realiza el método de bootstrapping en la cual se selecciona una porción del conjunto total de datos de entrenamiento, adicionalmente el árbol se compone del número de características seleccionadas aleatoriamente, que en el ejemplo tiene el valor de tres. (**Figuras 1, 2 y 3**).

Datos para el ARBOL 1				
Longitud Sépalo	**Ancho Sépalo**	**Longitud Pétalo**	**Ancho Pétalo**	**Clase**
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.6	3.1	1.5	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.8	2.1	5.1	1.9	virginica
6.3	2.9	5.6	1.8	virginica
6.5	3.0	5.8	2.2	virginica
7.6	3.0	6.6	2.1	virginica

Longitud Pétalo < 5
  Longitud Sépalo < 5.75: setosa (4 / 0)
  Longitud Sépalo >= 5.75: versicolor (3/ 0)
Longitud Pétalo >= 5: virginica (9 / 0)

Figura 1. Árbol generado para el conjunto de entrenamiento 1

Datos para el ARBOL 2				
Longitud Sépalo	**Ancho Sépalo**	**Longitud Pétalo**	**Ancho Pétalo**	**Clase**
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.5	2.3	4.0	1.3	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica
6.5	3.0	5.8	2.2	virginica
7.6	3.0	6.6	2.1	virginica

Longitud sépalo < 5.65
  Ancho sépalo < 2.65: versicolor (1 / 0)
  Ancho sépalo >= 2.65: setosa (6 / 0)
Longitud sépalo >= 5.65: virginica (9 / 0)

Figura 2. Árbol generado para el conjunto de entrenamiento 2

Datos para el ARBOL 3				
Longitud Sépalo	**Ancho Sépalo**	**Longitud Pétalo**	**Ancho Pétalo**	**Clase**
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	Setosa
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica
6.3	2.9	5.6	1.8	virginica
6.5	3.0	5.8	2.2	virginica
7.6	3.0	6.6	2.1	virginica

Longitud sépalo < 5.45: setosa (5 / 0)
Longitud sépalo >= 5.45: virginica (11 / 0)

Figura 3. Árbol generado para el conjunto de entrenamiento 3

Una vez se crean los árboles, se puede realizar el uso o prueba del clasificador. En este caso se pasan las instancias de prueba a cada uno de los árboles. La etiqueta de clase se asigna en función de un voto mayoritario. Los resultados se presentan en la **Tabla 2**. Donde se puede llegar a la conclusión de que el clasificador obtiene 75% de instancias correctamente clasificadas ya que en 3 de los 4 casos de prueba el valor real y el valor predicho coincidieron.

	Árbol 1	**Árbol 2**	**Árbol 3**	**Clase Real**	**Resultado (Voto mayoritario)**
Instancia de prueba 1	Setosa	setosa	setosa	setosa	**setosa**
Instancia de prueba 2	Setosa	virginica	virginica	versicolor	**virginica**
Instancia de prueba 3	virginica	virginica	virginica	virginica	**virginica**
Instancia de prueba 4	virginica	virginica	virginica	virginica	**virginica**

Tabla 2. Conjunto de Datos de Prueba

2.1.3 Covering Array

Un Covering Array (CA), arreglo de cubrimiento o arreglo de cobertura, es un objeto matemático, que puede describirse como una matriz de $N \times P$ elementos, tal que cada $N \times t$ subarreglo contiene todas las combinaciones de los v^t símbolos al menos una vez, la **ecuación (3)** especifica el concepto de CA.

$$CA(N; P, v, t) \quad (3)$$

Donde N representa las filas de la matriz, P es el número de parámetros (columnas de la matriz), v el alfabeto que indica el número de posibles valores que puede tomar cada componente o celda de la matriz y t representa la fuerza o grado de interacción de las columnas o parámetros del CA.

Si se tiene un arreglo cuyos valores son solo 0 y 1 se dice que tiene un alfabeto dos (2) o binario. Los CA Pueden aplicarse en cualquier disciplina donde se requiera realizar un gran número de pruebas o donde intervengan un gran número de parámetros que interactúan entre si [12][14].

EJEMPLO: La **Figura 4** muestra el CA (5; 4, 2, 2) con 5 filas (N), 4 columnas (P), alfabeto binario (v=2) y fuerza dos (t=2). Por ser un CA binario y tener fuerza 2, se espera que en cualquier par de columnas del CA se encuentren los valores {0,0}, {0,1}, {1,0}, {1,1} al menos una vez [14].

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

Figura 4. Ejemplo de CA (5; 4, 2, 2)

Con la configuración anterior se puede probar un software con 4 parámetros binarios como el que se presenta en la **Tabla 3**, el cual se espera funcione sobre 2 navegadores de internet, con 2 tipos de Servidores Web, que use 2 tipos de Bases de Datos y cuente con 2 tipos de métodos de pago.

Valor	Navegador Web	Servidor Web	Base de Datos	Método de Pago
0	Firefox	Apache	MySQL	Visa
1	Edge	IIS	PostgreSQL	MasterCard

Tabla 3. Valores de los parámetros de un sistema web

Para realizar todas las posibles combinaciones entre parámetros (pruebas exhaustivas) se requieren 16 (2^4) pruebas. No obstante, mediante el uso de un CA de fuerza 2 (**Figura 4**), se determina que solo es necesario realizar 5 casos de prueba (**Tabla 4**), cada fila corresponde a una prueba que se debe hacer para detectar errores, además esta combinación garantiza que se tenga en cuenta cada posible combinación de parejas de columnas al menos una vez.

Prueba	Navegador Web	Servidor Web	Base de Datos	Método de Pago
1	Edge	IIS	PostgreSQL	Visa
2	Firefox	Apache	MySQL	Visa
3	Firefox	IIS	MySQL	MasterCard
4	Edge	Apache	MySQL	MasterCard
5	Firefox	Apache	PostgreSQL	MasterCard

Tabla 4. Tabla de prueba usando CA (N=5, P=4, v=2 t=2)

2.1.4 Torres de arreglos de cobertura(TCA)

Un TCA (Torre de arreglo de cobertura o arreglo de cobertura incremental) de altura h se define como una sucesión de h + 1 arreglos de cobertura C_0, C_1,…, C_h, donde C_0 es un CA de fuerza t y P factores, este CA inicial es llamado base del TCA. Para el resto, i = 1,2,…, h , cada C_i es un CA de Nv^i filas, fuerza t + i y P + i factores [21]. De manera que cada TCA de una fuerza y número de factores superiores tiene contenido los CAs de fuerzas y factores inferiores [14]. La **Figura 5** ilustra el concepto de TCAs [22].

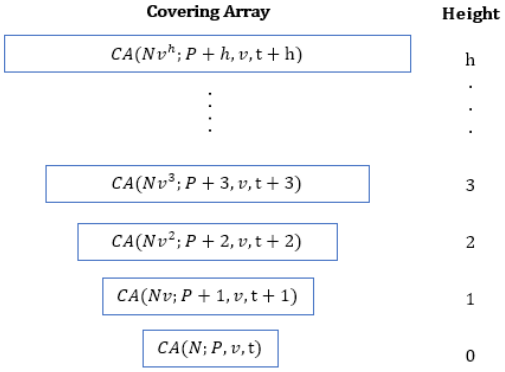

Figura 5. TCA de altura h conformado por h+1 CAs de fuerza t, t+1, …, t+h.

La **Figura 6** es un ejemplo de un TCA que en sus primeras cuatro filas contienen el CA (4; 3, 2, 2) con 4 filas (N), 3 columnas (P), alfabeto binario (v=2) y fuerza 2 (t), y luego en todas sus ocho filas contiene el CA (8; 4, 2, 3) con 8 filas (N), 4 columnas (P), alfabeto binario (v=2) y fuerza 3 (t).

$$\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & 0 \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & 0 \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Figura 6. TCA (8, 4, 2, 3)

2.2 ESTADO DEL ARTE

En este apartado, se presentan los trabajos recientes y más relevantes relacionados con análisis y mejoras teóricas al algoritmo original de RF, junto a mejoras prácticas en aplicaciones de diferentes campos.

En 2017 [23] se propuso la integración de un algoritmo denominado CURE-SMOTE y un algoritmo hibrido basado en RF. CURE (Clustering Using Representatives) agrupa las muestras de clase menos representativas y SMOTE (synthetic minority oversampling technique) elimina ruido y valores atípicos. Después de aplicar CURE y SMOTE, se genera el conjunto de datos que se usa para resolver el problema de clasificación, usando muestras aleatorias entre puntos representativos y datos de las clases menos representativas. Luego, la eliminación de características redundantes, la selección de características, la optimización de parámetros y la definición del número de sub-características, se realiza mediante tres algoritmos híbridos que usan RF: Un algoritmo de bosques aleatorios genéticos (GA-RF), un algoritmo de bosques aleatorios por enjambre de partículas (PSO-RF) y un algoritmo de bosques aleatorios basado en enjambre de peces (AFSA-RF). El algoritmo RF híbrido (GA-RF, PSO-RF o AFSA-R) tiene como función objetivo minimizar el error OOB (Out of bag). Los resultados muestran que el algoritmo CURE-SMOTE minimiza el ruido de la distribución de los datos original, y que los algoritmos híbridos superan al RF original [9] en la medida F, G-mean, AUC y error OOB.

También en 2017 [24] se propuso un sistema para mejorar la calidad de la clasificación de un RF. Inicialmente se descompone el conjunto de datos utilizando agrupamiento (clustering), luego para poder aplicar RF en dichos datos descompuestos se establecen tres parámetros principales: número de árboles que forman el conjunto, número de características para dividir en cada nodo y un vector que representa el número de clusters en cada clase. Para optimizar las configuraciones de los tres parámetros se usó un algoritmo genético. Los resultados muestran que el método propuesto aumenta la calidad del clasificador.

En 2016 [25] se propuso una versión de RF con un método de selección de características sensible al costo denominado feature-cost-sensitive Random Forest (FCS-RF). En FCS-RF se incorpora el costo de las características en el proceso de construcción del árbol base de decisión para producir subconjuntos de características de bajo costo. El algoritmo selecciona una característica con una probabilidad inversamente proporcional a su costo asociado, en lugar de ser seleccionada de manera aleatoria. los resultados muestran que FCS-RF es principalmente útil en los casos en que hay características redundantes o de mayor costo. En este mismo año (2016) [26] se aborda la identificación del número óptimo de clasificadores base (árboles) requeridos para un determinado conjunto de datos. Para lograr este objetivo plantean el uso de una poda selectiva, basada en las medidas estadísticas de los árboles individuales y el Coeficiente de Correlación de

Matthew, en el que se establece un umbral para podar del bosque a todos los árboles que no satisfagan esta condición. Los resultados experimentales revelan que, en promedio, en 78% de los árboles podados en 26 conjuntos de datos del repositorio UCI el impacto fue positivo, mostrando una exactitud de clasificación igual o superior en comparación con el algoritmo original propuesto por Breiman [9].

Como ejemplo del ensamble de técnicas, también en 2016 [27], se presentó un trabajo cuyo principal objetivo fue la generación de un bosque aleatorio compuesto de árboles de decisión diversos (mayor número de características distintas entre ellos) como sea posible, empleando durante la fase de entrenamiento el algoritmo de Naive Bayes para generar predicciones y probabilidades de pertenencia a clases que se concatenan como nuevas características al espacio de características original. Después de la concatenación, un clasificador de RF se entrena usando el nuevo vector de características. Los resultados demuestran que, en la mayoría de los casos, la ampliación del espacio de características de un RF mediante el método propuesto puede aumentar el rendimiento en términos de calidad de la clasificación.

En 2014 [28] se presentó una técnica para encontrar el número adecuado para el subespacio de características (K) empleado en la división de un nodo en cada árbol de decisión base. El número de características para el subespacio se establece a través de un número aleatorio perteneciente a un rango calculado mediante el número de instancias resultantes tras aplicar el criterio de partición del algoritmo CART (Classification and Regression Trees [29]), y el número total de instancias de la muestra Bootstrap del árbol de decisión. Este cálculo se realiza dinámicamente para cada uno de los nodos y permite garantizar la creación de árboles de decisión base diversos. Los resultados muestran que la técnica propuesta mejora significativamente la exactitud (accuracy) del clasificador.

En 2012 [30]. se presenta Dynamic Random Forest (DRF). DRF realiza un procedimiento adaptativo de inducción de árboles. La idea principal es guiar la inducción del árbol para que cada árbol complemente tanto como sea posible los árboles existentes en el conjunto. Esto se hace mediante un remuestreo de los datos de entrenamiento, inspirados por algoritmos de boosting, y combinados con otros procesos de aleatorización utilizados en los métodos tradicionales de RF. DRF muestra una mejora significativa en términos de exactitud en comparación con el algoritmo RF de Breiman [9]. En este mismo año (2012) [31] se propuso un framework de regularización de árboles, el cual permite a muchos modelos de árboles realizar de manera eficiente la selección de características. La idea clave del framework es penalizar con λ (un coeficiente $\in [0, 1]$ basado en la ganancia de información), la selección de una característica usada para la división de un nodo, en los casos en que su índice de calidad sea similar a la de características utilizadas en divisiones anteriores. Por tanto, se espera que un modelo de árbol regularizado contenga un conjunto de características informativas, pero no redundantes. Los

resultados muestran que emplear las características obtenidas por el método propuesto aumenta la calidad del Clasificador Random Forest.

En 2011 [32] se propone el uso de modelos de árboles oblicuos como aprendices base del algoritmo. Los RF “oblicuos” se centran en la partición recursiva óptima de los nodos, para lo cual, en cada división binaria recursiva, se muestrea un nuevo conjunto de características sin reemplazo, y se busca la división óptima en el subespacio abarcado por estas características. Para la búsqueda de la división optima se utilizan modelos discriminativos lineales, en lugar de coeficientes aleatorios usados en el RF de Breiman [9]. Los resultados muestran que Random Forest con divisiones ortogonales obtiene buenos resultados en conjuntos de datos factoriales, en los datos numéricos y espectrales. Esta propuesta superó un amplio rango de clasificadores.

En 2009 [11] se realizó un estudio sobre la parametrización del algoritmo de referencia Forest-RI. En este algoritmo, se usa un principio de aleatorización durante el proceso de inducción del árbol, que selecciona al azar las características K en cada nodo, entre las cuales se elige la mejor división. La fuerza de la aleatorización en la inducción del árbol es liderada por el hiper parámetro K, que juega un papel importante en la construcción de clasificadores de RF precisos. En el estudio se evaluó el algoritmo Forest-RI en varios problemas de aprendizaje automático y con diferentes configuraciones de K para comprender la forma en que actúa sobre el rendimiento de RF. Los resultados muestran que las configuraciones predeterminadas utilizadas tradicionalmente en la literatura no permiten producir el mejor RF posible en términos de precisión, en la mayoría de los casos. Sin embargo, los resultados ilustran que uno de ellos, es decir, $K=\sqrt[2]{V}$ con V como el número de características, es un ajuste razonable para inducir un Random Forest casi óptimo.

En 2008 [33] se propuso un algoritmo llamado Forest-RK, basado en Forest-RI [9]. En Forest-RK, se propone un nuevo método de inducción, en el cual se plantea una alternativa a los ajustes arbitrarios del hiper parámetro K. Para la selección aleatoria de características se escoge el valor de K aleatoriamente para cada división de un nodo con el objetivo de generar una mayor diversidad en los árboles que componen el bosque, contrario a Forest-RI donde el valor de K es idéntico para todos los árboles de decisión. Los resultados muestran que este nuevo método es estadísticamente más exacto que el RF de Breiman [9].

El algoritmo de RF (2001) [9] consiste en un ensamble de árboles de decisión. Inicialmente en el proceso de construcción de un RF se aplica bootstrapping sobre el conjunto de datos de entrenamiento para producir muchos subconjuntos de datos distintos[9][18]. Luego cada subconjunto se usa para construir un árbol de decisión. En el proceso de crecimiento del árbol, la partición de cada nodo depende de las características seleccionadas aleatoriamente con respecto a todas las características presentes en el conjunto de datos[11]. En el RF no se aplica poda y cada uno de los arboles generados funciona como un clasificador base. Para definir

la clase de una instancia, se recibe el voto de cada uno de los clasificadores base y se realiza una ponderación que determina la clase respectiva[7][19].

CAPÍTULO 3

3 PROPUESTA

En este capítulo primero se realiza un resumen del proceso de investigación realizado en el trabajo de grado. Luego, se describe la propuesta de Random Forest basada en Covering Array (RFCA), a partir de RFCA y su mecanismo de integración de los Covering Array (CA) se generaron variaciones al algoritmo con el fin de establecer la manera que mejor integra los Covering Array como mecanismo de selección de características. En el **Anexo 3** se presenta la propuesta de Random Forest basada en Torres de Covering Array (RFTCA) en detalle, no se presenta en este documento debido a que no fue la alternativa seleccionada.

3.1 PROCESO DE INVESTIGACION REALIZADO

A continuación, se presenta un recuento del desarrollo del proyecto que contiene. El proyecto se desarrolló en cuatro iteraciones, tres de las iteraciones centradas en los objetivos específicos y una cuarta de documentación y divulgación de resultados, en la que se incluye la escritura de la monografía y un artículo además de la presentación de dichos documentos. La cuarta iteración se llevó a cabo en forma paralela a las otras iteraciones. El cumplimiento de las iteraciones permitió realizar dos propuestas finales, una para Random Forest basado en CA (RFCA) y otra para Random Forest basado en TCA (RFTCA), ambas obtenidas en el tiempo definido para el trabajo de grado.

3.1.1 Primera Iteración

Esta etapa del proyecto se relacionó con el desarrollo del primer objetivo específico. En esta etapa se estudió la arquitectura e implementación de RF en Weka, para ello se descargó el código fuente de weka y se revisó la arquitectura general del framework, se analizó la manera en que interactúan las clases y la forma como se usan entre ellas, también se identificó la implementación de RF y las clases asociadas con el clasificador, entre las cuales destacan principalmente la clase Bagging y Random Tree.

En la clase Bagging se generan las muestras Bootstrap empleadas por el algoritmo Random Forest. La clase Random Tree permite construir un árbol de decisión con K características seleccionadas aleatoriamente, por otra parte, en la implementación, el número de árboles se fija de manera manual. En este sentido, se observó que los CA y TCA binarios permiten establecer un mecanismo distinto a la selección aleatoria, cada fila del CA o TCA binario define para su árbol

correspondiente cuales características usar en su construcción, donde 0 indica la ausencia y 1 la presencia de una característica. El grado de interacción entre las características se determina con el parámetro fuerza (t) del CA o TCA, adicionalmente, el número de filas del CA o TCA permite fijar el número de árboles. En consecuencia, se buscó, la manera de vincular los CA y los TCA en la nueva versión de RF, realizando ejecuciones paso a paso del algoritmo RF de Weka, para identificar, que componentes o clases específicas debían modificarse y cuales adicionarse en el framework, con el fin de incluir el uso de CA binarios de fuerza 2 hasta fuerza 7 y TCA binarios de fuerza 2 hasta 6. Los archivos de los CA y TCA binarios fueron creados en el CINESTAV Tamaulipas (México) y están en un formato .ca y. caGold respectivamente.

3.1.2 Segunda Iteración

Esta etapa del proyecto se relacionó con el segundo y tercer objetivo específico del proyecto. Con base en las consideraciones obtenidas en la primera iteración sobre la arquitectura de Weka y las características de los archivos .ca y. caGold, se buscó la manera de integrar el mecanismo de selección de características con CA y TCA. Bajo este contexto se optó por crear en Weka las siguientes clases relacionadas con arreglos de cubrimiento: CAs, BaggingCA, RFCA y RandomTree_RFCA, mientras que, para la propuesta de torres de arreglos de cubrimiento, las clases: TCAs, BaggingTCA, RFTCA y RandomTree_TCA. Dichas clases se crearon con base en las clases implicadas en el algoritmo Random Forest.

Además, se crearon clases adicionales para cumplir funciones de: Gestión de los archivos .ca y. caGold, identificación, lectura y carga del CA o TCA correspondiente al valor de fuerza solicitado, escritura de los resultados, entre otras. Luego, tras obtener una versión funcional de la propuesta RFCA y RFTCA, se generaron una serie de variaciones, las cuales fueron evaluadas en los conjuntos de datos seleccionados en el repositorio de la UCI obteniendo los resultados para exactitud, medida F y tiempo de ejecución.

A partir de los resultados de exactitud y medida F se aplicaron las pruebas estadísticas no paramétricas de Friedman y Wilcoxon mediante el software Keel [34] y con los resultados de dichas pruebas se determinó la propuesta que mejor integra los Covering Arrays en Random Forest y la propuesta que mejor integra las Torres de Covering Arrays en Random Forest.

3.1.3 Tercer Iteración

En esta tercera iteración, se terminó de cumplir con el segundo objetivo específico. Es decir, se centró en implementar e incorporar en un paquete weka las propuestas de Random Forest basada en CA y Random Forest basada en TCA. La implementación en un paquete weka permitió que las propuestas puedan ser utilizadas para cualquier Conjunto de Datos cargado en el software Weka.

Un paquete Weka contiene recursos como: código compilado, código fuente, javadocs, archivos de descripción de paquete (metadatos), bibliotecas de terceros y archivos de propiedades de configuración. La instalación, gestión y acceso a un paquete Weka se da a través del sistema de gestión de paquetes del software Weka. Además, se revisó que los resultados de los paquetes disponibles se pudieran instalar correctamente en otros equipos y que los resultados coincidieran con los de la iteración anterior, Además, en esta iteración se desarrolló la última parte del tercer objetivo específico, relacionada con la comparación de los algoritmos propuestos frente al algoritmo de referencia de Breiman [9]. Para ayudar en este propósito, se emplearon una serie de vistas minables a las cuales se les aplicaron modelos de árboles de decisión con el fin de identificar reglas o condiciones en que el algoritmo RFCA ganaba o perdía.

3.2 RANDOM FOREST BASADO EN COVERING ARRAYS (RFCA)

En el Random Forest de Breiman [9], el hiper parámetro número de árboles en el bosque (M), se establece de manera arbitraria o empírica, y aunque un aumento en el número de árboles puede aumentar linealmente la calidad del modelo, hay un cierto valor en el que aumentar el número de árboles no mejora e incluso disminuye la exactitud del modelo [10].

El proceso de selección de características denotado como Forest-RI emplea en la división de cada nodo, pequeños grupos de características seleccionadas aleatoriamente. El tamaño K de los grupos es fijo y generalmente es igual al primer entero menor que $\log_2 P + 1$, donde P es el número total de características del conjunto de datos, sin incluir la característica objetivo o de clase.

En este contexto RFCA integra los Covering Arrays (Arreglos de cubrimiento) como mecanismo para mejorar el proceso de selección de las características y eliminar la necesidad de fijar y afinar el híper parámetro número de árboles (M). RFCA emplea CAs de alfabeto binario (v = 2). Esto significa que cada componente del CA tiene únicamente valores {0, 1}, los subconjuntos de características candidatas empleadas en la construcción de cada uno de los arboles del RF se determinan con las filas del CA. El parámetro P del CA se establece a través del número total de características (P) del conjunto de datos a ser clasificado, sin incluir la característica de clase u objetivo. A continuación, se presenta un ejemplo de ejecución de RFCA y luego se describe el algoritmo.

3.2.1 Ejemplo de RFCA

Con el fin de ilustrar el comportamiento de RFCA se utiliza una versión reducida del Conjuntos de Datos Churn, del Repositorio de la UCI (Universidad de California en Irvine) [20]. El termino Churn se utiliza para indicar que un cliente abandona el servicio de una empresa para tomar el de la competencia. El conjunto de Datos de la UCI contiene 20 características de información sobre 3333 clientes, junto con una

indicación de si el cliente abandona o no la empresa. En esta versión de ejemplo se usan sólo 6 de las características de mayor importancia en 20 registros de clientes. Las características utilizadas son: Plan Internacional, plan correo de voz, total minutos usados en la mañana, total minutos que habla en la tarde, total minutos que habla en la noche, total minutos de llamadas internacionales y la característica objetivo Churn. En la **Tabla 5**, se observa la descripción de las características del conjunto de Datos, cada característica cuenta con un nombre corto, A1, A2 y así sucesivamente. Para la construcción de los arboles se emplea el CA de la **Figura 7** con 6 filas (N), 6 columnas (P), alfabeto binario (v=2) y fuerza 2 (t).

A1	A2	A3	A4	A5	A6	Clase
Plan Internacional	**Plan Correo de voz**	**Total minutos Mañana**	**Total minutos Tarde**	**Total minutos Noche**	**Total minutos Internacionales**	**Churn**
Conjunto de Datos de Entrenamiento						
no	no	178.7	233.7	131.9	9.1	FALSE
no	yes	148.5	114.5	178.3	6.5	FALSE
no	yes	164.1	219.1	220.3	12.3	FALSE
yes	no	197.2	188.5	211.1	7.8	FALSE
no	no	124.9	300.5	192.5	11.6	FALSE
no	no	115.4	209.9	280.9	15.9	FALSE
yes	no	140	196.4	120.1	9.7	TRUE
no	no	193.9	85	210.1	13.2	FALSE
no	no	321.1	265.5	180.5	11.5	TRUE
no	no	118.4	249.3	227	13.6	TRUE
no	no	169.8	197.7	193.7	11.6	FALSE
no	no	193.4	116.9	243.3	9.3	FALSE
no	no	106.6	284.8	178.9	14.9	FALSE
no	no	134.7	189.7	221.4	11.8	FALSE
no	yes	156.2	215.5	279.1	9.9	FALSE
no	no	231.1	153.4	191.3	9.6	FALSE
Conjunto de Datos de Prueba						
no	no	180.8	288.8	191.9	14.1	FALSE
yes	no	213.8	159.6	139.2	5	FALSE
no	yes	234.4	265.9	241.4	13.7	FALSE
no	yes	265.1	197.4	244.7	10	FALSE

Tabla 5. Descripción del Conjunto de Datos adaptado

En este ejemplo se utiliza el 80% del conjunto de datos para entrenamiento y el 20% restante como conjunto de prueba. El algoritmo Random Forest se basa en el proceso de bootstrapping, lo que implica que genera un conjunto M de árboles donde cada árbol es generado con su propio conjunto de datos de entrenamiento que es el resultado de una selección aleatoria de Datos del conjunto de datos de entrenamiento original.

Conforme al conjunto de datos del ejemplo se requiere de un Covering Array Binario que permita cubrir las 6 características de entrada, en el presente ejemplo se emplea el Covering Array Binario de fuerza 2 presentado en la **Figura 7**.Conforme a este CA, el número de árboles es seis (número filas CA), no obstante, debido a

que la fila 6 del CA contiene únicamente ceros, es decir, no selecciona ninguna característica, entonces los arboles totales a construir son cinco (5).

$$CA(N=6,P=6,v=2,t=2)=\begin{bmatrix}1&1&1&0&0&0\\1&0&1&0&1&1\\0&0&0&1&1&0\\1&1&0&1&1&1\\0&1&1&1&0&1\\0&0&0&0&0&0\end{bmatrix}$$

Figura 7. CA (6; 6, 2, 2)

Cada fila del CA binario define para su árbol correspondiente cuales características (características del conjunto de datos de entrenamiento original) serán usadas en su construcción. Donde 0 indica la usencia y 1 la presencia de una característica en el subconjunto de características candidatas. Teniendo en cuenta por ejemplo la fila dos $[1|0|1|0|1|1]$ del CA de la **Figura 7** se puede afirmar que el árbol se construye teniendo en cuenta las características uno, tres, cinco y seis (A1, A3, A5, y A6), que corresponden a Plan Internacional, Total minutos Mañana Total Minutos Noche y Total minutos Internacionales.

El RF original cuenta con un hiper parámetro K que define el número de características que se deben tener en cuenta para la creación de cada árbol. Este valor se define según la propuesta de Breiman [9], como el primer entero menor que $\log_2 P + 1$, donde P es el número de características de entrada del conjunto de datos de entrenamiento original que en este caso es igual a tres, $\mathrm{K} = \lfloor \log_2(6) + 1 \rfloor = 3$.

Construcción de $Arbol_1$ basado en la fila 1 $[1|1|1|0|0|0]$ del CA. Las características seleccionadas en el subconjunto son: A1 (Plan Internacional), A2 (Plan Correo de Voz) y A3 (Total Minutos Mañana). Como el número de características seleccionadas en la fila del CA coincide con el valor de k (3) se seleccionan todas. Tras hacer el muestreo de filas se obtiene por ejemplo la tabla de datos (muestra) de la **Figura 8**. Además, al lado derecho de la figura se muestra el árbol base que se obtiene con esa muestra de entrada.

Construcción de $Arbol_2$ basado en la fila 2 $[1|0|1|0|1|1]$ del CA. El número de características del subconjunto seleccionado por esta fila del CA es mayor al número K de características a seleccionar (K = 3) en consecuencia, el algoritmo selecciona 3 características del subconjunto de manera aleatoria y sin repetición. Para el ejemplo se seleccionaron las características: A1 (Plan Internacional), A3 (Total Minutos Mañana) y A5 (Total Minutos Noche). La **Figura 9** presenta la muestra de datos y el árbol base obtenido.

Construcción de $Arbol_3$ basado en la fila 3 $[0|0|0|1|1|0]$ del CA. En la selección de características, se escogen las características de la posición 4 (Total Minutos Tarde) y 5 (Total Minutos Noche). En este caso, el número de características del subconjunto seleccionado por esta fila del CA es menor al número k de características a seleccionar (k = 3), en consecuencia, para esos casos el algoritmo

selecciona las características faltantes de manera aleatoria y sin repetición (en este caso una característica adicional). Para el ejemplo se selecciona la característica A3 (Total Minutos Mañana). La **Figura 10** presenta la muestra de datos y el árbol base obtenido.

Datos para el ARBOL 1			
A1	**A2**	**A3**	**Clase**
no	no	178.7	FALSE
no	yes	148.5	FALSE
yes	no	197.2	FALSE
no	no	124.9	FALSE
no	no	115.4	FALSE
yes	no	140	TRUE
no	no	321.1	TRUE
no	no	193.4	FALSE
no	no	106.6	FALSE
no	yes	156.2	FALSE
no	No	231.1	FALSE

A1 = no
    A3 < 276.1: FALSE (12/0)
    A3 >= 276.1: TRUE (1/0)
A1 = yes
    A3 < 168.6: TRUE (2/0)
    A3 >= 168.6: FALSE (1/0)

Figura 8. Árbol generado para el conjunto de entrenamiento 1

Datos para el ARBOL 2			
A1	**A3**	**A5**	**Clase**
no	178.7	131.9	FALSE
no	148.5	178.3	FALSE
yes	197.2	211.1	FALSE
no	115.4	280.9	FALSE
yes	140	120.1	TRUE
no	193.9	210.1	FALSE
no	193.4	243.3	FALSE
no	134.7	221.4	FALSE
no	156.2	279.1	FALSE
no	231.1	191.3	FALSE

A5 < 126: TRUE (1/0)
A5 >= 126: FALSE (15/0)

Figura 9. Árbol generado para el conjunto de entrenamiento 2

Datos para el ARBOL 3			
A3	**A4**	**A5**	**Clase**
178.7	233.7	131.9	FALSE
148.5	114.5	178.3	FALSE
164.1	219.1	220.3	FALSE
197.2	188.5	211.1	FALSE
321.1	265.5	180.5	TRUE
169.8	197.7	193.7	FALSE
193.4	116.9	243.3	FALSE
106.6	284.8	178.9	FALSE
156.2	215.5	279.1	FALSE
231.1	153.4	191.3	FALSE

A4 < 249.6: FALSE (12/0)
A4 >= 249.6
    A3 < 213.85: FALSE (3/0)
    A3 >= 213.85: TRUE (1/0)

Figura 10. Árbol generado para el conjunto de entrenamiento 3

Construcción de $Arbol_4$ con base en la fila 4 $[1|1|0|1|1|1]$ del CA. Las características definidas por esta fila son la A1 (Plan Internacional), A2 (Plan Correo de Voz), A4 (Total Minutos Tarde), A5 (Total Minutos Noche) y A6 (Total Minutos Internacionales). Las 3 características seleccionados aleatoriamente y sin repetir de este subconjunto definido por el CA son para el ejemplo las características A1, A2 y A6. La **Figura 11** presenta la muestra de datos y el árbol base obtenido.

Datos para el ARBOL 4			
A1	**A2**	**A6**	**Clase**
no	no	9.1	FALSE
no	yes	6.5	FALSE
yes	no	7.8	FALSE
no	no	11.6	FALSE
no	no	15.9	FALSE
yes	no	9.7	TRUE
no	no	11.5	TRUE
no	no	9.3	FALSE
no	no	14.9	FALSE
no	yes	9.9	FALSE
no	no	9.6	FALSE

```
A6 < 9.65: FALSE (8/0)
A6 >= 9.65
    A6 < 11.55
        A1 = no
            A2 = yes: FALSE (1/0)
            A2 = no: TRUE (1/0)
        A1 = yes: TRUE (2/0)
    A6 >= 11.55: FALSE (4/0)
```

Figura 11. Árbol generado para el conjunto de entrenamiento 4

Finalmente, para la construcción de $Arbol_5$, se toma como base la fila 5 $[0|1|1|1|0|1]$ del CA. En este caso hay 4 características, a saber: A2 (Plan Correo de Voz), A3 (Total Minutos Mañana), A4 (Total Minutos Tarde) y A6 (Total Minutos Internacionales) de los cuales se seleccionan aleatoriamente 3 y sin repetir, a saber: la A2, A4 y A6. La **Figura 12** presenta la muestra de datos y el árbol base obtenido.

Datos para el ARBOL 5			
A2	**A4**	**A6**	**Clase**
yes	114.5	6.5	FALSE
no	188.5	7.8	FALSE
no	300.5	11.6	FALSE
no	196.4	9.7	TRUE
no	85	13.2	FALSE
no	116.9	9.3	FALSE
no	284.8	14.9	FALSE
yes	215.5	9.9	FALSE
no	153.4	9.6	FALSE

```
A4 < 192.45: FALSE (10/0)
A4 >= 192.45
    A6 < 9.8: TRUE (1/0)
    A6 >= 9.8: FALSE (5/0)
```

Figura 12. Árbol generado para el conjunto de entrenamiento 5

Una vez se crean los árboles, se puede realizar el uso o prueba del clasificador, enviando las instancias de prueba a cada uno de los árboles y la etiqueta de clase final se asigna en función de un voto mayoritario. A partir de los resultados presentados en la **Tabla 6,** se puede concluir que el clasificador obtiene 100% de instancias correctamente clasificadas ya que en todos los casos el valor real y el valor predicho coinciden.

	Árbol 1	Árbol 2	Árbol 3	Árbol 4	Árbol 5	Clase Real	Resultado (Voto mayoritario)
Instancia de prueba 1	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	**FALSE**
Instancia de prueba 2	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	**FALSE**
Instancia de prueba 3	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	**FALSE**
Instancia de prueba 4	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	**FALSE**

Tabla 6. Resultados Conjunto de Datos de prueba

El **Algoritmo 1** resume la técnica de construcción de un conjunto de árboles de decisión usando bagging y Covering Arrays para la selección de características (RFCA). La función *entrenarDT (T', K)* realiza el entrenamiento de un árbol de decisión en una muestra de arranque *T' y K* características seleccionadas con base en cada fila de un CA. El proceso de entrenamiento de un árbol de decisión se presenta en el **Algoritmo 2**.

Algoritmo RFCA (Random Forest basado en Covering Arrays)

Entradas: T **/*** Conjunto de datos de entrenamiento */
f /* Fuerza del Covering array */
lsize /* Tamaño límite de la hoja */
Test /* Conjunto de datos de prueba */
Salida: Etiqueta de clase para el conjunto de datos de prueba.

Inicio
01: r = numRows (T) /* Numero de instancias del Conjunto de Datos ***/**
02: P = numAttributes (T)/* Numero de características del Conjunto de Datos ***/**
03: ca = loadCA (f, P) /* Carga el Covering Array (CA) acorde a f y P */
04: M = numRowsCA (ca) /* Numero de árboles definido como número de filas del CA */
05: K = $\lfloor \mathbf{log_2}(\boldsymbol{P}) + \mathbf{1} \rfloor$ /* Numero de características a usar en la construcción del árbol ***/**
06: att = listAttributes (T) /* Índices de las características del conjunto de datos */
07: Desde i = 1 hasta M hacer
08: T' = bootstrap (T) **/*** Selecciona $\frac{2*r}{3}$ puntos (instancias del conjunto de datos), con reemplazo, uniformemente en T */
09: subAS = SelAttributesCA (ca[i]) **/*** Basado en la fila i selecciona las características ***/**
10: Tree = entrenarDT (T', K, subAS, att, lsize)
11: add Tree to RF
12: Fin desde
13: Una vez que se crean M árboles, se pasan la instancia de prueba en Test a cada árbol y la etiqueta de clase se asignará en función de la mayoría de los votos.
Fin

Algoritmo 1. RFCA (Random Forest basado en Covering Arrays)

En la línea 01 se define la variable r como el número de filas (instancias) del conjunto de datos de entrenamiento.

En la línea 02 se inicializa la variable P como el número de características del conjunto de datos de entrenamiento.

Para contextualizar el procedimiento indicado en la línea 03 es necesario señalar que: los CA son archivos en formato .ca, cada archivo esta nombrado de la siguiente manera: N**a**k**b**v2^**b**t**c**, donde **a** es el número de filas del CA, **b** es el número de columnas del CA, v2 indica que el archivo corresponde a un Covering array binario y **c** es la fuerza del CA, por ejemplo, el archivo N**6**k**10**v2^**10**t**2**.ca, corresponde a un CA de 6 filas, 10 columnas, binario, y de fuerza 2. Estos archivos .ca están distribuidos en 6 carpetas (una para cada fuerza) nombradas como t2, t3, t4, t5, t6 y t7. Bajo este contexto, con el parámetro f se identifica la carpeta que contiene los archivos .ca de fuerza f. Una vez identificada la carpeta, se carga la información de sus archivos .ca en una lista de objetos, donde cada objeto contiene los siguientes atributos: nombre completo del archivo .ca (nombreCA) y número de características del CA (numColumnas). Luego, la lista de objetos se organiza de manera ascendente con respecto al número de atributos. Por último, se recorre la lista y se retorna el objeto que tenga en su atributo numColumnas un valor mayor o igual al indicado en el parámetro P (número de características del conjunto de datos).

A partir del atributo nombreCA del objeto retornado se carga el Covering Array correspondiente. No obstante, si dicho Covering Array tiene un numero de columnas mayor que el valor indicado en el parámetro P, se procede a tomar las P primeras columnas de dicho Covering Array. Este proceso de acortar columnas es válido, ya que el Covering Array resultante también contiene todas las combinaciones de los v^t símbolos al menos una vez en las columnas que se extraen.

En la línea 04 se establece el número de árboles(M) como el número N de filas del Covering Array obtenido en la línea anterior (no se cuentan filas que tengan ceros en todas las celdas).

En la línea 05 se establece el número K de características a seleccionar para la construcción de los árboles de decisión de Random Forest. En este caso $\mathbf{K = \lfloor \log_2(P) + 1 \rfloor}$ corresponde al valor indicado por el algoritmo referencia de Breiman [9].

La línea 06 inicializa att con la lista de índices de las características del conjunto de datos (desde 0 hasta P-1). Desde la línea 07 hasta la línea 11 se crean los M árboles de decisión que componen el bosque aleatorio (Random Forest)

En la línea 08 se crea una muestra bootstrap, especificada en la **ecuación (4)** [9]. Como es una muestra bootstrap, una instancia puede ser seleccionada más de una vez y también puede que no sea seleccionada para la construcción de la muestra[9], [18].

$$Muestra\ Bootstrap = \frac{2 * r}{3} \qquad (4)$$

Donde r es el número de instancias del conjunto de datos de entrenamiento.

En la línea 09 se definen los índices de las características con las que se crea un árbol de decisión. Representando la fila i del CA como un conjunto de escalares $[C_{i0}, C_{i1}, \dots, C_{i2}, \dots C_{ij}, \dots, C_{ip-1}]$, donde $j = 0,1, \dots, p-1$ representa el índice de la variable j-esima y C_{ij} ϵ {0,1}. Luego, subAS se construye según la siguiente regla:

Si C_{ij} = 1 entonces j ∈ subAS

Sino j ∉ subAS

En la línea 10 se entrena el árbol de decisión con base en la muestra Bootstrap T', el número K de características requeridos en la construcción de un árbol, las características seleccionadas en la fila del CA (subAS) y el tamaño límite de la hoja (número mínimo de instancias en una hoja de un árbol de decisión). En la línea 11 se añade el árbol i de decisión al bosque aleatorio (RF).

El **Algoritmo 2** muestra una función recursiva que permite la construcción de un árbol base. En la línea 01 se comprueba que el número de instancias de la muestra Bootstrap T' supere el tamaño límite de una hoja en el árbol de decisión, sino es así se detiene el llamado recursivo y se retorna el subárbol (línea 12).

```
Función entrenarDT
Entradas:  T'      /* Muestra Bootstrap */
           K       /* Numero de características aleatorios a seleccionar*/
           subAS   /* Índices de las características seleccionados en la fila del CA */
           att     /* Conjunto de características del conjunto de datos */
           lsize   /* Tamaño límite de una hoja */
Salida:  Árbol, un árbol de decisión entrenado
Inicio
01:  Si numInstances (T') > lsize entonces
02:      subK = Ø
03:      Si size (subAS) >= K entonces
04:          subK = RandomSelect (subAS, K) /*Selecciona uniformemente y sin remplazo un
             subconjunto subK de K características en subAS, subK ⊂ subAS */
05:      Sino
06:          subK = RandomSelectAttributes (att, subAS, K) /* En subK se añaden todos los
             índices de las características de subAS, también, se adicionan índices de
             características del conjunto att seleccionados uniformemente y sin remplazo, que no
             pertenezcan a subAS. Al final, subK queda con K características */
07:      Fin si
08:      (leftT, rightT) = Split (T', subK)      /* Selecciona la mejor división en T' optimizando el
                                                    criterio de partición CART */
09:      Tree.add (entrenarDT (leftT, K, subAS, att, lsize))      /* Hijo izquierdo */
10:      Tree.add (entrenarDT (rightT, K, subAS, att, lsize))     /* Hijo derecho */
11:  Sino
12:      retornar Tree
13:  Fin si
Fin Función
```

Algoritmo 2. Árbol de decisión para Random Forest basado en Covering Arrays

La línea 02 inicializa el subconjunto vacío subK, encargado de almacenar los índices de las características utilizados en la construcción del árbol de decisión.

En la línea 03 se comprueba que el tamaño de subAS (índices de características seleccionados en la fila del CA) sea mayor o igual al número K de características requeridos en la construcción del árbol, si se cumple la condición, la línea 04 selecciona uniformemente y sin remplazo (elegir una característica una única vez) K características en subAS, subK ⊂ subAS.

De no cumplirse la condición (línea 05), en la línea 06 se añaden a subK todos los índices de las características de subAS, también, se adicionan (K – tamaño de subAS) índices de las características del conjunto att seleccionados uniformemente y no pertenecientes a subAS. Al final, subK queda con K características.

En la línea 08, 09 y 10 se crea un árbol de decisión binario a partir del criterio CART, el árbol se construye dividiendo un nodo en dos nodos hijo de manera recursiva, comenzando con el nodo raíz que contiene toda la muestra Bootstrap (T'). Las divisiones se seleccionan de modo que la "impureza" de los nodos hijos sea menor que la del nodo madre. La idea básica es formar grupos homogéneos respecto a la característica que se desea discriminar y a su vez mantener el árbol razonablemente pequeño [29].

3.3 VARIACIONES DE RFCA

En esta sección se describen las variaciones generadas sobre la propuesta Random Forest basada en Covering Array(RFCA) y su mecanismo de selección de características.

3.3.1 RFCA sin parámetro K (RFCA SINK)

En esta variación del algoritmo RFCA, no se emplea el híper parámetro K en la construcción de los árboles base o árboles de decisión. Las características seleccionadas para la construcción de cada árbol se determinan únicamente con la información de la fila del CA, en consecuencia, el número de características presentes en la construcción de los distintos arboles puede variar de acuerdo con el número de unos que se presenten en cada fila. Esta versión elimina las líneas 02 a 07 del **Algoritmo 2** y las sustituye por subK = subAS.

3.3.2 RFCA variación 1 y RFCA variación 2

En las variaciones 1 y 2 de RFCA se modifica la línea 06 del **Algoritmo 2** correspondiente a la selección del conjunto de características en los casos donde el número de características indicado en la fila del CA es menor al valor de K.

En la variación 1 (**RFCA VAR1**) el conjunto de características está compuesto únicamente por las características indicadas en la fila del CA, es decir, a pesar de

que hacen falta características para cumplir con el parámetro K, estos NO se completan. Lo que implica que la línea 06 del **Algoritmo 2** se elimina.

Por otra parte, en la variación 2 (**RFCA VAR2**) se emplea el mecanismo estándar de Random Forest, es decir la selección de K características aleatorios distintos para la construcción del árbol, es decir, se ignora por completo la información de la fila del CA.

3.3.3 RFCA y variaciones raíz cuadrada

En [11][35] se propone la **ecuación (5)** para definir el número K de características seleccionadas. Este valor propuesto para el hiper parámetro K se experimentó en el algoritmo RFCA y sus variaciones mediante la modificación de la línea 05 del **Algoritmo 1**. A las 3 variaciones en el algoritmo que emplean este valor se les denomino: RFCA_SQRT, RFCA_VAR1_SQRT y RFCA_VAR2_SQRT.

$$K = \sqrt[2]{P} \qquad (5)$$

Donde P corresponde al número de características del conjunto de datos de entrenamiento.

CAPÍTULO 4

4 IMPLEMENTACION DE RFCA Y RFTCA EN WEKA

El software libre weka, es un sistema que soporta el desarrollo de diferentes tareas de minería de datos, como: visualización, selección de atributos, clasificación, clustering y reglas de asociación. Además, permite que se agreguen nuevos algoritmos a los ya existentes, entre ellos algoritmos de clasificación. Para añadir un nuevo clasificador se utiliza un paquete weka. Un paquete Weka es un archivo .zip que contiene, código fuente, javadocs, un archivo de descripción del paquete (Description.props), un archivo build_package.xml y un archivo .jar con clases java previamente compiladas. La instalación, gestión y acceso a un paquete weka se realiza a través del sistema de gestión de paquetes del software weka [36].

Con el fin de implementar adecuadamente el paquete weka con la propuesta RFCA (para RFTCA es muy similar), se siguió y respetó la arquitectura general del funcionamiento del algoritmo Random Forest (ver **Figura 13**).

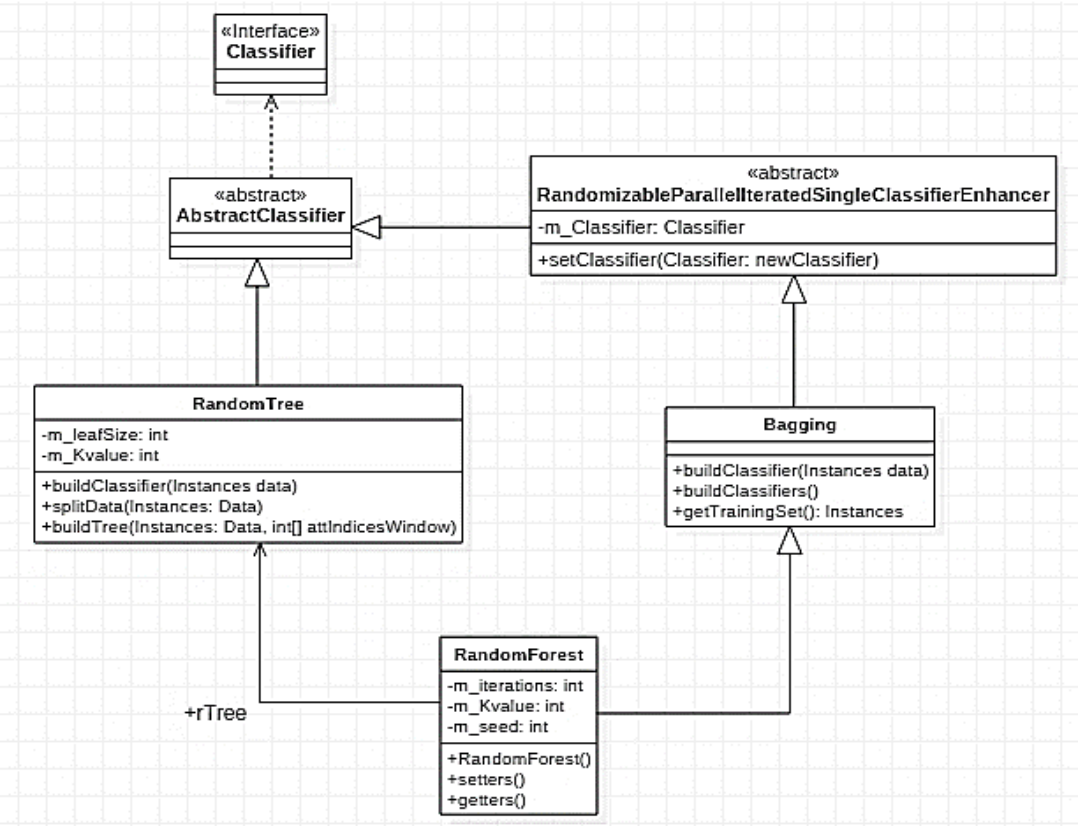

Figura 13. Arquitectura General de Random Forest en Weka (Fuente Propia)

En cuanto al comportamiento de las clases involucradas en el algoritmo, se determinó que la clase Bagging de la **Figura 13** representa un meta clasificador, es decir, un clasificador que utiliza una combinación de múltiples clasificadores. Dicha combinación se realiza de la siguiente manera: el método getTrainingSet, crea subconjuntos de entrenamiento múltiples a partir de un conjunto de entrenamiento. El método buildClassifiers, construye cada clasificador según el algoritmo y el subconjunto de entrenamiento de datos. Finalmente el método buildClassifier integran los resultados de los clasificadores base y obtiene los resultados finales mediante voto ponderado[18].

La relación de herencia que la clase Bagging tiene con la clase abstracta **RandomizableParallelIteratedSingleClassifierEnhancer** permite que la construcción del conjunto de los clasificadores base se realice en paralelo y con una misma semilla. En cuanto a la clase Random Forest, esta presenta una relación de herencia con la clase Bagging, por lo cual indica que la construcción del modelo de clasificación Random Forest se realiza de acuerdo con el comportamiento indicado para Bagging. También, es la clase donde se fija el número de árboles (m_iterations) indicado por el usuario, además, el constructor de RandomForest, es el lugar donde se establece el clasificador base (Random Tree) a través del método setClassifier, el cual fija el objeto rTree. La clase Random Tree construye un árbol de decisión a partir de K características seleccionadas aleatoriamente en cada nodo.

A continuación, la **Figura 14** muestra la arquitectura general de la propuesta RFCA.

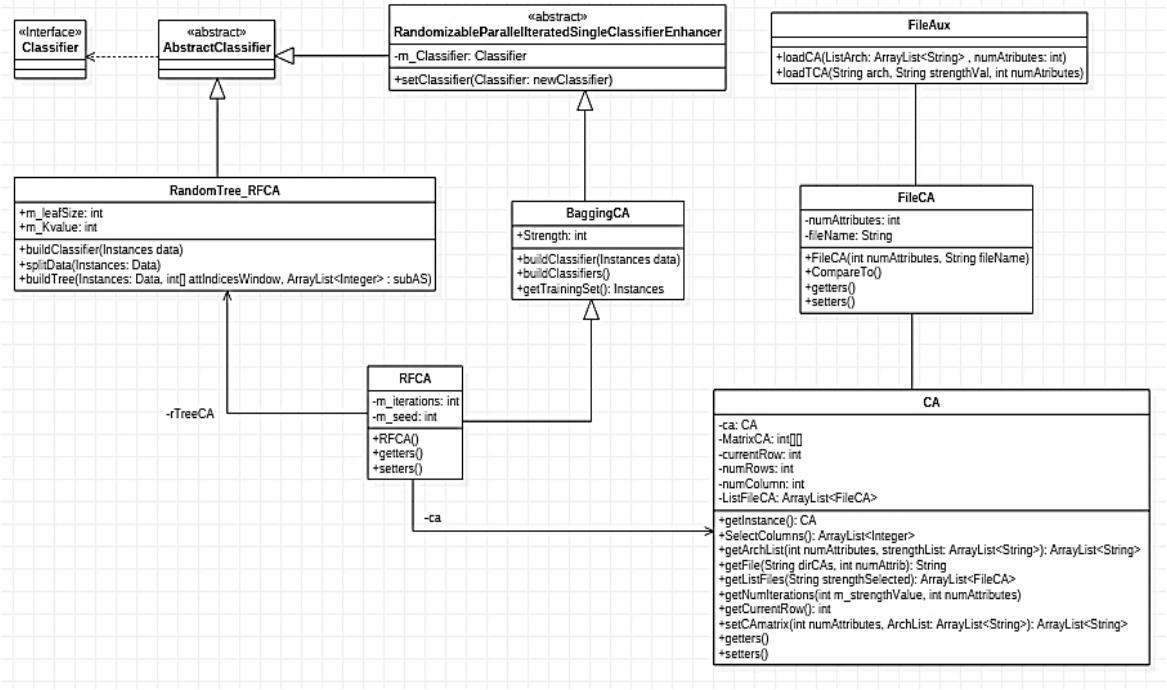

Figura 14. Arquitectura General propuesta para RFCA (Fuente Propia)

La clase BaggingCA de la **Figura 14** representa un meta clasificador, es decir, un clasificador que utiliza una combinación de múltiples clasificadores. Dicha combinación se realiza de la siguiente manera: el método getTrainingSet, crea subconjuntos de entrenamiento múltiples a partir de un conjunto de entrenamiento. Luego, en el método buildClassifiers se recibe el parámetro strength (fuerza), se obtiene el número de características y se envían estos dos parámetros al método setCAMatrix del objeto ca para cargar el Covering Array adecuado con respecto a la cantidad de características del conjunto de datos, tras finalizar la carga del CA se obtiene el número de filas del Covering Array con el método getNumRows, y se fija ese valor en numIterations (número de árboles). El método buildClassifiers, construye cada clasificador según el algoritmo y el subconjunto de entrenamiento de datos. Finalmente, el método buildClassifier integra los resultados de los clasificadores base y obtiene los resultados finales mediante voto ponderado.

La relación de herencia que la clase BaggingCA tiene con la clase abstracta **RandomizableParallelIteratedSingleClassifierEnhancer** indica que la construcción del conjunto de clasificadores base se realiza en paralelo y con una misma semilla. La clase RFCA presenta una relación de herencia con la clase BaggingCA, con lo cual se construye el RFCA de acuerdo con el comportamiento indicado para BaggingCA, en el constructor de RFCA, se establece el clasificador base (RandomTree_RFCA) a través del método setClassifier, el cual fija el objeto rTreeCA. La clase RandomTree_RFCA construye cada árbol de decisión de acuerdo con las K características seleccionadas en el parámetro subAS (ver **Algoritmo 2**).

4.1 INSTALACION DEL PAQUETE WEKA

Hay una serie de paquetes disponibles para Weka que agregan modelos de aprendizaje o extienden la funcionalidad del sistema central de alguna manera. Muchos son proporcionados por el equipo de Weka y otros son de terceros. El sistema de gestión de paquetes del software Weka incluye una instalación para la gestión de paquetes y un mecanismo para cargarlos dinámicamente en tiempo de ejecución. El administrador de paquetes intenta cargar todos los archivos .jar que encuentre en el directorio raíz. Al instalar, el sistema de gestión de paquetes del software usará el valor del campo "Nombre del paquete" en el archivo Descripción.props. La **Figura 15** muestra el contenido del paquete weka con la propuesta RFCA y RFTCA.

La instalación de un paquete en Weka, se realiza por medio del sistema de gestión de paquetes del software, esta funcionalidad se accede, seleccionando la opción Package manager del menú desplegado por la opción Tools (ver **Figura 16**).

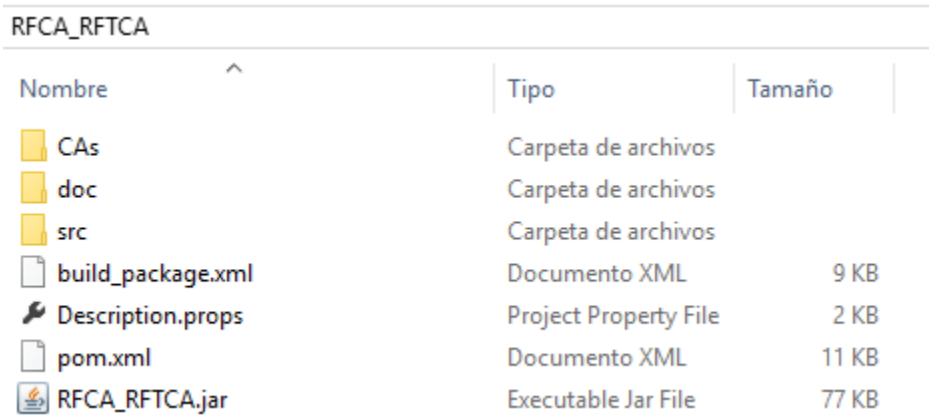

Figura 15. Paquete Weka con propuesta RFCA y RFTCA

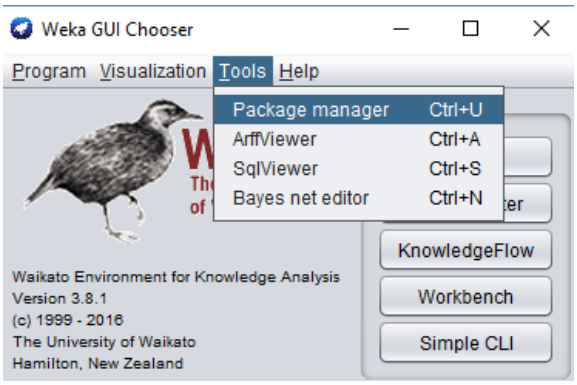

Figura 16. Paso 1 de Instalación de un paquete Weka

Luego en la ventana desplegada correspondiente al sistema de gestión de paquetes del software se selecciona la opción File/URL (ver lado derecho **Figura 17**).

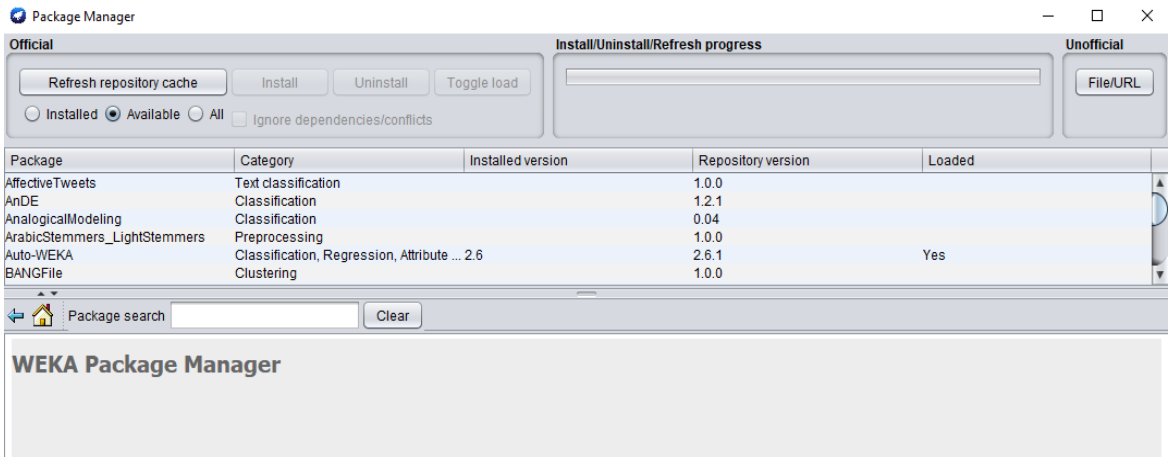

Figura 17. Paso 2 de Instalación de un paquete Weka

En la ventana desplegada en la figura se selecciona la opción Browse (ver parte superior de la **Figura 18**), y luego se despliega la ventana mostrada en la parte inferior de la **Figura 18**, en esa esa se procede a ubicar el archivo en formato .zip

correspondiente al paquete weka, una vez encontrado se selecciona la opción select.

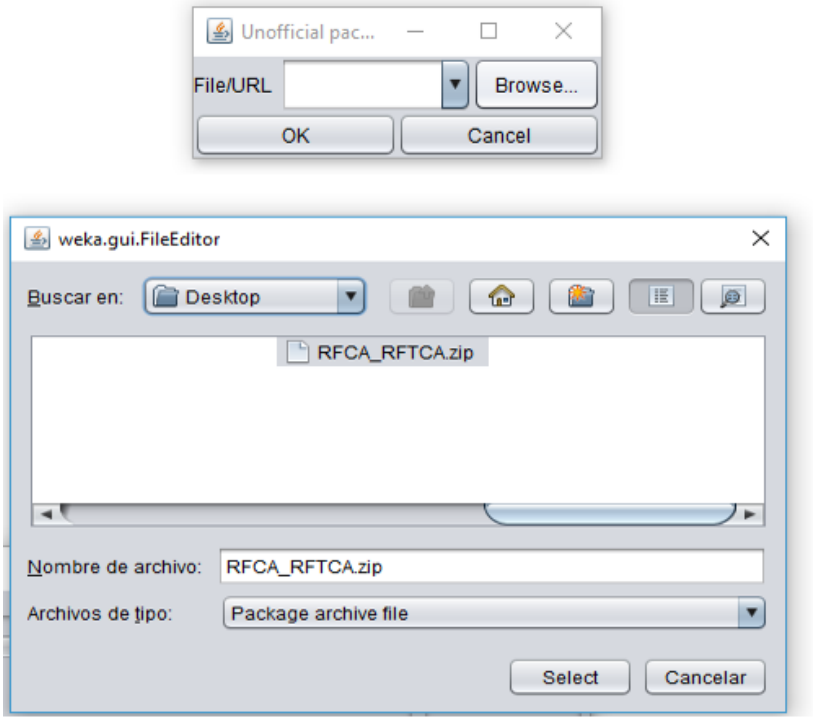

Figura 18. Paso 3 y 4 de Instalación de un paquete Weka

Una vez instalado aparece la dirección del paquete en la manera mostrada en la **Figura 19**, al presionar la opción OK, se inicia con la instalación del paquete weka, una vez finaliza la instalación, el software weka indica que es necesario cerrar y abrir de nuevo el software (ver **Figura 20**).

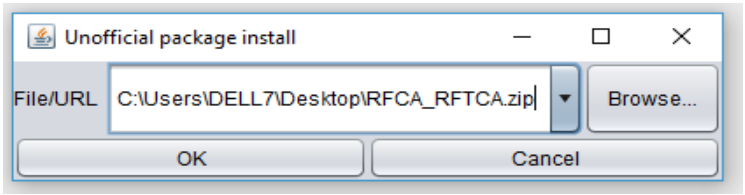

Figura 19. Paso 5 de Instalación de un paquete Weka

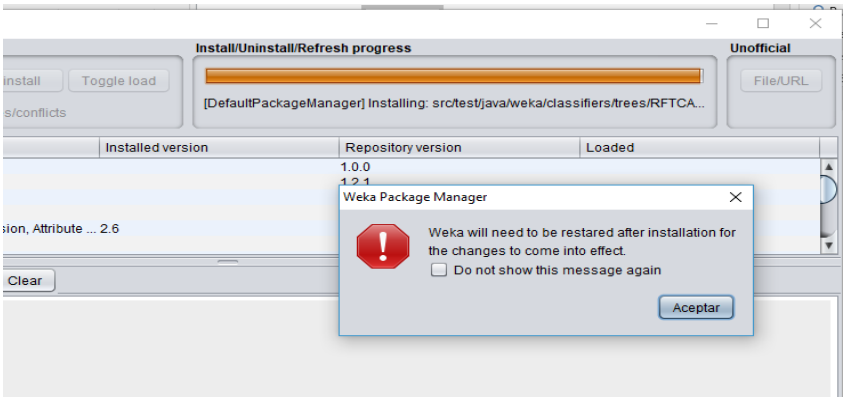

Figura 20. Paso 6 de Instalación de un paquete Weka

Tras iniciar de nuevo el software es posible verificar que en la categoría “trees", se encuentran instaladas ambas propuestas (ver **Figura 21**).

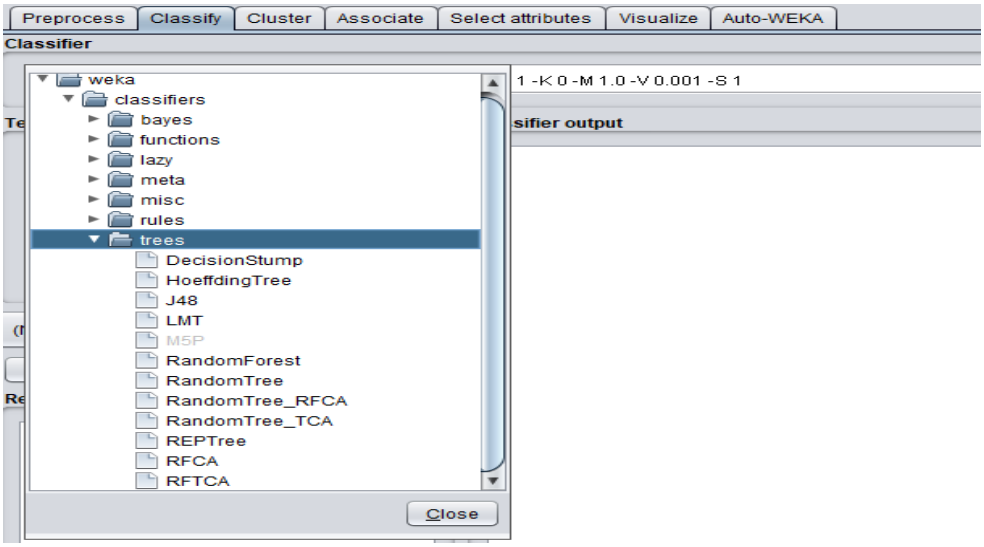

Figura 21. Paso 7 Verificación de Instalación de un paquete Weka

En la **Figura 22** se observa la manera de seleccionar un valor de fuerza (de 2 hasta 7) para la propuesta RFCA, en el caso de la propuesta RFTCA es posible seleccionar un valor de fuerza de 2 hasta 6. Dadas las características del software se optó por emplear el termino en inglés para fuerza (Strength), en la **Figura 23** se presenta la ejecución del algoritmo sobre el conjunto de datos Iris.

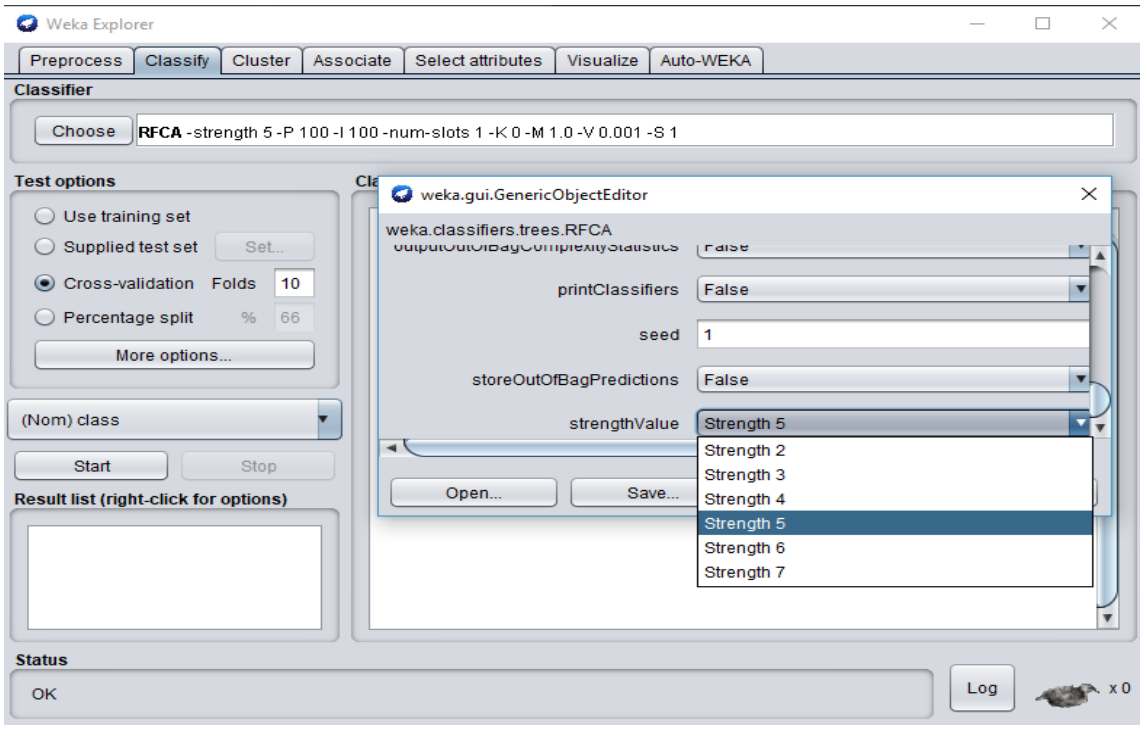

Figura 22. Paso 9 Verificación de Instalación de un paquete Weka

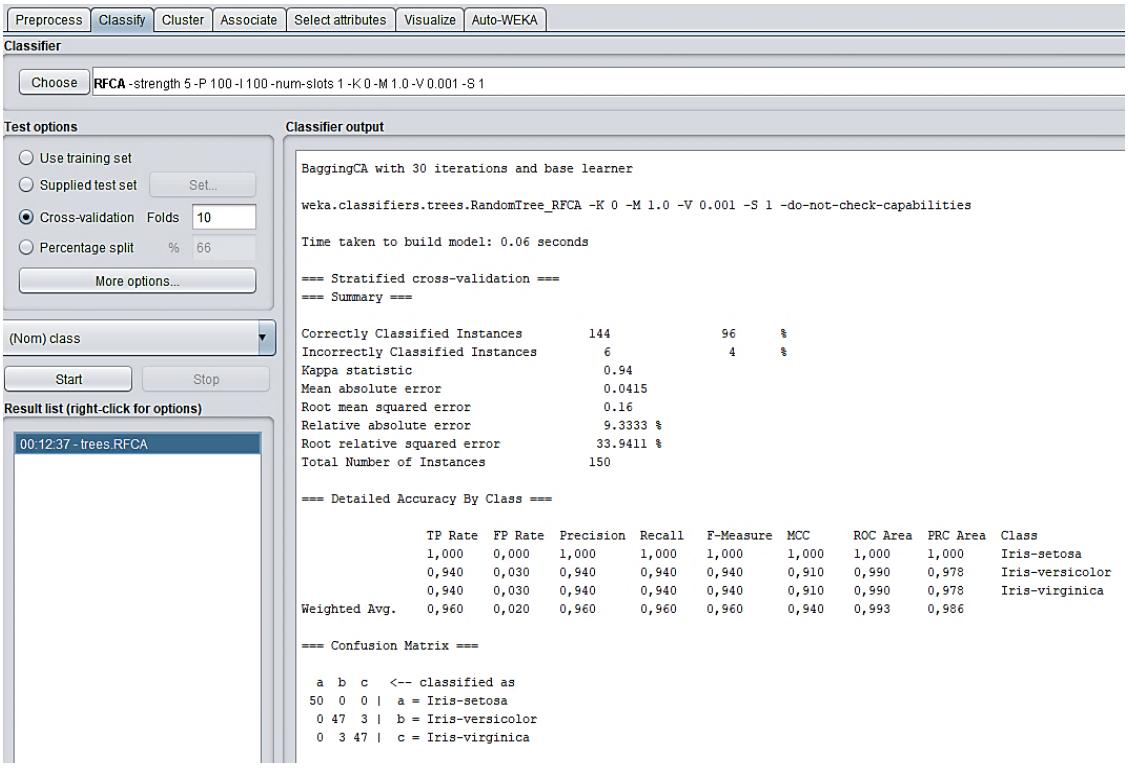

Figura 23. Ejecución de algoritmo RFCA sobre un conjunto de Datos

CAPÍTULO 5

5 EXPERIMENTACION

En este capítulo se presenta inicialmente las medidas de desempeño usadas para comparar los resultados de la experimentación. Después, se presenta la calidad de los resultados de exactitud de RF con diferente número de árboles. Luego de esto, se describen los conjuntos de datos de clasificación, utilizados en la evaluación y comparación de los algoritmos propuestos frente a los algoritmos del estado del arte. Seguidamente, se muestra el afinamiento de los parámetros empleados durante la experimentación, los resultados obtenidos usando validación cruzada de los modelos obtenidos, y el análisis de los resultados. En la parte final de esta sección se aplica minería de datos sobre los resultados obtenidos en la exactitud del algoritmo RFCA buscando identificar las características del conjunto de datos donde se obtienen resultados con mayor calidad al estado del arte. El análisis de resultados de la propuesta RFTCA (Random Forest con Torres de Covering Arrays) se encuentran en el **Anexo 3**, ya que no fue la mejor propuesta obtenida.

El algoritmo fue implementado como un paquete en el software de aprendizaje automático y minería de datos escrito en Java Weka 3.8. Los CAs binarios de fuerza 2 hasta fuerza 7 y los TCAs binarios de fuerza 2 hasta 6 fueron compartidos por el CINESTAV-Tamaulipas (México), la ejecución de los experimentos se realizó en un computador con un procesador Intel Core i7 4510U, 2.0 GHz, 8 GB RAM, Windows 10.

5.1 MEDIDAS DE DESEMPEÑO

La evaluación del desempeño de un modelo de clasificación permite comparar clasificadores de manera equitativa. Cada clasificador trabajo con un enfoque específico, en consecuencia, no se tiene un único clasificador que funcione mejor para los diferentes tipos de problemas de calificación [37]. Una de las medidas de desempeño más utilizada en modelos de clasificación es la exactitud (accuracy), calculada como el cociente entre el número de instancias de evaluación correctamente clasificadas y el número total de instancias del conjunto de datos de evaluación [38] [39]. Por otra parte, el error de clasificación se calcula como el cociente entre el número de instancias de evaluación incorrectamente clasificadas y el número total de instancias del conjunto de datos de evaluación [38].

La **ecuación (6)** [38], describe el cálculo del error de clasificación, la **ecuación (7)** [39], muestra el cálculo de la exactitud y la **ecuación (8)** expresa la relación entre ambas medidas de desempeño.

$$Error\ Clasificación = \frac{Instancias\ incorrectamente\ clasificadas}{Numero\ total\ de\ intancias} \quad (6)$$

$$Exactitud = \frac{Instancias\ correctamente\ clasificadas}{Numero\ total\ de\ intancias} \quad (7)$$

$$Exactitud = 1 - Error\ Clasificación \quad (8)$$

En [39] se señala que la medida de desempeño que se menciona con más frecuencia en la literatura referente a modelos de clasificación, es la exactitud expresada en porcentaje . Dado que la exactitud y el error de clasificación expresan un resultado semejante, uno en la manera de instancias correctas y el otro en incorrectas, se opta por el porcentaje de exactitud para la exposición y análisis de los resultados.

La segunda medida de desempeño empleada durante la exposición y análisis de los resultados es la Medida F (F measure). La medida F se calcula de acuerdo con la **ecuación (9)** [40], integra en su cálculo a la precisión (precisión) y el recuerdo (recall).

$$Medida\ F = 2\left(\frac{precisión * recuerdo}{precisión + recuerdo}\right) \quad (9)$$

La medida F tendrá un valor alto cuando tanto la precisión como el recuerdo tengan valores altos. La medida F puede verse como una manera de encontrar el mejor balance entre precisión y recuerdo [40].

5.2 NUMERO DE ARBOLES Y EXACTITUD EN RANDOM FOREST

Generalmente el aumento en el número de árboles en un Random Forest (RF) produce mejoras en la exactitud del modelo, no obstante, la exactitud puede verse afectada o invariable ante aumentos excesivos o no medidos en el número de árboles [10]. Además, si la cantidad de árboles es grande, la carga computacional para construir el modelo y predecir los registros puede ser muy alta, particularmente cuando se tienen gran cantidad de registros [41].

En la **Figura 24** se muestra la gráfica resultante de la evaluación en distintos conjuntos de datos del algoritmo Random Forest en relación con la exactitud obtenida usando distintos valores en el número de árboles.

En la **Figura 24** se observa en los conjuntos de datos Blood, Fertility y Vehicle como el aumento en el número de árboles puede generar una disminución en la exactitud del algoritmo Random Forest. También, se evidencia en los conjuntos de datos Leaf, Car e Indian como la exactitud varia poco ante el aumento del número de árboles empleado para construir el Random Forest. Finalmente, la exactitud en los conjuntos de datos Hayes y Haberman presenta un comportamiento anómalo, del cual no es viable deducir el impacto del número de árboles. Dado que la cantidad optima de árboles en un Random Forest puede variar entre los distintos conjuntos

de datos, es importante encontrar una manera adecuada de definir el parámetro número de árboles, evitando la prueba de distintos valores y obteniendo una exactitud óptima. En este sentido, la técnica propuesta permite seleccionar el valor del número de árboles por medio del número de filas del CA o TCA obtenido según el valor de fuerza.

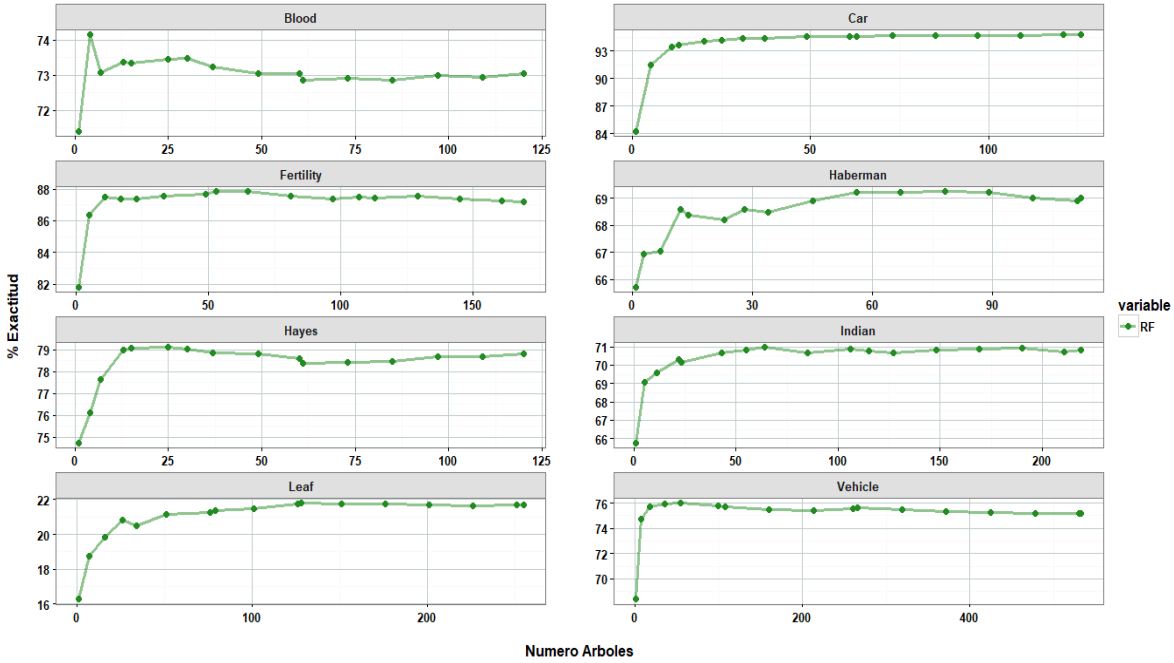

Figura 24. Numero de Árboles y exactitud en Random Forest

5.3 DISEÑO DE LOS EXPERIMENTOS

Inicialmente se definieron un total de 33 conjuntos de datos de clasificación disponibles en el repositorio de la UCI (University of California at Irvine), que representan problemas específicos del mundo real. Luego, se evaluaron 9 algoritmos Random Forest, donde 7 de los algoritmos surgen de la propuesta de Random Forest basado en Covering Arrays y 2 de los algoritmos corresponden al estado del arte. Posteriormente, se utilizan los test estadísticos no paramétricos de Friedman y Wilcoxon en la exactitud y medida F con el fin de determinar la mejor propuesta de Random Forest basado en Covering Arrays. En los 7 algoritmos del tipo Random Forest basado en Covering Arrays (RFCA y variaciones) el parámetro número de árboles toma como valor el número de filas del CA. De ahí que, para lograr una mayor consistencia en los resultados, el parámetro número de árboles (numIterations Weka) en RF y RF_SQRT se fija después de la ejecución de RFCA.

5.4 CONJUNTOS DE DATOS DE PRUEBA

Los 33 conjuntos de prueba empleados se presentan en la **Tabla 7** [20]. Para cada conjunto de datos se presenta el número de instancias de entrenamiento, número

de instancias de prueba, número de características, y número de clases en la característica objetivo.

Conjuntos de Datos	#Instancias Entrenamiento	#Instancias Prueba	#Características	#Clases
Banknote	922	450	4	2
Blood	500	248	4	2
Car	1152	576	6	4
Chart	400	200	60	6
Climate	540	360	18	2
Contraceptive	1000	473	9	3
Dermatology	244	122	34	6
Diabetes	576	192	8	2
Ecoli	224	112	7	8
Fertility	70	30	9	2
Glass	154	60	9	7
Haberman	204	102	3	2
Hayes	88	44	4	3
Indian	393	190	10	2
Ionosphere	234	117	34	2
Iris	100	50	4	3
Knowledge	172	86	5	4
Leaf	230	110	15	36
Libras	260	100	90	15
Planning	182	122	12	2
QSARBiodegradation	705	350	41	2
Seeds	140	70	7	3
Segment	1500	810	19	7
Sonar/Connectionist	138	70	60	2
Soy Bean	307	376	35	19
Spectf	178	89	44	2
Vowel	528	462	10	11
Vehicle	846	100	18	4
Wdbc	379	190	31	2
Wine	118	60	13	3
Wine Red	1066	533	11	6
Yeast	989	495	8	10
Zoo	70	31	17	7

Tabla 7. Descripción de los conjuntos de Datos

5.5 AFINAMIENTO DE PARÁMETROS

Para evaluar el algoritmo RFCA, se realizó una comparación con el algoritmo Random Forest original propuesto por Breiman [9] (RF) y una versión de Random Forest denominada RF_SQRT, en la que el tamaño K de los subconjuntos de características seleccionadas aleatoriamente se define como: $K = \sqrt[2]{P}$, donde P corresponde a el número de características del conjunto de datos sin incluir la característica objetivo o de clase [11][35].

Los valores de los parámetros comunes a los tres algoritmos son los designados por defecto en Weka (**Tabla 8**). El hiper parámetro K (numFeautres Weka) para RF

y RFCA se define según la propuesta de Breiman [9], como el primer entero menor que $\log_2 P + 1$. En RFCA no se define el parámetro M (número de árboles), pero si se define el parámetro fortaleza t (strength) del Covering Array (CA) o Torres de Coverng Array (TCA) binario que se empleará en la construcción del bosque. Este parámetro toma los valores de fuerza 2 hasta fuerza 7. Para realizar una comparación justa entre RFCA, RF y RF_SQRT se define en los dos últimos algoritmos el parámetro M (número de árboles) igual al que define RFCA con el número de filas del Covering Array.

Parámetros	Valor
bagSizePercent	100
batchSize	100
breakTiesRandomly	False
maxDepth	0
numExecutionSlots	1

Tabla 8. Parámetros del algoritmo por defecto en Weka

5.6 RESULTADOS OBTENIDOS Y COMPARACIÓN

Los resultados presentados a continuación, se obtuvieron a partir del promedio de 30 ejecuciones de cada algoritmo en cada conjunto de datos usando validación cruzada con una configuración de 10 folders. Cada una de las 30 ejecuciones mencionadas fue realizada aplicando una semilla (parámetro seed en Weka) diferente (de 1 a 30). Los CA se evaluaron usando fuerza 2 hasta 7, los TCA se evaluaron usando fuerza 2 hasta 6, todos con alfabeto binario. La evaluación se llevó a cabo con diferentes valores de fuerza con el fin de identificar cuál valor logra los mejores resultados en términos de exactitud, medida F y tiempos de ejecución. Cada conjunto de datos evaluado requirió de un CA o TCA definido en función del número de características en el conjunto (parámetro P en el CA o TCA). El número de árboles aumenta a medida que aumenta la fuerza (t).

Para tener una mayor comprensión de los resultados obtenidos por todos los algoritmos se utilizaron dos métodos de análisis estadístico mediante el software Keel [34], con el fin de obtener las pruebas estadísticas no paramétricas de Wilcoxon y Friedman.

5.6.1 Resultados RFCA y variaciones

5.6.1.1 Análisis estadístico de los algoritmos Random Forest basados en Covering arrays

Tras finalizar las 30 ejecuciones de cada algoritmo Random Forest basado en Covering Arrays (RFCA y variaciones propuestas), se realizaron las pruebas estadísticas no paramétricas de Friedman y Wilcoxon, para determinar cuál de los algoritmos en términos de exactitud y medida F integra mejor el mecanismo de selección de características para Random Forest con Covering Arrays. El test de

Friedman genera un ranking del desempeño de los algoritmos y asigna un menor valor al algoritmo que presente mejores resultados. El test de Wilcoxon presenta la tabla de dominancia de los algoritmos, en donde un punto negro indica que el algoritmo de la fila domina al algoritmo de la columna, un punto blanco indica que el algoritmo de la columna domina al de la fila y la casilla vacía indica que no es posible establecer una dominancia de un algoritmo sobre el otro. Los resultados por debajo de la diagonal tienen un nivel de significancia del 0.95 (95% confianza) y los que están por encima de 0.90 (90% confianza).

En el caso de la exactitud, la **Tabla 9** muestra el ranking del test de Friedman de los algoritmos basados en Covering Arrays y los algoritmos del estado del arte. En esta tabla se evidencia claramente que el mejor algoritmo basado en Covering Arrays es RFCA. El valor p obtenido para la prueba fue de 0.00881. Teniendo en cuenta que este valor es menor a 0.05, se considera válido estadísticamente.

Puesto	Algoritmo	Ranking
1	**RFCA**	**4.5833**
2	RFCA-VAR2	4.6338
3	RFCA-SQRT-VAR2	4.904
4	RFCA-VAR1	4.9242
5	RFCA-SQRT	4.9268
6	RFCA-SQRT-VAR1	5.1288
6	RF-SQRT	5.1288
7	RF	5.1439
8	RFCA-SINK	5.6263
Valor de P	0. 00881 (chi cuadrado con 8 grados de libertad: 20.4360)	

Tabla 9. Ranking con el resultado de exactitud en los algoritmos según Test de Friedman

La **Figura 25** muestra los resultados de la prueba de Wilcoxon para los resultados en la medida de exactitud. El propósito de la prueba es determinar la dominancia de los resultados de un algoritmo sobre los resultados de los demás. El mejor algoritmo determinado en el ranking del test Friedman (RFCA) numerado en la **Figura 25** como (2) domina a tres de las variaciones DE RFCA (RFCA_SINK, RFCA_VAR1 Y RFCA_SQRT_VAR1) y a los algoritmos del estado del arte (RF y RF_SQRT). Asimismo, el algoritmo que no emplea el parámetro K (RFCA_SINK) denotado como (1) en la **Figura 25** es dominado por RFCA y las variaciones propuestas, pero no es dominado por los algoritmos del estado del arte.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
RFCA-SINK (1)	-	○	○	○	○	○	○		
RFCA (2)	●	-	●			●		●	●
RFCA-VAR1 (3)	●		-	○					
RFCA-VAR2 (4)	●			-				●	●
RFCA-SQRT (5)	●				-	●			
RFCA-SQRT-VAR1 (6)	●				○	-	○		
RFCA-SQRT-VAR2 (7)	●					●	-		
RF (8)								-	
RF-SQRT (9)		○		○					-

Figura 25. Test Wilcoxon de los resultados de exactitud

La **Tabla 10** muestra el ranking del test de Friedman para los resultados obtenidos en la medida F de los algoritmos basados en Covering Arrays y los algoritmos del estado del arte. En la tabla se evidencia que el mejor algoritmo basado en Covering Arrays es RFCA. El valor p obtenido para la prueba fue 4.7434E-9. Luego, al ser este valor menor a 0.05, se considera válido estadísticamente.

Puesto	Algoritmo	Ranking
1	**RFCA**	**4.4091**
2	RFCA-VAR2	4.4167
3	RF-SQRT	4.5631
4	RF	4.5758
5	RFCA-SQRT-VAR2	5.2071
6	RFCA-VAR1	5.2475
7	RFCA-SQRT	5.298
8	RFCA-SQRT-VAR1	5.6086
9	RFCA-SINK	5.6742
p-value	4.7434E-9 (chi cuadrado con 8 grados de libertad: 54.86397)	

Tabla 10. Ranking con el resultado de medida F de los algoritmos según Test de Friedman

Los resultados del test de Wilcoxon para los resultados en la medida F se presentan en la **Figura 26.** A diferencia de los resultados obtenidos para la exactitud, la dominancia del mejor algoritmo para la medida F determinado en el ranking del test Friedman (RFCA) numerado en la **Figura 26** como (2), domina a las variaciones de RFCA con excepción de la variación RFCA-VAR2, no obstante, la dominancia es indeterminada con respecto a los algoritmos del estado del arte.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
RFCA-SINK (1)	-	○	○	○	○		○	○	○
RFCA (2)	●	-	●		●	●	●		
RFCA-VAR1 (3)		○	-	○		●		○	○
RFCA-VAR2 (4)	●		●	-	●	●	●		
RFCA-SQRT (5)		○		○	-	●	○	○	○
RFCA-SQRT-VAR1 (6)		○	○	○	○	-	○	○	○
RFCA-SQRT-VAR2 (7)	●	○		○		●	-	○	○
RF (8)	●				●	●	●	-	
RF-SQRT (9)	●		●		●	●	●		-

Figura 26. Test Wilcoxon de los resultados en Medida F

En general los resultados de las pruebas estadísticas no paramétricas de Friedman y Wilcoxon para la exactitud y medida F, permiten determinar que el algoritmo RFCA es el que mejor integra el mecanismo de selección de características basado Covering Array.

5.6.2 Resultados RFCA y Estado del Arte

En la siguiente sección se analizan los resultados del algoritmo RFCA, los resultados para la exactitud, medida F y tiempo de ejecución presentados a continuación se obtuvieron a partir del promedio de las 30 ejecuciones de cada algoritmo en cada conjunto de datos usando validación cruzada con una

configuración de 10 folders, evaluando Covering Arrays con alfabeto binario de fuerza 2 hasta 7.

5.6.2.1 Resultados y análisis para la exactitud en RFCA

Los resultados en términos de exactitud presentados en las **figuras 27** y **28** son los conjuntos de datos donde la propuesta RFCA obtiene mejores resultados con respecto a los algoritmos del estado del arte.

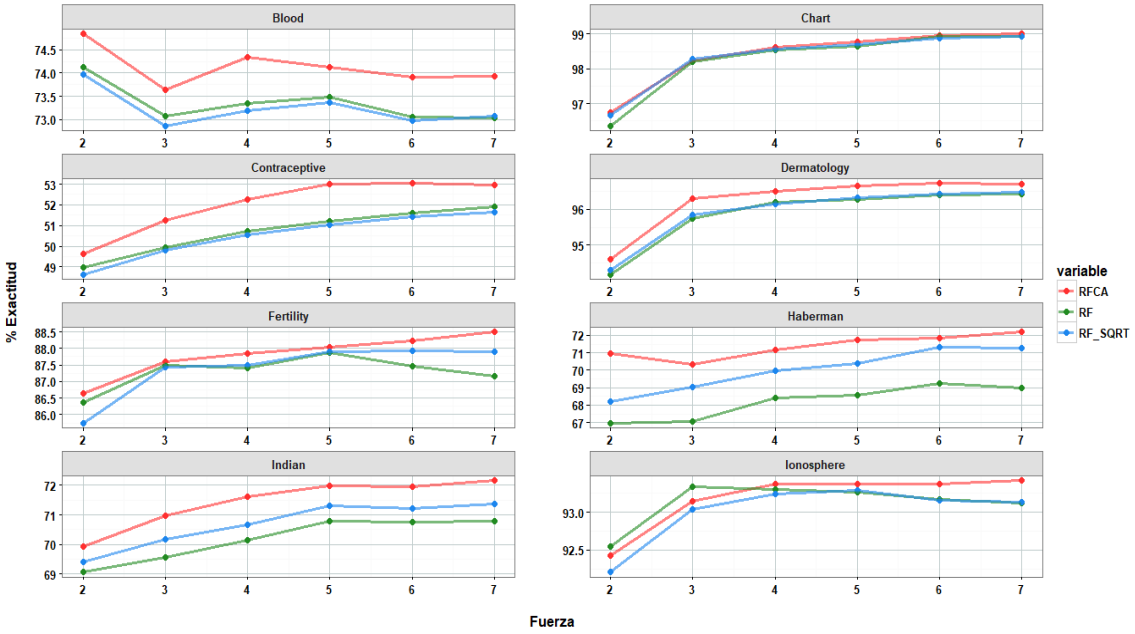

Figura 27. Parte 1 de los resultados más altos en términos de Exactitud para la propuesta RFCA

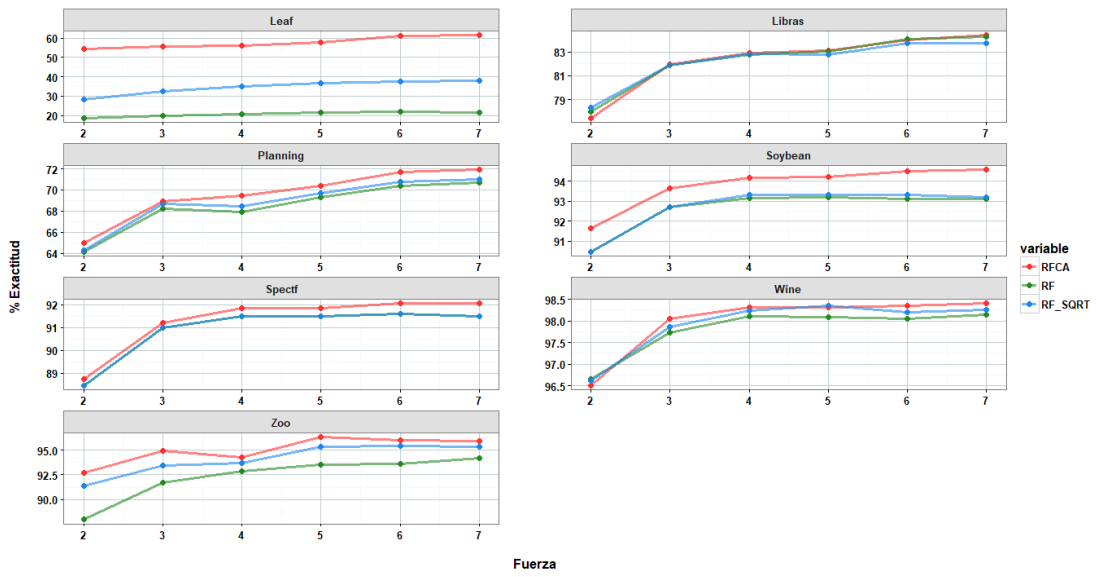

Figura 28. Parte 2 de los resultados más altos en términos de Exactitud para la propuesta RFCA

La **Figura 29** corresponde a los conjuntos de datos donde la propuesta RFCA obtiene resultados similares a los del estado de arte.

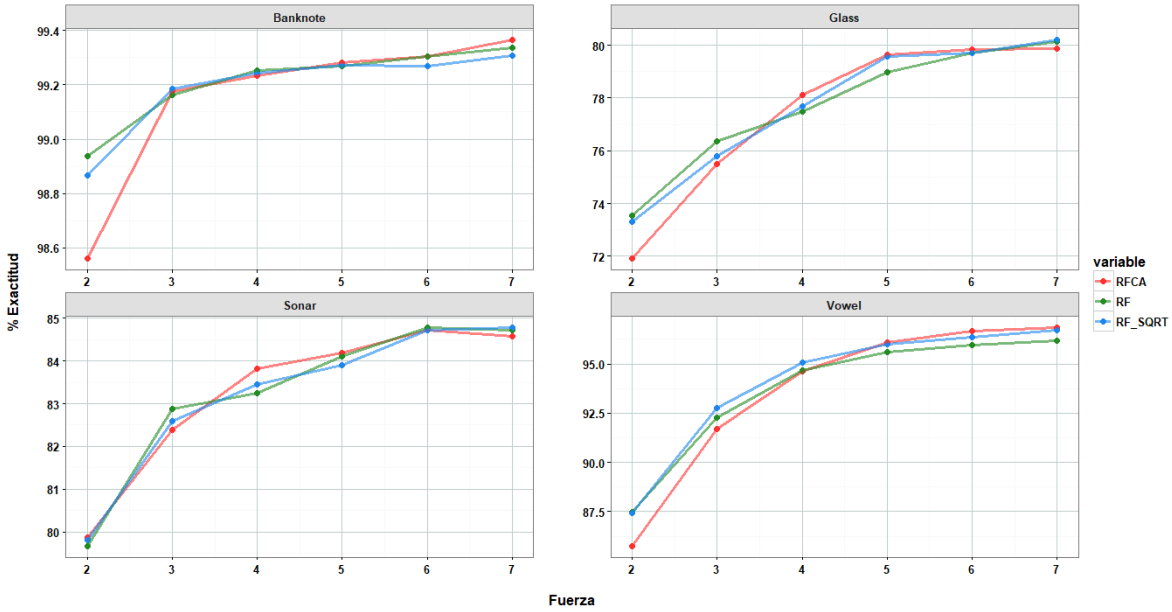

Figura 29. Resultados en términos de Exactitud donde la propuesta RFCA obtiene resultados similares a los del Estado de Arte

Las **figuras 30 y 31** son los conjuntos de datos donde la propuesta RFCA obtiene resultados más bajos.

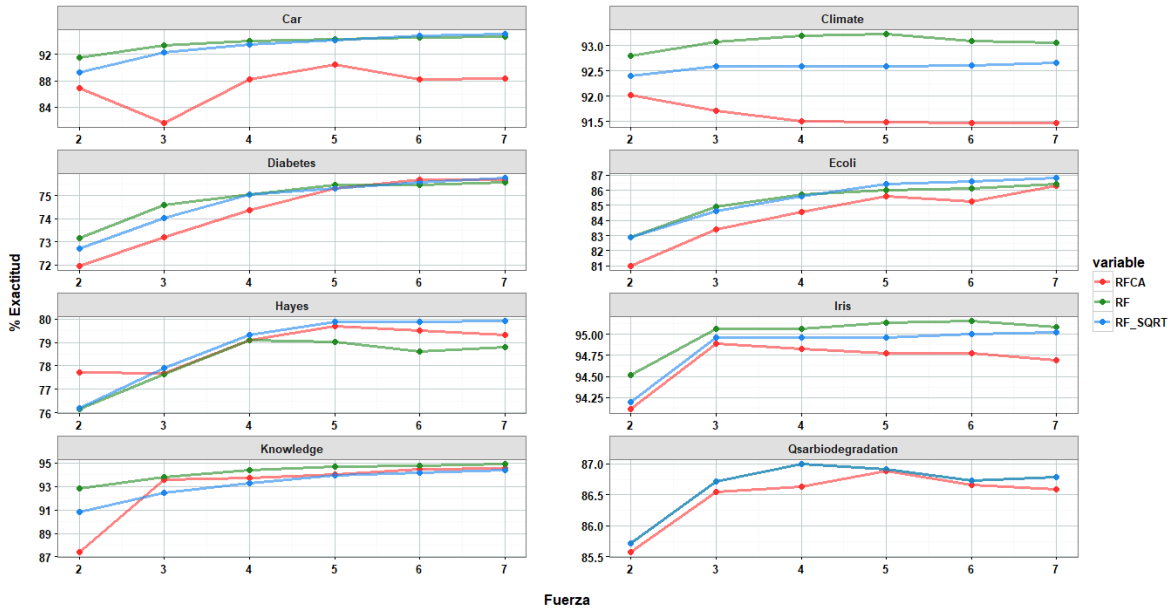

Figura 30. Parte 1 de los resultados más bajos en términos de Exactitud para la propuesta RFCA

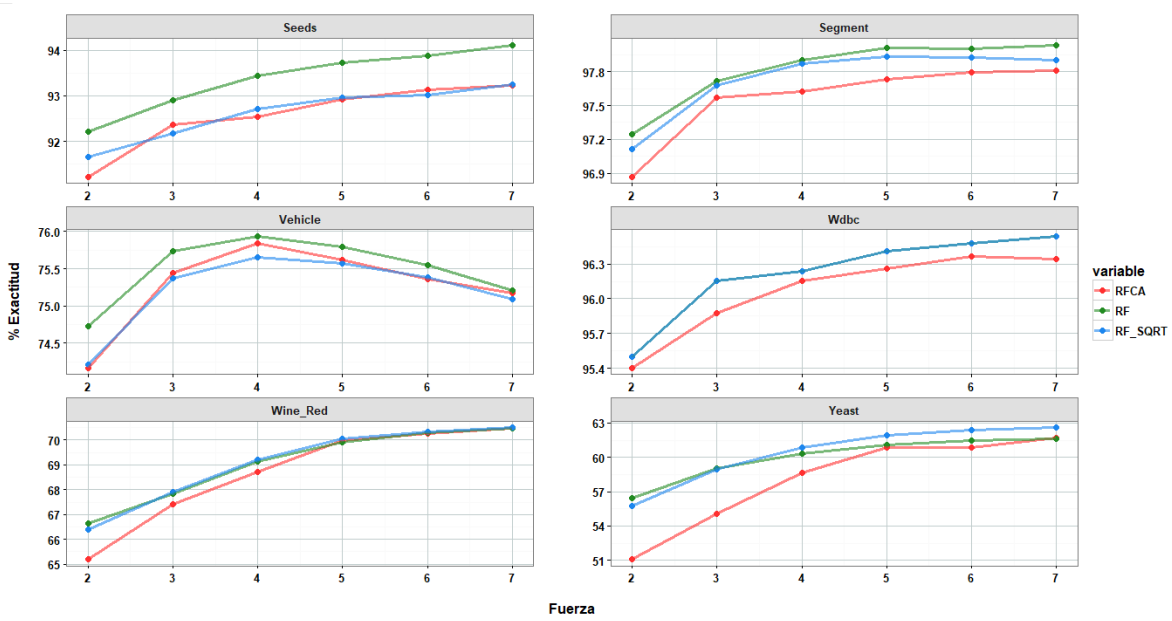

Figura 31. Parte 2 de los resultados más bajos en términos de Exactitud para la propuesta RFCA

En total se obtuvieron 198 escenarios (33 conjuntos de datos por 6 Valores de Fuerza del CA) de evaluación del algoritmo propuesto con respecto a los del estado de arte. En la **Tabla 11** se resumen los resultados para cada valor de fuerza de los 33 conjuntos de datos, de los cuales se indica el número y porcentaje de conjuntos donde un algoritmo supera o empata a otro, en cuanto al número total de escenarios(198) se observa que el algoritmo RFCA se desempeña mejor en aproximadamente la mitad de los escenarios de evaluación, el porcentaje restante del total de escenarios se reparte entre los dos algoritmos del estado del arte.

Fuerza	Gana			Empata		# Conjuntos de Datos
	RFCA	RF	RF SQRT	RF y RF_SQRT	RFCA y RF	
2	14 (42,42%)	15 (45,45%)	1 (3,03%)	3 (9,09%)	0	33
3	13 (39,39%)	13 (39,39%)	5 (15,15%)	2 (6,06%)	0	33
4	17 (51,52%)	9 (27,27%)	5 (15,15%)	2 (6,06%)	0	33
5	18 (54,55%)	8 (24,24%)	5 (15,15%)	2 (6,06%)	0	33
6	17 (51,52%)	8 (24,24%)	5 (15,15%)	2 (6,06%)	1 (3,03%)	33
7	17 (51,52%)	6 (18,18%)	8 (24,24%)	2 (6,06%)	0	33
# Total Escenarios	96 (48,48%)	59 (29,79%)	29 (14,64%)	13 (6,56%)	1 (0,50%)	198

Tabla 11. Resumen de resultados para exactitud en RFCA y RF Y RF_SQRT

La **Tabla 12** muestra el resumen de resultados tras contrastar la propuesta RFCA con el algoritmo referencia de Breiman (RF).

La propuesta RFCA supera a RF con los valores de fuerza 4, 5, 6 y 7, destacando principalmente los resultados para fuerza 5 (63,64%) y 7 (60,61%). En cuanto al número total de escenarios (198), el algoritmo RFCA se desempeña mejor en 107

(54,04%) de los escenarios de evaluación, el porcentaje restante del total de escenarios se reparte entre los dos algoritmos del estado del arte.

Fuerza	Gana		Empata	# Conjuntos de Datos
	RFCA	RF	RFCA y RF	
2	14 (42,42%)	19 (57,58%)	0 (0%)	33
3	16 (48,48%)	17 (51,52%)	0 (0%)	33
4	18 (54,55%)	15 (45,45%)	0 (0%)	33
5	21 (63,64%)	12 (36,36%)	0 (0%)	33
6	18 (54,55%)	14 (42,42%)	1 (3,03%)	33
7	20 (60,61%)	13 (39,39%)	0 (0%)	33
# Total Escenarios	107 (54,04%)	90 (45,45%)	1 (0.50%)	198

Tabla 12. Resumen de resultados para exactitud en RFCA y RF

La **Tabla 13** muestra el resumen de resultados tras contrastar la propuesta RFCA con el algoritmo RF_SQRT.

Fuerza	Gana		Empata	# Conjuntos de Datos
	RFCA	RF_ SQRT	RFCA y RF_SQRT	
2	15 (45,45%)	18 (54,54%)	0 (0%)	33
3	17 (51,52%)	16 (48,48%)	0 (0%)	33
4	19 (57,58%)	14 (42,42%)	0 (0%)	33
5	20 (60,61%)	13 (39,39%)	0 (0%)	33
6	21 (63,64%)	12 (36,36%)	0 (0%)	33
7	19 (57,58%)	14 (42,42%)	0 (0%)	33
# Total Escenarios	111 (56,06%)	87 (43,94%)	0 (0%)	198

Tabla 13. Resumen de resultados para exactitud en RFCA y RF_SQRT

Al observar para cada fuerza el número o porcentaje de escenarios en que un algoritmo supera o empata a otro, se determina que, con en cinco de los seis valores de fuerza experimentados, el algoritmo RFCA supera a RF-SQRT, en especial, al utilizar fuerza 5 (60,61%) y 6 (63,64%). Tomando en cuenta el número total de escenarios (198), el algoritmo RFCA se desempeña mejor en 111 (56,06%) de los escenarios de evaluación, el porcentaje restante del total de escenarios se reparte entre los dos algoritmos del estado del arte.

De acuerdo con los resultados de la **Tabla 11**, **Tabla 12** y **Tabla 13** se deduce que el valor de fuerza que mejores resultados obtiene en términos de exactitud en el algoritmo RFCA es la fuerza 5. Asimismo, el valor de fuerza que obtiene los resultados más bajos en el algoritmo RFCA es la fuerza 2.

Por otra parte, RFCA supera a RF con los valores de fuerza 4, 5, 6 y 7, destacando principalmente los resultados para fuerza 5 y 7. Por consiguiente, de acuerdo a los resultados de la **Tabla 11** se deduce que el valor de fuerza que mejores resultados obtiene en términos de exactitud en el algoritmo RFCA es la fuerza 5.

Tomando los resultados del test estadístico no paramétrico de Friedman se obtiene el ranking de la **Tabla 14**, en el que se confirma el mejor desempeño de RFCA con respecto a RF y RF_SQRT, aunque, el valor p de la prueba no fue menor que 0.05, lo que significa que los resultados no son estadísticamente significativos. No obstante, el test de Wilcoxon para los resultados en la exactitud presentados en la **Figura 32** exponen con un 90% de confianza, que el algoritmo RFCA domina a RF y RF_SQRT, y en un 95% de confianza, que el algoritmo RFCA domina a RF_SQRT**.**

Algoritmo	Ranking
RFCA	**1.8965 (1)**
RF-SQRT	2.0278 (2)
RF	2.0758 (3)
Valor de p	0.18162 (chi-cuadrado con 2 grados de libertad: 3.411616)

Tabla 14. Test de Friedman para exactitud en RFCA y RF Y RF-SQRT

	(1)	(2)	(3)
RFCA (1)	-	●	●
RF (2)		-	
RF-SQRT (3)	○		-

Figura 32. Test de Wilcoxon para exactitud en RFCA y RF Y RF-SQRT

En los resultados obtenidos destacan principalmente los correspondientes a los conjuntos de datos Car y Leaf. En Car, RF supera a RFCA por un amplio margen, mientras que en Leaf, RFCA supera por una ventaja considerable a RF. Al revisar la estructura de las características del conjunto de datos Car se encuentra que está constituido por 6 características ordinales, 4 clases y 1728 instancias, datos que son similares a los de otros conjunto de datos y por ello no se logró identificar porque en este caso RF supera por mucho a RFCA (ver el lado izquierdo de la **Figura 33**).

En cuanto a Leaf se encuentra que está constituido por 14 características continuas, 1 nominal, 36 clases y 340 instancias. Aunque no es concluyente, el alto número de clases de este conjunto de datos puede verse beneficiado del análisis de las interacciones de las columnas que se realiza con el CA. En este conjunto de datos, desde el valor más bajo de fuerza (2) hasta el más alto (7) se presenta una considerable diferencia de RFCA respecto a RF y RF_SQRT (ver el lado derecho de la **Figura 33**).

En la **Figura 34** se muestra la gráfica promedio de todas las fuerzas, en la gráfica se evidencia la perturbación causada por los resultados atípicos de los conjuntos de datos Car y Leaf. Asimismo, se evidencia que el aumento gradual en la fuerza del CA genera en promedio un mejor resultado en términos de exactitud.

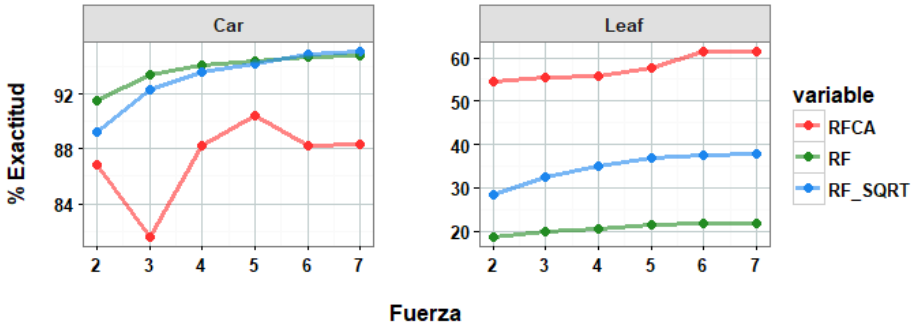

Figura 33. Exactitud en Conjuntos de Datos Car y Leaf

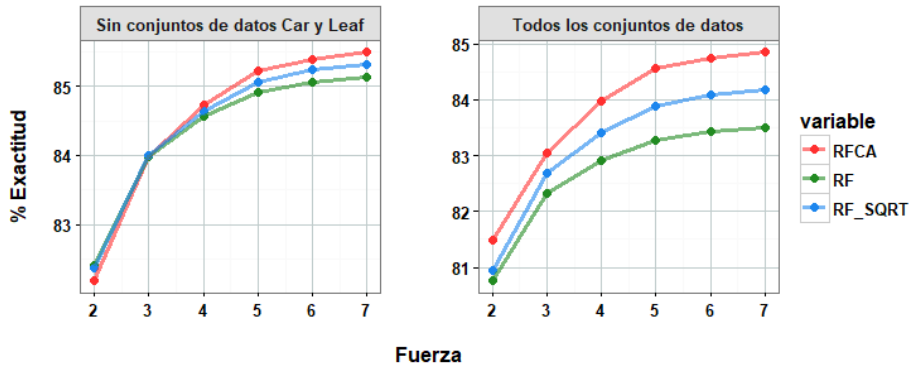

Figura 34. Promedio de Exactitud sobre todas las fuerzas

5.6.2.2 Resultados y análisis para la medida F en RFCA

Los resultados en términos de medida F presentados en las **figuras 35** y **36** son los conjuntos de datos donde la propuesta RFCA obtiene mejores resultados con respecto a los algoritmos del estado del arte.

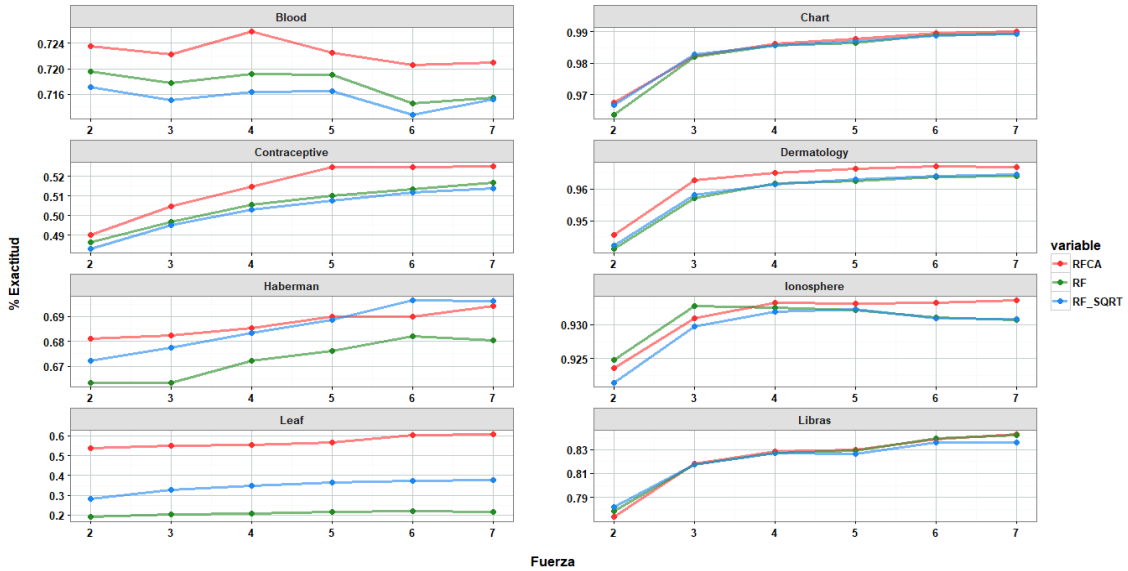

Figura 35. Parte 1 de los resultados más altos en términos de Medida F para la propuesta RFCA

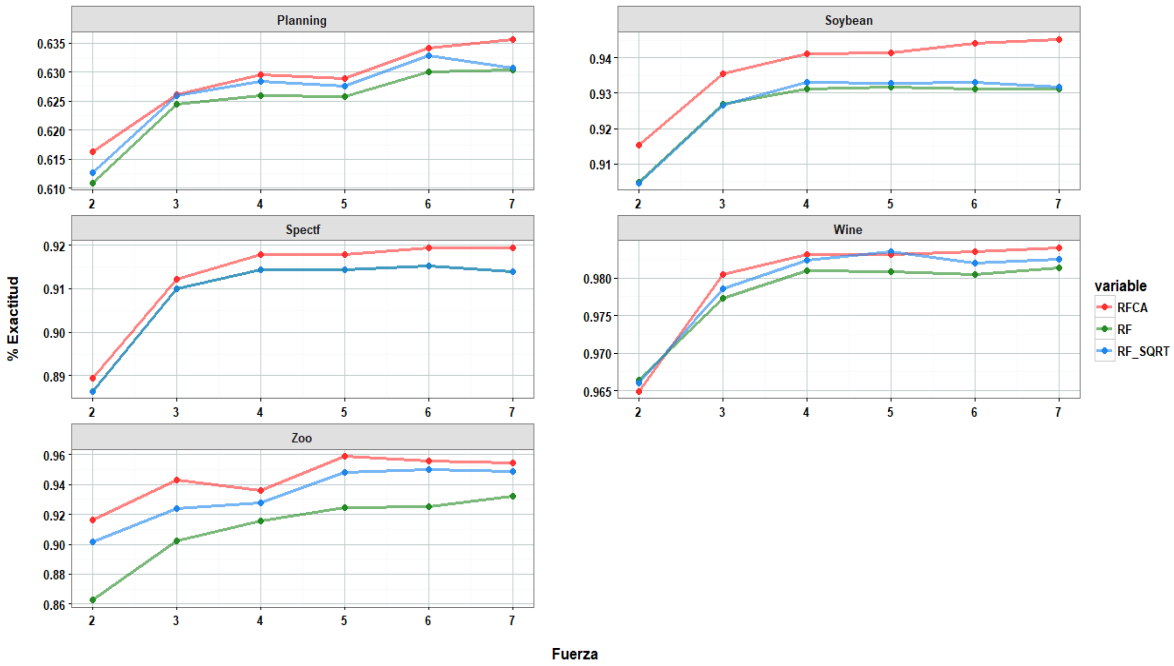

Figura 36. Parte 2 de los resultados más altos en términos de Medida F para la propuesta RFCA

La **Figura 37** corresponde a los conjuntos de datos donde la porpuesta RFCA obtiene resultados similares a los del estado del arte

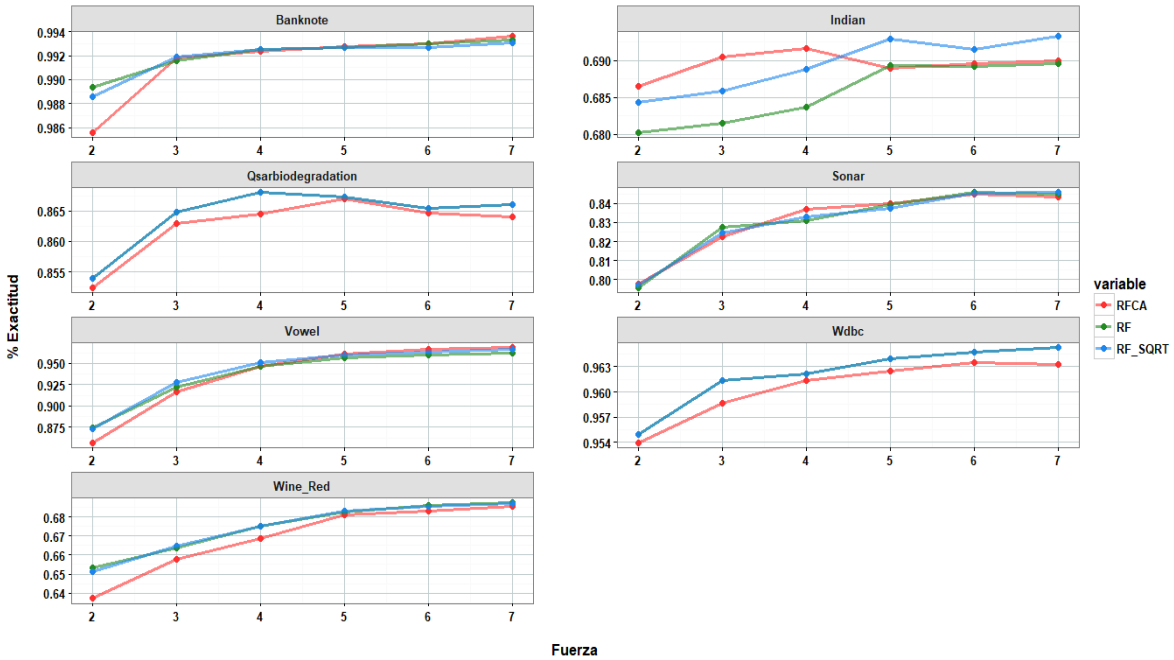

Figura 37. Resultados en términos de Medida F donde la propuesta RFCA obtiene resultados similares a los del Estado de Arte

Las **figuras 38** y **39** son los conjuntos de datos donde la propuesta RFCA obtiene resultados más bajos

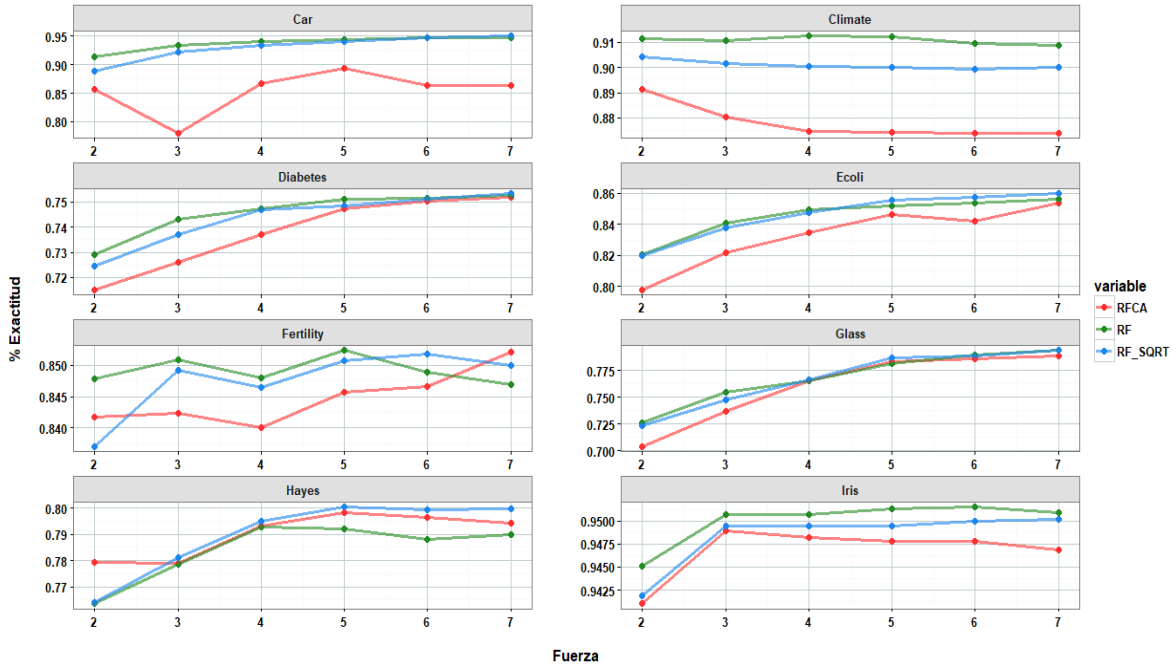

Figura 38. Parte 1 de los resultados más bajos en términos de Medida F para la propuesta RFCA

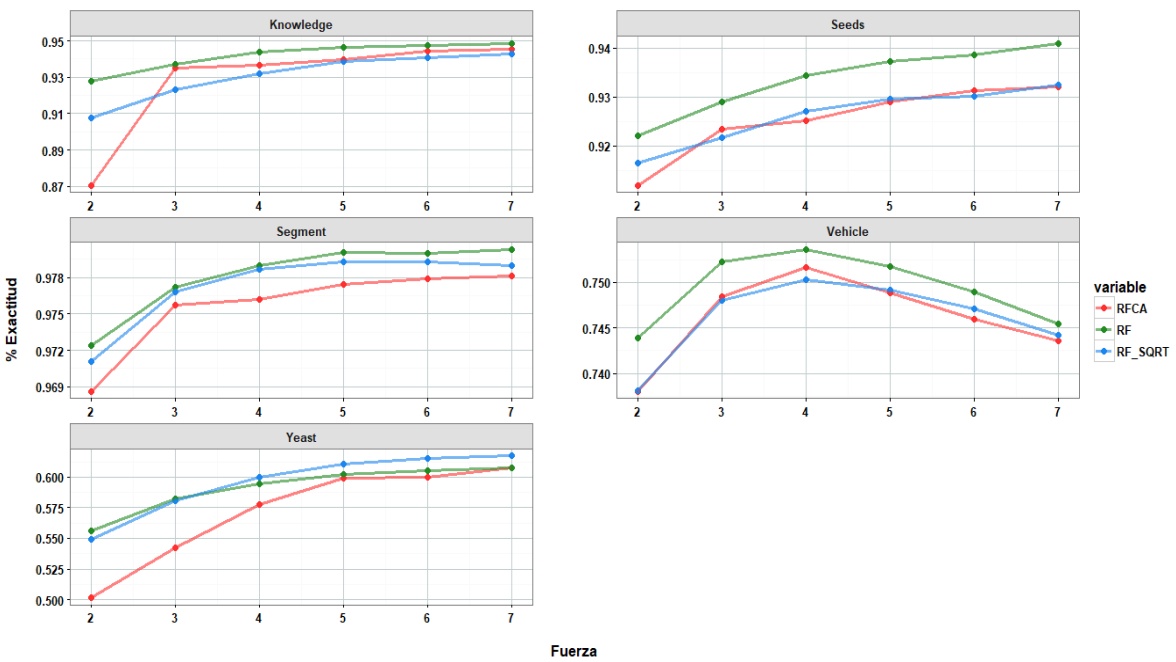

Figura 39. Parte 2 de los resultados más bajos en términos de Medida F para la propuesta RFCA

En total se tienen 198 escenarios (33 conjuntos de datos por 6 Valores de Fuerza del CA) de evaluación del algoritmo propuesto con respecto a los del estado de arte.

La **Tabla 15** muestra para cada valor de fuerza se tienen 33 conjuntos de datos de los cuales se indica el número y porcentaje de conjuntos donde un algoritmo supera o empata a otro ,en cuanto al número total de escenarios(198) se observa que el algoritmo RFCA se desempeña mejor en 41,41% de los escenarios de evaluación, el porcentaje restante del total de escenarios se reparte entre los dos algoritmos del estado del arte.

Fuerza	Supera			Empata		# Conjuntos de Datos
	RFCA	RF	RF SQRT	RF y RF_SQRT	RFCA y RF	
2	13 (39,39%)	17 (51,52%)	1 (3,03%)	2 (6,06%)	0	33
3	12 (36,36%)	14 (42,42%)	5 (15,15%)	2 (6,06%)	0	33
4	15 (45,45%)	10 (30,30%)	5 (15,15%)	3 (9,09%)	0	33
5	15 (45,45%)	9 (27,27%)	7 (21,21%)	2 (6,06%)	0	33
6	12 (36,36%)	11 (33,33%)	7 (21,21%)	2 (6,06%)	1 (3,03%)	33
7	15 (45,45%)	7 (21,21%)	8 (24,24%)	3 (9,09%)	0	33
# Total Escenarios	82 (41,41%)	68 (34,34%)	33 (16,67%)	14 (7,07%)	1 (0,51%)	198

Tabla 15: Resumen de resultados para Medida F en RFCA y RF Y RF-SQRT

Al observar para cada fuerza el número o porcentaje de escenarios en que un algoritmo supera o empata a otro, se determina que, para la medida F en los seis valores de fuerza experimentados, el algoritmo RFCA supera a RF-SQRT, en especial, al utilizar fuerza 4 y 5 en RFCA. Por otra parte, RFCA supera a RF con los valores de fuerza 4, 5 y 7, destacando principalmente los resultados para fuerza 5 y 7. Por consiguiente, de acuerdo a los resultados de la **Tabla 15** se deduce que para los resultados en Medida F, el valor de fuerza que mejores resultados obtiene en el algoritmo RFCA es la fuerza 5.

Tomando los resultados del test estadístico no paramétrico de Friedman en los resultados de medida F se obtiene el ranking de la **Tabla 16**, donde se confirma el mejor desempeño de RFCA con respecto a RF y RF_SQRT, aunque, el valor p de la prueba no fue menor que 0.05, lo que significa que los resultados no son estadísticamente significativos. La **Figura 40** correspondiente a los resultados del test de Wilcoxon, muestran que no se presenta dominancia de un algoritmo respecto a otro de los algoritmos.

Algoritmo	Ranking
RF	**1.9773 (1)**
RF-SQRT	2.0000 (2)
RFCA	2.0227 (3)
Valor de p	0.90278 (chi-cuadrado con 2 grados de libertad: 0.204545)

Tabla 16. Test de Friedman para Medida F en RFCA y RF Y RF-SQRT

	(1)	(2)	(3)
RFCA (1)	-		
RF (2)		-	
RF-SQRT (3)			-

Figura 40. Test de Wilcoxon para medida F en RFCA y RF Y RF-SQRT

5.6.2.3 Resultados y análisis de tiempo de ejecución

Los resultados de tiempos de ejecución se obtuvieron conforme al procedimiento descrito en la sección 5.6. El tiempo de ejecución de un algoritmo se considera desde el instante que comienza el proceso de creación del modelo de clasificación (RFCA y algoritmos del estado del arte) y finaliza cuando se completa la construcción del modelo.

La **Tabla 17** muestra el tiempo total de ejecución (suma de los tiempos de ejecución para los 6 valores de fuerza), en milisegundos (Mili-Seg) y segundos (Seg) con cada algoritmo para cada conjunto datos.

Conjuntos de Datos	RFCA (Mili-Seg)	RFCA (Seg)	RF (Mili-Seg)	RF (Seg)	RF_SQRT (Mili-Seg)	RF_SQRT (Seg)
Banknote	386,40	0,387	334,73	0,335	267,97	0,268
Blood	300,43	0,300	285,27	0,285	219,60	0,220
Car	134,70	0,135	167,40	0,167	186,13	0,186
Chart	7017,83	7,018	7060,23	7,060	7977,47	7,978
Climate	1116,13	1,116	960,37	0,960	855,10	0,855
Contraceptive	672,63	0,673	900,43	0,901	808,23	0,808
Dermatology	668,97	0,669	576,30	0,576	607,87	0,608
Diabetes	621,87	0,622	572,67	0,573	358,10	0,358
Ecoli	213,07	0,213	165,23	0,165	135,70	0,136
Fertility	63,73	0,064	44,70	0,045	43,17	0,043
Glass	248,20	0,248	222,13	0,222	181,60	0,182
Haberman	75,67	0,076	58,37	0,058	50,17	0,050
Hayes	34,33	0,034	23,43	0,023	24,67	0,025
Indian	693,30	0,693	629,10	0,629	531,10	0,531
Ionosphere	1798,57	1,799	1723,87	1,724	1534,30	1,534
Iris	39,30	0,039	26,13	0,026	20,93	0,021
Knowledge	96,83	0,097	73,10	0,073	69,43	0,069
Leaf	1052,87	1,053	988,73	0,989	786,00	0,786
Libras	8510,17	8,510	8226,80	8,227	9936,33	9,936
Planning	216,00	0,216	198,13	0,198	161,67	0,162
QSARBiodegradation	6008,53	6,009	5819,33	5,819	5828,60	5,829
Seeds	67,23	0,067	54,97	0,055	44,20	0,044
Segment	5696,33	5,696	5090,93	5,091	4358,60	4,359
Sonar/Connectionist	2035,10	2,035	1899,37	1,899	2119,23	2,119
Soy Bean	3277,43	3,278	4828,30	4,828	4484,70	4,485

Spectf	1932,23	1,932	1865,77	1,866	1870,53	1,871
Vowel	2474,90	2,475	2302,50	2,303	1931,43	1,932
Vehicle	1858,50	1,859	1664,27	1,664	1319,77	1,320
Wdbc	1556,90	1,557	1522,87	1,523	1525,37	1,525
Wine	122,93	0,123	106,97	0,107	88,83	0,089
Wine Red	2430,73	2,431	2325,30	2,325	1852,70	1,853
Yeast	1493,53	1,494	1400,73	1,401	888,13	0,888
Zoo	72,97	0,073	58,10	0,058	59,00	0,059
Tiempo total de ejecución	52988,31	52,991	52176,53	52,175	51126,63	51,129

Tabla 17. Resultados para tiempo de ejecución

El comportamiento en los resultados del tiempo de ejecución muestra la influencia de aspectos como el número de instancias y numero de características, de un conjunto de datos sobre el tiempo de ejecución, tal influencia resulta evidente en los tiempos de ejecución de los conjuntos de datos: Chart, Libras, QSARBiodegradation, Segment y Soy Bean, los cuales tuvieron los valores más altos. También, mediante la **ecuación (10)** se evaluó el porcentaje de diferencia en términos de tiempo total de ejecución en segundos, entre los algoritmos RFCA y los del estado del arte.

$$Porcentaje\ Diferencia\ = \left(\frac{R_r - R_e}{R_r}\right) * 100\% \qquad (10)$$

Donde R_r es el resultado de referencia y R_e el resultado a evaluar.

El porcentaje de diferencia del algoritmo RFCA con respecto al algoritmo de referencia de Breiman (RF).

$$Porcentaje\ Diferencia\ = \left(\frac{52{,}175 - 52{,}991}{52{,}175}\right) * 100\% = -1{,}56\%$$

El porcentaje de diferencia del algoritmo RFCA con respecto al algoritmo RF_SQRT

$$Porcentaje\ Diferencia\ = \left(\frac{51{,}129 - 52{,}991}{51{,}129}\right) * 100\% = -3{,}64\%$$

Los valores en el porcentaje de diferencia del Algoritmo RFCA con respecto al algoritmo referencia de Breiman (RF) y RF_SQRT fueron de $-1{,}56\%$ y $-3{,}64\%$ respectivamente, con lo cual se deduce que el algoritmo propuesto no genera un aumento significativo en el tiempo de ejecución, incluso en conjuntos como Car, Chart, Contraceptive y Soy Bean obtuvo tiempos de ejecución más bajos. En general, los resultados de los tres algoritmos, en términos de tiempo de ejecución son cercanos, y el aumento o disminución de tiempo en los resultados, depende más de los aspectos propios del conjunto de datos, como los son: número de clases, número de instancias, numero de características y tipos de características (discretas, continuas, nominales, ordinales).

5.7 MINERIA DE DATOS EN EXACTITUD

Con el fin de identificar el comportamiento del algoritmo RFCA con respecto al algoritmo de referencia de Breiman [9], se crearon, seis vistas minables (una para cada fuerza) con los distintos conjuntos de datos empleados en la experimentación. Los dos valores posibles para la clase de la vista minable son: Gana (G) o Pierde (P). El valor de la clase en una instancia será G cuando la diferencia de la exactitud entre RFCA y RF sea mayor a cero y P cuando la diferencia sea menor o igual a cero. En total son 6 vistas minables y cada vista minable contiene 10 valores de características de información para los 33 conjuntos de datos. Las características utilizadas son:

- Numero de Arboles: Es el número de árboles empleados en la construcción del modelo. Esté valor corresponde al número de filas del CA y varía según el valor de fuerza seleccionado para el CA.
- Numero Características: Es el número total de características del conjunto de Datos.
- Numero Instancias: Es la suma del número de instancias de entrenamiento y prueba.
- Numero de Clases: Es el número de posibles valores para la clase en un conjunto de Datos.
- Características Numéricas Discretas: Es el total de características representadas como números enteros. Por ejemplo, la edad de un paciente.
- Características Numéricas Continuas: Es el total de características representadas por números reales o de punto flotante. Por ejemplo, la altura en cm de un edificio.
- Características Categóricas Nominales: Es el total de características que representan categorías sin un orden significativo. Por ejemplo, el estado civil (casado, soltero, unión libre).
- Características Categóricas Ordinales: Es el total de características que representan categorías con un orden definido. Por ejemplo, el resultado de una competencia (primer, segundo o tercer lugar).
- Valores Faltantes: Los dos valores posibles para esta característica de la vista minable son: SI y NO. El valor SI indica que en el conjunto de datos hay instancias en las cuales una o más de sus características no tienen ningún valor [42]. NO indica que se tienen ingresados valores para todas las características de las instancias.
- Desbalanceado: Se tienen dos valores posibles para esta característica de la vista minable: SI y NO. El valor SI indica que, en el conjunto de datos, el número total de instancias que perteneces a una clase es mucho menor que el número total de instancias pertenecientes a otra de las clases del conjunto de datos [43]. NO indica que el número de instancias de cada clase es similar.

A partir de cada vista minable correspondiente a un valor de fuerza, se creó un modelo de árbol de decisión con el software de análisis y minería de datos Rapid Miner [44].

La **Figura 41** muestra el árbol de decisión obtenido en la vista minable de los resultados en exactitud con fuerza 2.

```
Caracteristicas Categoricas Nominales > 0.500
|   Numero de Instancias > 791.500: P {P=1, G=1}
|   Numero de Instancias ≤ 791.500: G {P=0, G=8}
Caracteristicas Categoricas Nominales ≤ 0.500
|   Numero Caracteristicas > 42.500
|   |   Numero de Instancias > 313.500: P {P=1, G=1}
|   |   Numero de Instancias ≤ 313.500: G {P=0, G=2}
|   Numero Caracteristicas ≤ 42.500
|   |   Desbalanceado = NO: P {P=12, G=0}
|   |   Desbalanceado = SI
|   |   |   Numero de Clases > 3: P {P=5, G=0}
|   |   |   Numero de Clases ≤ 3: G {P=0, G=2}
```

Figura 41. Árbol de decisión para vista minable en fuerza 2

Del árbol de decisión resultante para fuerza 2 se deduce principalmente:

- En conjuntos de datos con una o más características categóricas nominales y un número de instancias menor o igual a 791, es preferible ejecutar la propuesta RFCA, ya que la probabilidad de que RFCA obtenga mejores resultados es 100% (8 de 8 casos donde se presentó la regla).
- Para conjuntos de datos sin características categóricas nominales, con un número total de características menor o igual a 42, desbalanceado y con más de 3 clases, es preferible la ejecución del RF de Breiman, puesto que la probabilidad de que RF obtenga mejores resultados es 100% (5 de 5 casos donde se presentó la regla).
- En conjuntos de datos sin características categóricas nominales, con un número total de características menor o igual a 42 y NO desbalanceado, es preferible la ejecución del RF de Breiman, la probabilidad de que RF obtenga mejores resultados es 100% (12 de 12 casos donde se presentó la regla).

La **Figura 42** muestra el árbol de decisión obtenido en la vista minable de los resultados en exactitud con fuerza 3.

```
Caracteristicas Categoricas Nominales > 1.500: G {G=5, P=0}
Caracteristicas Categoricas Nominales ≤ 1.500
|   Numero de Clases > 13: G {G=2, P=0}
|   Numero de Clases ≤ 13
|   |   Valores Faltantes = NO
|   |   |   Numero Arboles > 8.500
|   |   |   |   Numero Caracteristicas > 42.500: G {G=2, P=1}
|   |   |   |   Numero Caracteristicas ≤ 42.500
|   |   |   |   |   Caracteristicas Numericas Continuas > 10.500
|   |   |   |   |   |   Numero Arboles > 16: P {G=0, P=5}
|   |   |   |   |   |   Numero Arboles ≤ 16: G {G=2, P=0}
|   |   |   |   |   Caracteristicas Numericas Continuas ≤ 10.500: P {G=0, P=10}
|   |   |   Numero Arboles ≤ 8.500: G {G=3, P=1}
|   |   Valores Faltantes = SI: G {G=2, P=0}
```

Figura 42. Árbol de decisión para vista minable en fuerza 3

Conforme al árbol de decisión para fuerza 3 se concluye principalmente:

- En conjuntos de datos con dos o más características categóricas nominales es preferible ejecutar la propuesta RFCA, la probabilidad de que RFCA obtenga mejores resultados es 100% (5 de 5 casos donde se presentó la regla).
- Para conjuntos de datos con una o ninguna característica categórica nominal, con un numero de clases menor o igual a 13, sin valores faltantes, fijando un número de árboles mayor o igual a 9, un número total de características menor o igual a 42 y con 10 o menos características numéricas continuas, es preferible la ejecución del RF de Breiman, puesto que la probabilidad de que RF obtenga mejores resultados es 100% (10 de 10 casos donde se presentó la regla).

En la **Figura 43** se muestra el árbol de decisión obtenido en la vista minable de los resultados en exactitud con fuerza 4.

```
Numero de Instancias > 758: P {P=10, G=1}
Numero de Instancias ≤ 758
|   Numero Caracteristicas > 8
|   |   Numero de Instancias > 467.500
|   |   |   Valores Faltantes = NO: P {P=1, G=1}
|   |   |   Valores Faltantes = SI: G {P=0, G=2}
|   |   Numero de Instancias ≤ 467.500: G {P=0, G=11}
|   Numero Caracteristicas ≤ 8
|   |   Caracteristicas Numericas Continuas > 2: P {P=4, G=0}
|   |   Caracteristicas Numericas Continuas ≤ 2: G {P=0, G=3}
```

Figura 43. Árbol de decisión para vista minable en fuerza 4

Con el árbol de decisión para fuerza 4 se deduce principalmente:

- En conjuntos de datos con un número total de instancias menor o igual a 758, con más de 8 características y que el número total de instancias sea menor o igual a 467 es preferible ejecutar la propuesta RFCA, la probabilidad de que RFCA obtenga mejores resultados es 100% (11 de 11 casos donde se presentó la regla).
- En conjuntos de datos con un número total de instancias mayor a 758 es preferible ejecutar RF de Breiman, ya que, la probabilidad de que RF tenga mejores resultados es aproximadamente 91% (10 de 11 casos donde se presentó la regla).

La **Figura 44** muestra el árbol de decisión obtenido en la vista minable de los resultados en exactitud con fuerza 5.

```
Numero de Instancias > 1663.500: P {G=0, P=2}
Numero de Instancias ≤ 1663.500
|   Caracteristicas Numericas Continuas > 1
|   |   Caracteristicas Numericas Continuas > 32: G {G=4, P=0}
|   |   Caracteristicas Numericas Continuas ≤ 32
|   |   |   Numero Arboles > 89: P {G=0, P=4}
|   |   |   Numero Arboles ≤ 89
|   |   |   |   Numero Arboles > 52.500: G {G=7, P=0}
|   |   |   |   Numero Arboles ≤ 52.500
|   |   |   |   |   Numero Arboles > 30.500: P {G=0, P=5}
|   |   |   |   |   Numero Arboles ≤ 30.500: G {G=1, P=1}
|   Caracteristicas Numericas Continuas ≤ 1: G {G=9, P=0}
```

Figura 44. Árbol de decisión para vista minable en fuerza 5

El árbol de decisión para fuerza 5 se deduce principalmente:

- En conjuntos de datos con un número total de instancias menor o igual a 1663, con un numero de características numéricas continuas entre 2 y 32, es mejor no utilizar el RF de Breiman con un número de árboles establecido entre 53 y 89, puesto que, la propuesta RFCA con el número de árboles fijado por el CA de fuerza 5 tiene un 100% de probabilidad de obtener mejores resultados (7 de 7 casos donde se presentaron las condiciones descritas).
- En conjuntos de datos con un número total de instancias menor o igual a 1663 y ninguna característica numérica continua es preferible ejecutar la propuesta RFCA, ya que, la probabilidad de que RFCA obtenga mejores resultados es 100% (9 de 9 casos donde se presentó la regla).
- En conjuntos de datos con un número total de instancias menor o igual a 1663 y un número de características numéricas continuas mayor a 32, es preferible ejecutar la propuesta RFCA, la probabilidad de que RFCA tenga mejores resultados es 100% (4 de 4 casos donde se presentó la regla).
- En conjuntos de datos con un número total de instancias menor o igual a 1663, con un numero de características numéricas continuas entre 2 y 32, fijando un número de árboles mayor a 89 o entre 31 y 52 árboles es mejor utilizar RF de Breiman, puesto que, se tiene un 100% de probabilidad de obtener mejores resultados (4 de 4 y 5 de 5 casos respectivamente).

La **Figura 45** muestra el árbol de decisión obtenido en la vista minable de los resultados en exactitud con fuerza 6.

```
Numero de Instancias > 1478.500: P {P=4, G=0}
Numero de Instancias ≤ 1478.500
|   Caracteristicas Numericas Continuas > 3
|   |   Numero Arboles > 85.500
|   |   |   Numero Arboles > 194: P {P=6, G=2}
|   |   |   Numero Arboles ≤ 194: G {P=0, G=6}
|   |   Numero Arboles ≤ 85.500: P {P=5, G=0}
|   Caracteristicas Numericas Continuas ≤ 3: G {P=0, G=10}
```

Figura 45. Árbol de decisión para vista minable en fuerza 6

En el árbol de decisión para fuerza 6 se concluye principalmente:

- En conjuntos de datos con un número total de instancias menor o igual a 1478 y con un numero de características numéricas continuas menor o igual a 3, es preferible ejecutar la propuesta RFCA, la probabilidad de que RFCA obtenga mejores resultados es 100% (10 de 10 casos donde se presentó la regla).
- En conjuntos de datos con un número total de instancias menor o igual a 1478, con un numero de características numéricas continuas mayor a 3, es mejor no utilizar el RF de Breiman con un número de árboles establecido entre 86 y 194, puesto que, la propuesta RFCA con el número de árboles fijado por el CA de fuerza 6 tiene un 100% de probabilidad de obtener mejores resultados (6 de 6 casos donde se presentaron las condiciones descritas).
- En conjuntos de datos con un número total de instancias menor o igual a 1478, con un numero de características numéricas continuas mayor a 3 y un número de árboles fijado con un valor menor o igual a 85, es preferible emplear el RF de Breiman, ya que, la probabilidad de que RF obtenga mejores resultados es 100% (5 de 5 casos donde se presentó la regla).

La **Figura 46** muestra el árbol de decisión obtenido en la vista minable de los resultados en exactitud con fuerza 7.

```
Numero de Instancias > 1541.500: P {G=0, P=3}
Numero de Instancias ≤ 1541.500
|   Caracteristicas Numericas Continuas > 3
|   |   Numero de Instancias > 1213.500: G {G=2, P=0}
|   |   Numero de Instancias ≤ 1213.500
|   |   |   Numero Arboles > 148
|   |   |   |   Numero de Instancias > 750: P {G=0, P=3}
|   |   |   |   Numero de Instancias ≤ 750: G {G=8, P=3}
|   |   |   Numero Arboles ≤ 148: P {G=0, P=4}
|   Caracteristicas Numericas Continuas ≤ 3: G {G=10, P=0}
```

Figura 46. Árbol de decisión para vista minable en fuerza 7

En el árbol de decisión para fuerza 7 se concluye principalmente:

- En conjuntos de datos con un número total de instancias menor o igual a 1541, con un numero de características numéricas continuas menor o igual a 3, es preferible ejecutar la propuesta RFCA, la probabilidad de que RFCA obtenga mejores resultados es 100% (10 de 10 casos donde se presentó la regla).
- En conjuntos de datos con un número total de instancias menor o igual a 750, con un numero de características numéricas continuas mayor a 3, es mejor no utilizar el RF de Breiman con un número de árboles mayor a 148, puesto que, la propuesta RFCA con el número de árboles fijado por el CA de fuerza 7 tiene una probabilidad aproximada de 73% de obtener mejores resultados (8 de 11 casos donde se presentaron las condiciones descritas).
- En conjuntos de datos con un número total de instancias menor o igual a 1213, con un numero de características numéricas continuas mayor a 3 y un número de árboles fijado con un valor menor o igual a 148, es preferible emplear el RF de Breiman, ya que, la probabilidad de que RF obtenga mejores resultados es 100% (4 de 4 casos donde se presentó la regla).

CAPÍTULO 6

6 CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se propuso y evaluó un nuevo método para la selección de características en el algoritmo Random Forest basada en covering arrays binarios de fuerza 2 hasta 7. En los CA, cada fila representa un subconjunto de características seleccionadas para construir las muestras bootstrap y el número de filas del CA permite definir el número de árboles, suprimiendo de esta forma la selección aleatoria de características del algoritmo Random Forest propuesto por Breiman. Los experimentos se realizaron sobre 33 conjuntos de datos evaluados mediante validación cruzada de 10 folders y los resultados obtenidos son prometedores, en los cuales se logra una mejora promedio en la exactitud entre 0.5% y 2%.

Generalmente el aumento en el número de árboles en Random Forest (RF) produce mejoras en la exactitud del modelo, no obstante, la exactitud puede verse afectada o no variar ante aumentos excesivos o no medidos en el número de árboles [10]. Además, la cantidad optima de árboles en un Random Forest puede variar entre los distintos conjuntos de datos. En este sentido es importante encontrar una manera adecuada de definir el número de árboles; en este contexto, la técnica propuesta suprime la necesidad de definir este hiper parámetro, ya que en la propuesta se define el valor de fuerza para el CA y con éste se define automáticamente el número de árboles. Es preciso destacar además que el rango de valores del parámetro fuerza esta entre 2 y 7, siendo este un rango bien limitado y mucho menor al actual rango no limitado (únicamente debe cumplir con ser un valor entero positivo) del número de árboles.

El nuevo clasificador RFCA mejora la exactitud en la mitad de los escenarios de evaluación con respecto a los algoritmos del estado del arte empleados, en promedio la mayor exactitud en el algoritmo RFCA se obtiene en fuerza 7, no obstante, considerando que una mayor fuerza (t) representa un mayor número de árboles, se considera a la fuerza 5 como el valor adecuado para el parámetro t del RFCA. Por otra parte, el uso de fuerza 2 presenta los resultados más bajos en términos de exactitud. En términos de la medida F se obtiene un mayor valor en el algoritmo RFCA con los valores de fuerza 4,5 y 7 y los menores valores en medida F se obtuvieron al usar fuerza 3 y 6, lo que motiva un estudio teórico posterior, ya que la fuerza 6 no se comporta apropiadamente.

La diferencia en términos de tiempos de ejecución de la propuesta RFCA con respecto al algoritmo referencia de Breiman (RF) y RF_SQRT no supero el 3.7%,

con lo cual se deduce que la inclusión de los covering arrays en el algoritmo propuesto no genera un aumento significativo en el tiempo de ejecución, y que incluso en algunos conjuntos de datos como Car, Chart, Contraceptive y Soy Bean se obtuvieron menores tiempos de ejecución. Con lo cual se considera que el aumento o disminución de tiempo en los resultados depende más de los aspectos propios de los conjuntos de datos, como los son: número de clases, número de instancias, numero de características y tipos de características (discretas, continuas, nominales, ordinales).

Mediante la comparación de los resultados de exactitud del algoritmo RFCA con respecto al algoritmo de referencia de Breiman [9], se obtuvieron seis modelos de árboles de decisión (uno para cada fuerza). Con estos seis modelos se identificaron los tipos de conjuntos de datos donde la propuesta RFCA tiene una mayor probabilidad de obtener mejores resultados.

Se estudió la arquitectura e implementación de RF en Weka, descargando el código fuente de weka y revisando la arquitectura general del framework. Se analizó la manera en que interactúan las clases y la forma como se usan entre ellas, lo que permitió identificar la implementación de RF y las clases asociadas con dicho clasificador.

Con el fin mantener y seguir la arquitectura del software weka (versión 3.8), se implementó e integro en un paquete weka, la propuesta Random Forest basada en CA y Random Forest basado en TCA. El paquete desarrollado funciona de manera independiente, no genera alteraciones sobre el comportamiento de las demás funcionalidades del software e incluye clases para el manejo de CAs y TCAs. Además, es de fácil distribución, puesto que la instalación se realiza a través del sistema de gestión de paquetes del software, y luego de instalarse es posible utilizar ambas propuestas en Conjunto de Datos cargados en el software weka.

Como trabajo futuro se espera estudiar el desempeño de RFCA sobre variaciones de Random Forest en los que la muestra Bootstrap sea de tamaño distinto a las 2/3 partes del conjunto de datos de entrenamiento [9], o en el que el muestreo de las instancias se realice sin remplazo [45]. Por otra parte, la ejecución de pruebas en otros conjuntos de datos de prueba del repositorio de la UCI o similares. Asimismo, la realización de pruebas sobre el parámetro k de grupos de características que sean diferentes a los empleados en el presente trabajo.

CAPÍTULO 7

7 BIBLIOGRAFÍA

[1] C. C. Aggarwal, *Data Mining: The Textbook*. New York: Springer Publishing Company, Incorporated, 2015.

[2] I. H. (Ian H. . Witten, E. Frank, and M. A. (Mark A. Hall, *Data mining : practical machine learning tools and techniques*. Morgan Kaufmann, 2011.

[3] A. Ahlemeyer-Stubbe and S. Coleman, *A practical guide to data mining for business and industry*. 2014.

[4] M. Mazid, S. Ali, and K. Tickle, “Improved C4. 5 algorithm for rule based classification,” *Proc. 9th WSEAS Int. Conf. Artif. Intell. Knowl. Eng. data bases*, pp. 296–301, 2010.

[5] J. Quinlan, *C4. 5: programs for machine learning*, vol. 240. Morgan Kaufmann Publishers, 1993.

[6] A. Bar-Hen, S. Gey, and J. M. Poggi, “Influence Measures for CART Classification Trees,” *J. Classif.*, vol. 32, no. 1, pp. 21–45, Apr. 2015.

[7] A. Ziegler and I. R. König, “Mining data with random forests: Current options for real-world applications,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 4, no. 1, pp. 55–63, Jan. 2014.

[8] G. a Marcoulides, *Discovering Knowledge in Data: an Introduction to Data Mining:Discovering Knowledge in Data: An Introduction to Data Mining*, vol. 100, no. 472. Wiley-Interscience, 2005.

[9] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[10] H. Parmar, S. Bhanderi, and G. Shah, “Sentiment Mining of Movie Reviews using Random Forest with Tuned Hyperparameters,” 2014.

[11] S. Bernard, L. Heutte, and S. Adam, “Influence of hyperparameters on random forest accuracy,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5519 LNCS, 2009, pp. 171–180.

[12] C. A. Cobos-lozada and J. Torres-jimenez, “Metaheuristic algorithms for building Covering Arrays : A review Algoritmos metaheurísticos para construir Covering Arrays : Revisión Algoritmos metaheurísticos para construir Covering Arrays :,” *Rev. Fac. Ing.*, vol. 25, no. 43, pp. 31–45, 2016.

[13] H. Ordoñez, J. Torres-Jimenez, A. Ordoñez, and C. Cobos, “Clustering business process models based on multimodal search and covering arrays,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10062 LNAI, Springer, Cham, 2017, pp. 317–328.

[14] H. Dorado, C. Cobos, and J. Torres-Jimenez, “Wrapper para la construcción de modelos de aprendizaje supervisado basado en arreglos de cobertura que permite la estimación de la importancia de las variables de entrada y la

selección de atributos (Propuesta de Tesis)," Universidad del Cauca, 2017.
[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software," *ACM SIGKDD Explor. Newsl.*, vol. 11, no. 1, p. 10, Nov. 2009.
[16] P. Cichosz, *Data Mining Algorithms*. 2015.
[17] M. J. Zaki and W. Meira, "Data Mining and Analysis: fundamental concepts and algorithms.," *Data Min. Anal.*, p. 604, 2014.
[18] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[19] A. Verikas, A. Gelzinis, and M. Bacauskiene, "Mining data with random forests: A survey and results of new tests," *Pattern Recognit.*, vol. 44, no. 2, pp. 330–349, Feb. 2011.
[20] K. Bache and M. Lichman, "UCI Machine Learning Repository," *University of California Irvine School of Information*, 2013. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html. [Accessed: 20-Jul-2017].
[21] J. Torres-Jimenez, I. Izquierdo-Marquez, R. N. Kacker, and D. Richard Kuhn, "Tower of covering arrays," *Discret. Appl. Math.*, vol. 190–191, pp. 141–146, 2015.
[22] I. I. Márquez, "Construcción de Torres de Covering Arrays," Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2013.
[23] L. Ma, S. Fan, A. Haywood, Z. Ming-tian, and J. Rigol-Sanchez, "CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests," *BMC Bioinformatics*, vol. 18, no. 1, p. 169, Dec. 2017.
[24] E. Elyan and M. M. Gaber, "A genetic algorithm approach to optimising random forests applied to class engineered data," *Inf. Sci. (Ny).*, vol. 384, pp. 220–234, Apr. 2017.
[25] Q. Zhou, H. Zhou, and T. Li, "Cost-sensitive feature selection using random forest: Selecting low-cost subsets of informative features," *Knowledge-Based Syst.*, vol. 95, pp. 1–11, 2016.
[26] K. Dheenadayalan, G. Srinivasaraghavan, and V. N. Muralidhara, "Pruning a Random Forest by Learning a Learning Algorithm," Springer, Cham, 2016, pp. 516–529.
[27] C. K. Aridas, S. B. Kotsiantis, and M. N. Vrahatis, "Increasing Diversity in Random Forests Using Naive Bayes," Springer, Cham, 2016, pp. 75–86.
[28] N. Adnan, "On Dynamic Selection of Subspace for Random Forest," Springer, Cham, 2014, pp. 370–379.
[29] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, vol. p. 1984.
[30] S. Bernard, S. Adam, and L. Heutte, "Dynamic Random Forests," *Pattern Recognit. Lett.*, vol. 33, no. 12, pp. 1580–1586, Sep. 2012.
[31] H. Deng and G. Runger, "Feature selection via regularized trees," in *Proceedings of the International Joint Conference on Neural Networks*, 2012, pp. 1–8.
[32] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht,

"On oblique random forests," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6912 LNAI, no. PART 2, Springer, Berlin, Heidelberg, 2011, pp. 453–469.

[33] S. Bernard, L. Heutte, and S. Adam, "Forest-RK: A new random forest induction method," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5227 LNAI, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 430–437.

[34] J. Alcalá-Fdez *et al.*, "KEEL: a software tool to assess evolutionary algorithms for data mining problems," *Soft Comput.*, vol. 13, no. 3, pp. 307–318, Feb. 2009.

[35] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006.

[36] I. H. Witten, E. Frank, and M. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*. 2016.

[37] F. Gorunescu, *Data mining: Concepts, models and techniques*, vol. 12. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[38] M. Bramer, *Principles of Data Mining*. 2016.

[39] F. Guillet and H. J. Hamilton, *Quality measures in data mining*. Springer, 2007.

[40] P. C. Goiser, "Quality and Complexity Measures for Data Linkage and Deduplication," *Qual. Meas. Data Min.*, vol. NA, 2007.

[41] M. N. Adnan and M. Z. Islam, "Optimizing the number of trees in a decision forest to discover a subforest with high ensemble accuracy using a genetic algorithm," *Knowledge-Based Syst.*, vol. 110, pp. 86–97, Oct. 2016.

[42] R. Houari, A. Bounceur, T. Kechadi, T. Abdelkamel, and R. Euler, "A New Method for Estimation of Missing Data Based on Sampling Methods for Data Mining," Springer, Heidelberg, 2013, pp. 89–100.

[43] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. Appl.*, vol. 73, pp. 220–239, May 2017.

[44] M. (Computer scientist) Hofmann and R. Klinkenberg, *RapidMiner : data mining use cases and business analytics applications*. .

[45] E. Scornet, G. Biau, and J.-P. Vert, "Consistency of random forests," May 2014.