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

Juan Sebastian Vivas Mendez

Director: PhD. Carlos Alberto Cobos Lozada
Co-Director: 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

1 PROPUESTA

En esta sección del documento se describe la propuesta de Random Forest basada en Torres de Covering Arrays(RFTCA), a partir de RFTCA y su mecanismo de integración de Torres de Covering Arrays (TCA) se generaron variaciones al algoritmo con el fin de establecer la manera que mejor integra las Torres Covering Arrays como mecanismo de selección de características.

1.1 RANDOM FOREST BASADO EN TORRES DE COVERING ARRAYS(RFTCA)

En el Random Forest de Breiman [1] 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. 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 nivel en el que aumentar el número de árboles no mejora e incluso disminuye la exactitud del modelo [2]. En este contexto RFTCA integra las Torres de Covering Arrays (Torres de 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).

RFTCA emplea TCAs de alfabeto binario (v = 2). Esto significa que cada componente del TCA tiene únicamente valores {0, 1}, los subconjuntos de características candidatas empleadas en la construcción de cada uno de los árboles del RF se determinan con las filas del TCA. El parámetro P del TCA se establece a través del número total de características del conjunto de datos a ser clasificado. A continuación, se presenta un ejemplo de ejecución de RFTCA y luego se describe el algoritmo.

1.1.1 Ejemplo de RFTCA

Con el fin de ilustrar el comportamiento de RFTCA se propone 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) que contiene 3 clases (setosa, versicolor, virginica) de 50 instancias cada una, 4 características (longitud, ancho del sépalo y longitud, ancho del pétalo), cada clase se refiere a una especie de la planta iris [3]. En esta versión adaptada se hará uso de 20 instancias. En la **Tabla 1**, se observa la descripción de las características del conjunto de Datos. Para la construcción de los árboles se emplea el TCA de la **Figura 1** con 8 filas (N), 4 columnas (P), alfabeto binario (v=2) y fuerza 2 (t).

Longitud Sépalo	Ancho Sépalo	Longitud Pétalo	Ancho Pétalo	Clase
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 adaptado

$$\mathrm{TCA}(N=8,P=4,v=2,t=2)=\begin{bmatrix}0&0&1&0\\0&1&0&1\\1&0&0&1\\1&0&1&0\\1&0&1&1\\1&1&0&0\\1&1&0&0\\1&1&1&1\end{bmatrix}$$

Figura 1. TCA (8;4,2,2)

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.

El número de árboles a construir en RFTCA está determinado por el número de filas del TCA binario elegido de acuerdo con el número de características del conjunto de datos de entrenamiento original. Para el ejemplo y basado en la **Figura 1** el número M de árboles a generar es igual a cinco (8) que corresponde a las 8 filas del TCA.

Cada fila del TCA binario define para su árbol correspondiente cuales características (atributos del conjunto de datos de entrenamiento original) serán usadas en su construcción. Donde 0 indica la usencia y 1 la presencia de una variable en el subconjunto de características candidatas. Teniendo en cuenta por ejemplo la fila cinco $[1|0|1|1]$ del TCA de la **Figura 1** se puede afirmar que el árbol se construye teniendo en cuenta las características uno, tres y cuatro (Longitud Sépalo, Longitud Pétalo y Ancho Pétalo).

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 [1], 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 $K = \lfloor \log_2(4) + 1 \rfloor = 3$.

Como se puede observar para el ejemplo de la fila dos del TCA, solamente selecciona dos características, por esta razón el algoritmo RFTCA añade una en forma aleatoria para completar los tres requeridos por el parámetro K. También puede ocurrir que la fila del TCA tenga más características seleccionadas que las definidas por el parámetro K, en este caso se seleccionan aleatoriamente K características del subconjunto definido por la línea del TCA.

Conforme al conjunto de datos del ejemplo se requiere de una Torre de Covering Array Binario que permita cubrir las 4 características de entrada, en el presente ejemplo se emplea la Torre de Covering Array Binario de la **Figura 1**. Conforme a este TCA, el número de árboles es ocho (número filas TCA).

Construcción de $Arbol_1$ basado en la fila 1 $[0|0|1|0]$ del TCA. En la selección de características, se escoge la característica de la posición 3 (Longitud Pétalo). En este caso, el número de características del subconjunto seleccionado por esta fila del TCA 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 2 características adicionales). Para el ejemplo se selecciona la característica 2 (Ancho Sépalo) y 4 (Ancho Pétalo). La **Figura 2** presenta la muestra de datos y el árbol base obtenido.

Datos para el ARBOL 1			
Ancho Sépalo	**Longitud Pétalo**	**Ancho Pétalo**	**Clase**
3.5	1.4	0.2	setosa
3	1.4	0.2	setosa
3.1	1.5	0.2	setosa
3.6	1.4	0.2	setosa
3.2	4.7	1.4	versicolor
3.2	4.5	1.5	versicolor
2.3	4	1.3	versicolor

Longitud Pétalo < 4.9
    Longitud Pétalo < 2.75 : setosa (4/0)
    Longitud Pétalo >= 2.75 : versicolor (5/0)
Longitud Pétalo >= 4.9 : virginica (7/0)

2.7	5.1	1.9	virginica
3	5.9	2.1	virginica
3	5.8	2.2	virginica
3	6.6	2.1	virginica

Figura 2. Árbol generado para el conjunto de entrenamiento 1

Construcción de $Arbol_2$ basado en la fila 2 $[0|1|0|1]$ del TCA. En la selección de características, se escogen las características de la posición 2 (Ancho Sépalo) y 4 (Ancho Pétalo). En este caso, el número de características del subconjunto seleccionado por esta fila del TCA es menor al número k de características a seleccionar (k = 3), en consecuencia, para esos casos el algoritmo selecciona la característica faltante de manera aleatoria y sin repetición (en este caso 1 característica adicional). Para el ejemplo se selecciona la característica 1 (Longitud Sépalo). La **Figura 3** presenta la muestra de datos y el árbol base obtenido.

Datos para el ARBOL 2			
Longitud Sépalo	**Ancho Sépalo**	**Ancho Pétalo**	**Clase**
5.1	3.5	0.2	setosa
4.9	3	0.2	setosa
4.6	3.1	0.2	setosa
7	3.2	1.4	versicolor
6.4	3.2	1.5	versicolor
6.9	3.1	1.5	versicolor
5.8	2.7	1.9	virginica
6.3	2.9	1.8	virginica
6.5	3	2.2	virginica
7.6	3	2.1	virginica

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

Figura 3. Árbol generado para el conjunto de entrenamiento 2

Construcción de $Arbol_3$ basado en la fila 3 $[1|0|0|1]$ del TCA. En la selección de características, se escogen las características de la posición 1(Longitud Sépalo) y 4 (Ancho Pétalo). En este caso, el número de características del subconjunto seleccionado por esta fila del TCA es menor al número k de características a seleccionar (k = 3), en consecuencia, para esos casos el algoritmo selecciona la característica faltante de manera aleatoria y sin repetición (en este caso 1 característica adicional). Para el ejemplo se selecciona la característica 3 (Longitud Pétalo). La **Figura 4** presenta la muestra de datos y el árbol base obtenido.

Datos para el ARBOL 3			
Longitud Sépalo	**Longitud Pétalo**	**Ancho Pétalo**	**Clase**
5.1	1.4	0.2	setosa
4.9	1.4	0.2	setosa
4.7	1.3	0.2	setosa
4.6	1.5	0.2	setosa
5.5	4	1.3	versicolor
6.3	6	2.5	virginica
5.8	5.1	1.9	virginica
7.1	5.9	2.1	virginica
6.5	5.8	2.2	virginica
7.6	6.6	2.1	virginica

Longitud Sépalo < 5.65
    Longitud Sépalo < 5.3 : setosa (6/0)
    Longitud Sépalo >= 5.3 : versicolor (1/0)
Longitud Sépalo >= 5.65 : virginica (9/0)

Figura 4. Árbol generado para el conjunto de entrenamiento 3.

Construcción de $Arbol_4$ con base en la fila 4 $[1|0|1|0]$ del TCA. En la selección de características, se escogen las características de la posición 1 (Longitud Sépalo) y 3 (Longitud Pétalo). En este caso, el número de características del subconjunto seleccionado por esta fila del TCA es menor al número k de características a seleccionar (k = 3), en consecuencia, para esos casos el algoritmo selecciona la característica faltante de manera aleatoria y sin repetición (en este caso 1 característica adicional). Para el ejemplo se selecciona la característica 2 (Ancho Sépalo). La **Figura 5** presenta la muestra de datos y el árbol base obtenido.

Datos para el ARBOL 4			
Longitud Sépalo	**Ancho Sépalo**	**Longitud Pétalo**	**Clase**
5.1	3.5	1.4	setosa
4.9	3	1.4	setosa
4.7	3.2	1.3	setosa
4.6	3.1	1.5	setosa
6.3	3.3	6	virginica
5.8	2.7	5.1	virginica
7.1	3	5.9	virginica
6.3	2.9	5.6	virginica
6.5	3	5.8	virginica
7.6	3	6.6	virginica

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

Figura 5. Árbol generado para el conjunto de entrenamiento 4

Finalmente, para la construcción de $Arbol_5$, se toma como base la fila 5 $[1|0|1|1]$ del TCA. Las características definidas por esta fila son la 1 (Longitud Sépalo), 3 (Longitud Pétalo) y 4 (Ancho Pétalo). La **Figura 6** presenta la muestra de datos y el árbol base obtenido.

Datos para el ARBOL 5			
Longitud Sépalo	**Longitud Pétalo**	**Ancho Pétalo**	**Clase**
4.9	1.4	0.2	setosa
4.6	1.5	0.2	setosa
5	1.4	0.2	setosa
6.4	4.5	1.5	versicolor
6.9	4.9	1.5	versicolor
5.8	5.1	1.9	virginica
7.1	5.9	2.1	virginica
6.5	5.8	2.2	virginica
7.6	6.6	2.1	virginica

Longitud Pétalo < 5
    Longitud Sépalo < 5.7 : setosa (4/0)
    Longitud Sépalo >= 5.7 : versicolor (5/0)
Longitud Pétalo >= 5 : virginica (7/0)

Figura 6. Árbol generado para el conjunto de entrenamiento 5

Construcción de $Arbol_6$ basado en la fila 6 $[1|1|0|0]$ del TCA. En la selección de características, se escogen las características de la posición 1 (Longitud Sépalo) y 2 (Ancho Sépalo). En este caso, el número de características del subconjunto seleccionado por esta fila del TCA es menor al número k de características a seleccionar (k = 3), en consecuencia, para esos casos el algoritmo selecciona la característica faltante de manera aleatoria y sin repetición (en este caso 1 característica adicional). Para el ejemplo se selecciona la característica 3 (Longitud Pétalo). La **Figura 7** presenta la muestra de datos y el árbol base obtenido.

Datos para el ARBOL 6			
Longitud Sépalo	**Ancho Sépalo**	**Longitud Pétalo**	**Clase**
5.1	3.5	1.4	setosa
4.9	3	1.4	setosa
7	3.2	4.7	versicolor
6.9	3.1	4.9	versicolor
5.5	2.3	4	versicolor
6.5	2.8	4.6	versicolor
6.3	3.3	6	virginica
5.8	2.7	5.1	virginica
7.1	3	5.9	virginica
6.5	3	5.8	virginica

Longitud Pétalo < 5
    Longitud Sépalo < 5.3 : setosa (2/0)
    Longitud Sépalo >= 5.3 : versicolor (5/0)
Longitud Pétalo >= 5 : virginica (7/0)

Figura 7. Árbol generado para el conjunto de entrenamiento 6.

Construcción de $Arbol_7$ con base en la fila 7 $[1|1|0|0]$ del TCA. En la selección de características, se escogen las características de la posición 1(Longitud Sépalo) y 2(Ancho Sépalo). En este caso, el número de características del subconjunto seleccionado por esta fila del TCA es menor al número k de características a seleccionar (k = 3), en consecuencia, para esos casos el algoritmo selecciona la característica faltante de manera aleatoria y sin repetición (en este caso 1 característica adicional). Para el ejemplo se selecciona la característica 4 (Ancho Pétalo). La **Figura 8** presenta la muestra de datos y el árbol base obtenido.

Datos para el ARBOL 7			
Longitud Sépalo	**Ancho Sépalo**	**Ancho Pétalo**	**Clase**
4.9	3	0.2	setosa
5	3.6	0.2	setosa
7	3.2	1.4	versicolor
6.4	3.2	1.5	versicolor
6.9	3.1	1.5	versicolor
6.5	2.8	1.5	versicolor
6.3	3.3	2.5	virginica
5.8	2.7	1.9	virginica
7.1	3	2.1	virginica
6.3	2.9	1.8	virginica
6.5	3	2.2	virginica

Longitud Pétalo < 5
    Longitud Sépalo < 5.7 : setosa (3/0)
    Longitud Sépalo >= 5.7 : versicolor (5/0)
Longitud Pétalo >= 5 : virginica (8/0)

Figura 8. Árbol generado para el conjunto de entrenamiento 7

Finalmente, para la construcción de $Arbol_8$, se toma como base la fila 5 $[1|1|1|1]$ del TCA. En este caso las 4 características son candidatas a ser seleccionadas, a saber: 1 (Longitud Sépalo ,2 (Ancho Sépalo), 3 (Longitud Pétalo) y 4 (Ancho Pétalo), de los cuales se seleccionan aleatoriamente 3 y sin repetir, a saber: el 2, 3 y 4. La **Figura 9** presenta la muestra de datos y el árbol base obtenido.

Datos para el ARBOL 8			
Ancho Sépalo	**Longitud Pétalo**	**Ancho Pétalo**	**Clase**
3.5	1.4	0.2	setosa
3	1.4	0.2	setosa
3.1	1.5	0.2	setosa
3.6	1.4	0.2	setosa
3.2	4.7	1.4	versicolor
3.2	4.5	1.5	versicolor
2.3	4	1.3	versicolor
2.8	4.6	1.5	versicolor
3.3	6	2.5	virginica
2.7	5.1	1.9	virginica
3	5.9	2.1	virginica
3	5.8	2.2	virginica

Ancho Pétalo < 1.7
    Longitud Sépalo < 5.3 : setosa (4/0)
    Longitud Sépalo >= 5.3 : versicolor (5/0)
Ancho Pétalo >= 1.7 : virginica (7/0)

Figura 9. Árbol generado para el conjunto de entrenamiento 8

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 100% de instancias correctamente clasificadas ya que en todos los casos el valor real y el valor predicho coinciden.

	Instancia de Prueba 1	Instancia de Prueba 2	Instancia de Prueba 3	Instancia de Prueba 4
Árbol 1	setosa	versicolor	virginica	virginica
Árbol 2	setosa	versicolor	virginica	virginica
Árbol 3	setosa	virginica	virginica	virginica
Árbol 4	setosa	virginica	virginica	virginica
Árbol 5	setosa	versicolor	virginica	virginica
Árbol 6	setosa	versicolor	virginica	virginica
Árbol 7	setosa	versicolor	virginica	virginica
Árbol 8	setosa	versicolor	virginica	virginica
Clase Real	setosa	versicolor	virginica	virginica
Resultado (Voto Mayoritario)	**setosa**	**versicolor**	**virginica**	**virginica**

Tabla 2. Resultados Conjunto de Datos de prueba

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

Algoritmo RFTCA (Random Forest basado en Torres de Covering Arrays)

Entradas: T **/*** Conjunto de datos de entrenamiento*/
f /* Fuerza de la Torre de Covering array */
lsize /* Tamaño límite de la hoja */
Test **/*** Conjunto de datos de prueba */

Salida: Etiqueta de clase para los datos de entrada.

Inicio

1: r = numRows (T) /* Numero de instancias del Conjunto de Datos ***/**
2: P = numAttributes (T) /* Numero de características del Conjunto de Datos ***/**
3: **t**ca = loadTCA (f, P) /* Carga la Torre de Covering Array (TCA) acorde a f y P * / */
4: M = numRowsTCA (tca) **/*** Numero de árboles definido como número de filas del TCA */
5: K = $\lfloor \mathbf{log_2}(\boldsymbol{P}) + \mathbf{1} \rfloor$ /* Numero de características a usar en la construcción del árbol ***/**
6: att = listAttributes (T) /* Índices de las características del conjunto de datos */
7: **Desde i = 1 hasta M hacer**
8: T' = bootstrap (T) **/*** Selecciona $\frac{2*r}{3}$ puntos (instancias del conjunto de datos), con reemplazo, uniformemente en T */
9: 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. RFTCA (Random Forest basado en Torres de Covering Arrays)

En la línea 1 se define la variable r como el número de filas(instancias) del conjunto de datos de entrenamiento.

En la línea 2 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 TCA son archivos en formato .caGold, cada archivo esta nombrado de la siguiente manera: N**a**k**b**v2^**b**t6, donde **a** es el número de filas del TCA, **b** es el número de columnas del CA, v2 indica que el archivo corresponde a un Covering array binario y t6 es la fuerza del TCA, la cual indica que estos TCAs tienen contenidos los CAs de fuerzas y factores inferiores (fuerza 2,3,4,5). por ejemplo, el archivo N**128**k**11**v2^**11**t6, corresponde a un TCA de 128 filas, 11 columnas, binario, y de fuerza 6 , por lo cual este TCA contiene los CAs de fuerza 2,3,4,5. La **Figura 10** muestra parte del contenido del archivo N128K11v2^11t6.caGold, como se puede observar la identificación de las distintas fuerzas presentes en el TCA se encuentra al interior del archivo .caGold.

```
0 0 1 0 1 0 1 1 1 0 0
0 1 0 1 0 1 0 0 0 1 1
1 0 0 1 0 0 1 0 1 1 0
1 0 1 0 0 1 0 0 1 0 1
1 0 1 1 1 1 1 1 0 1 1
1 1 0 0 1 0 0 1 1 1 1
1 1 0 0 1 1 1 0 0 0 0
1 1 1 1 0 0 0 1 0 0 0
--t=2
0 0 0 0 1 1 0 1 0 0 1
0 0 1 1 0 0 1 0 0 0 1
0 0 1 1 0 1 0 1 1 1 0
0 1 0 0 0 0 1 1 0 1 0
0 1 0 1 1 0 0 0 1 0 0
0 1 1 0 1 1 1 0 1 1 1
1 0 1 0 1 0 0 0 0 1 0
1 1 0 1 0 1 1 1 1 0 1
--t=3
0 0 0 1 1 0 0 1 0 1 0
0 0 0 1 1 1 1 0 1 0 1
0 0 1 0 0 1 1 0 0 1 0
.   .   .   .   .   .
.   .   .   .   .   .
.   .   .   .   .   .
--t=6
```

Figura 10. Parte del contenido del archivo N128K11v2^11t6.caGold

Bajo este contexto, inicialmente, se carga la información de los archivos .caGold en una lista de objetos, donde cada objeto contiene los siguientes atributos: nombre completo del archivo .caGold(nombreTCA) y número de características del

TCA(numColumnas). Luego, la lista de objetos se organiza de manera ascendente con respecto al número de atributos, se recorre la lista y se retorna el objeto que tenga en su atributo numColumnas un valor mayor o igual a el indicado en el parámetro P (número de características del conjunto de datos).

A partir del atributo nombreTCA del objeto retornado se carga el Torre de Covering Array correspondiente. Luego, el parámetro f se emplea para identificar hasta que fila debe cargarse de la Torre de Covering Array (comparando f con la línea de texto --t = valor Fuerza). Si la Torre de 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 dicha Torre de Covering Array. Este proceso de acortar columnas es válido, ya que la Torre de Covering Array resultante también contiene todas las combinaciones de los v^t símbolos al menos una vez.

En la línea 04 se establece el número de árboles(M) como el número N de filas de la Torre de 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 $\mathbf{K} = [\mathbf{log_2}(\boldsymbol{P}) + \mathbf{1}]$ corresponde al valor indicado por el algoritmo referencia de Breiman [1].

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 8 se crea una muestra bootstrap, especificada en la **ecuación (1)**. Como es una muestra boostrap, una instancia puede ser seleccionada más de una vez y también puede que no sea seleccionada para la construcción de la muestra [1], [4].

$$Muestra\ Bootstrap = \frac{2 * r}{3} \qquad (1)$$

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

En la línea 9 se definen los índices de las características con las que se crea un árbol de decisión. Representando una fila TCA[i] 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 TCA (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** se 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 por la rama de datos específica y se retorna el subárbol vacío (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

1: Si numInstances (T') > lsize entonces
2:     subK = Ø
3:     Si size (subAS) >= K entonces
4:         subK = RandomSelect (subAS, K) /*Selecciona uniformemente y sin remplazo un
           subconjunto subK de K características en subAS, subK ⊂ subAS */
5:     Sino
6:         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 */
       Fin si
       /* Selecciona la mejor división en T' optimizando el criterio de partición CART */
7:     (leftT, rightT) = Split (T', subK)
8:     Tree.add (trainDT(leftT, K, subAS, att, lsize))      /* Hijo izquierdo */
9:     Tree.add (trainDT(rightT, K, subAS, att, lsize))     /* Hijo derecho */
   Sino
       retorna Árbol
   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 TCA) 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 variable que se desea discriminar y a su vez mantener el árbol razonablemente pequeño [5].

2 VARIACIONES RANDOM FOREST BASADO EN TORRES DE COVERING ARRAYS(RFTCA)

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

2.1.1 RFTCA sin parámetro K (RFTCA SINK)

En esta variación del algoritmo RFTCA, 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 TCA, 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.

2.1.2 RFTCA variación 1 y RFTCA variación 2

En las variaciones 1 y 2 de RFTCA 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 TCA es menor al valor de K.

En la variación 1 (**RFTCA VAR1**) el conjunto de características está compuesto únicamente por las características indicadas en la fila del TCA, 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 (**RFTCA 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 TCA.

2.1.3 RFTCA y variaciones raíz cuadrada

En [6][7] se propone la **ecuación (2)** para definir el número K de características seleccionadas. Este valor propuesto para el hiper parámetro K se experimentó en el algoritmo RFTCA 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: RFTCA_SQRT, RFTCA_VAR1_SQRT y RFTCA_VAR2_SQRT.

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

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

3 EXPERIMENTACION

3.1 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 TCA se evaluaron usando fuerza 2 hasta 6, todos con el 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 TCA definido en función del número de características en el conjunto (parámetro P en el 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 [8] , con el fin de obtener las pruebas estadísticas no paramétricas de Wilcoxon y Friedman.

3.1.1 Resultados RFTCA y variaciones

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

Tras finalizar las 30 ejecuciones de cada algoritmo Random Forest basado en Torres de Covering Arrays (RFTCA 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 Torres de 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 3** muestra el ranking del test de Friedman de los algoritmos basados en Torres de Covering Arrays y los algoritmos del estado del arte. En esta tabla se evidencia claramente que el mejor algoritmo basado en Torres

de Covering Arrays es RFTCA. El valor p obtenido para la prueba fue de 0.00123. Luego, al ser este valor menor a 0.05, se considera válido estadísticamente.

Puesto	Algoritmo	Ranking
1	**RFTCA**	**4.4242**
2	RFTCA-VAR1	4.9606
3	RFTCA-VAR2	4.497
4	RF-SQRT	4.9697
5	RF	4.9848
6	RFTCA-SQRT	5.103
7	RFTCA-SQRT-VAR2	5.133
8	RFTCA-SQRT-VAR1	5.2212
9	RFTCA-SINK	5.7061
Valor de P	0. 00123 (chi cuadrado con 8 grados de libertad: 25.587879)	

Tabla 3. Ranking con el resultado de exactitud en los algoritmos según Test de Friedman

La **Figura 11** 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 un algoritmo sobre los demás. El mejor algoritmo determinado en el ranking del test Friedman (RFTCA) numerado en la **Figura 11** como (2) domina a cinco de las variaciones DE RFTCA (RFTCA_SINK, RFTCA_VAR1, RFTCA_SQRT, RFTCA_SQRT_VAR1 Y RFTCA_SQRT_VAR2), no obstante no se establece dominancia en los algoritmos del estado del arte(RF y RF_SQRT), Asimismo, el algoritmo que no emplea el parámetro K (RFTCA_SINK) denotado como (1) en la **Figura 11** es dominado por RFTCA, las variaciones propuestas y uno de los algoritmos del estado del arte(RF-SQRT).

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

Figura 11. Test Wilcoxon de los resultados de exactitud en los algoritmos

La **Tabla 4** muestra el ranking del test de Friedman para los resultados obtenidos en la medida F de los algoritmos basados en Torres de Covering Arrays y los algoritmos del estado del arte. En la tabla se evidencia que el mejor algoritmo basado en Torres Covering Arrays es RFTCA. 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	**RFTCA**	**4.2091**
2	RFTCA-VAR2	4.2273
3	RF-SQRT	4.3515
4	RF	4.5636
5	RFTCA-VAR1	5.2576
6	RFTCA-SQRT	5.3364
7	RFTCA-SQRT-VAR2	5.3606
8	RFTCA-SQRT-VAR1	5.7727
9	RFTCA-SINK	5.9212
p-value	4.7434E-9 (chi cuadrado con 8 grados de libertad: 78.95474)	

Tabla 4. 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 12,** en los resultados la dominancia del mejor algoritmo para la medida F determinado en el ranking del test Friedman (RFTCA) numerado en la **Figura 12** como (2), domina a las variaciones de RFTCA con excepción de la variación RFTCA-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)
RFTCA-SINK (1)	-	○	○	○	○		○	○	○
RFTCA (2)	●	-	●		●	●	●		
RFTCA-VAR1 (3)		○	-	○		●		○	○
RFTCA-VAR2 (4)	●		●	-	●	●	●		
RFTCA-SQRT (5)	●	○		○	-	●		○	○
RFTCA-SQRT-VAR1 (6)		○	○	○	○	-	○	○	○
RFTCA-SQRT-VAR2 (7)	●	○		○		●	-	○	○
RF (8)	●		●		●	●	●	-	
RF-SQRT (9)	●		●		●	●	●		-

Figura 12. 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 RFTCA es el que mejor integra el mecanismo de selección de características basado en Torres de Covering Array.

3.1.2 Resultados RFTCA y Estado del Arte

En la siguiente sección se exponen y analizan los resultados del algoritmo RFTCA, los resultados para la exactitud, medida F y tiempo de ejecución 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, evaluando Torres de Coveirng Arrays con alfabeto binario de fuerza 2 hasta 6.

3.1.2.1 Resultados y análisis para la exactitud en RFTCA

Los resultados en términos de exactitud presentados en las figuras 13 y 14 son los conjuntos de datos donde la propuesta RFTCA obtiene mejores resultados con respecto a los algoritmos del estado del arte. La figura 15 corresponde con los a los conjuntos de datos donde la propuesta RFTCA obtiene resultados similares a los del Estado de Arte. Las figuras 16 y 17 son los conjuntos de datos donde la propuesta RFTCA obtiene resultados más bajos.

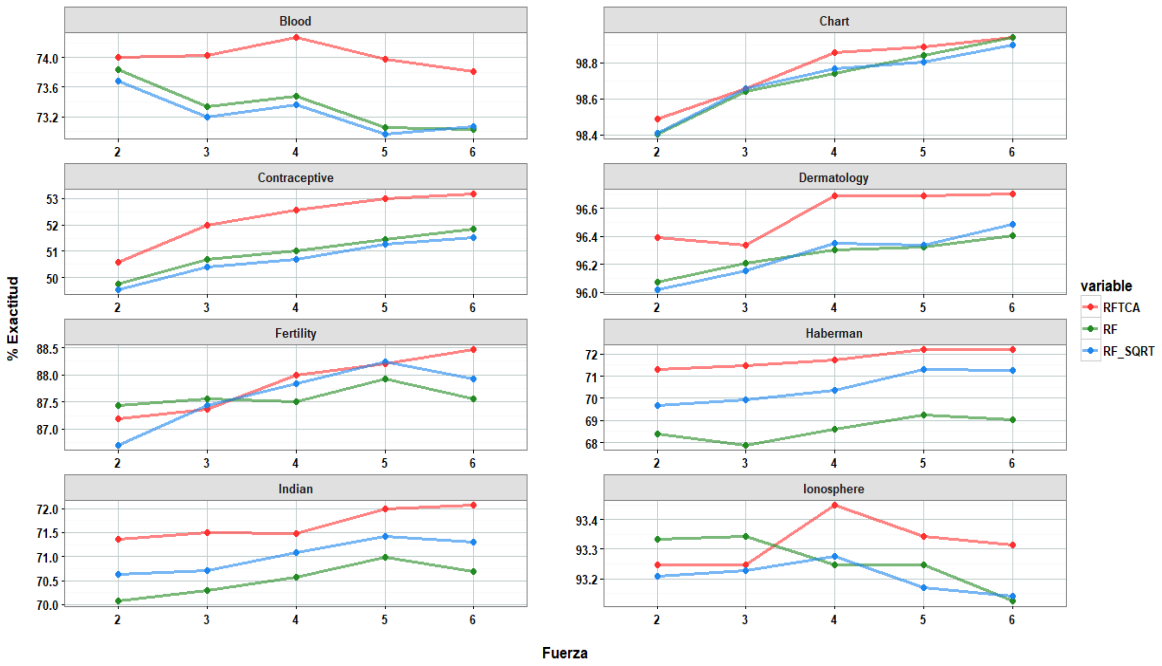

Figura 13. Parte 1 de los resultados más altos en términos de Exactitud para la propuesta RFTCA

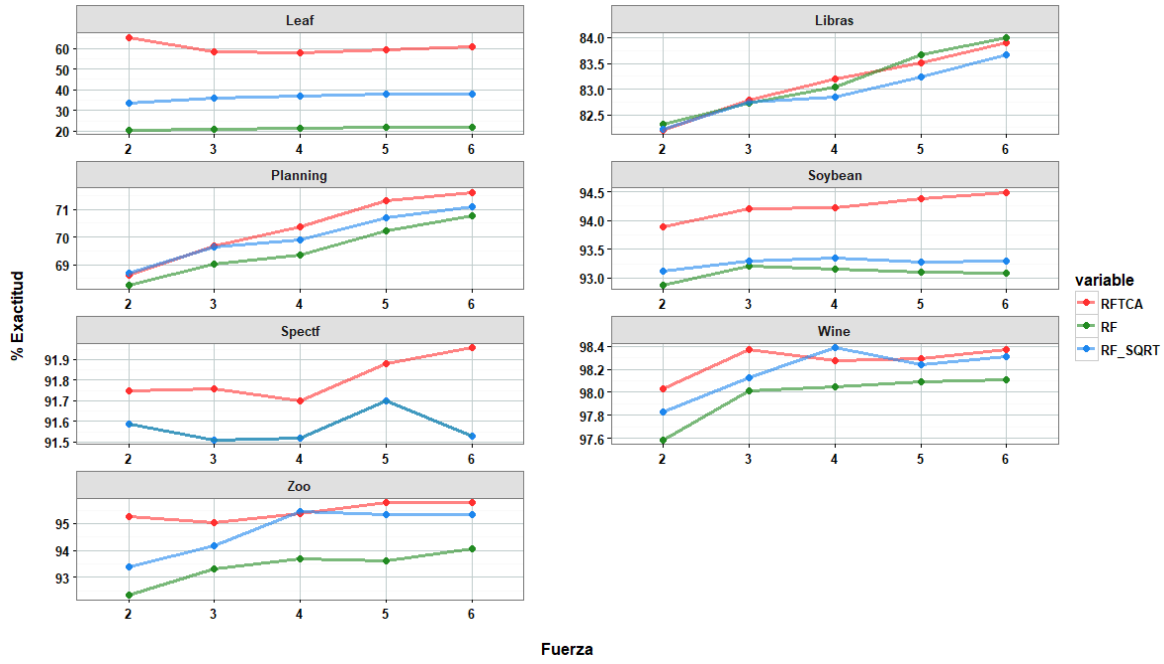

Figura 14. Parte 2 de los resultados más altos en términos de Exactitud para la propuesta RFTCA

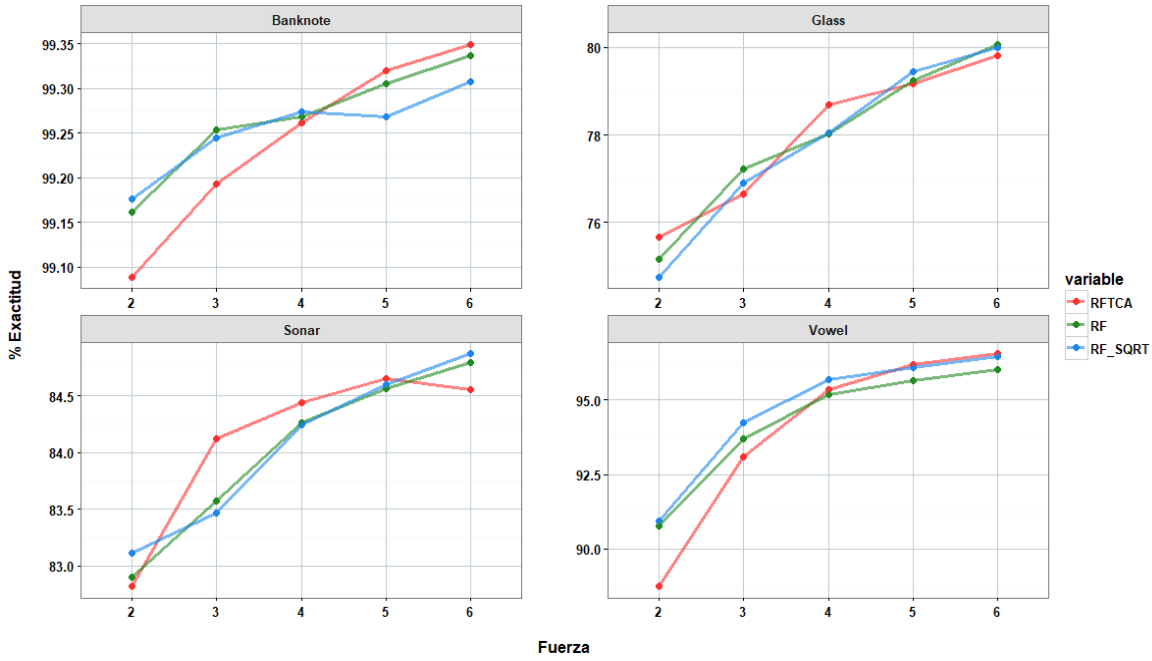

Figura 15. Resultados en términos de Exactitud donde la propuesta RFTCA obtiene resultados similares a los del Estado de Arte

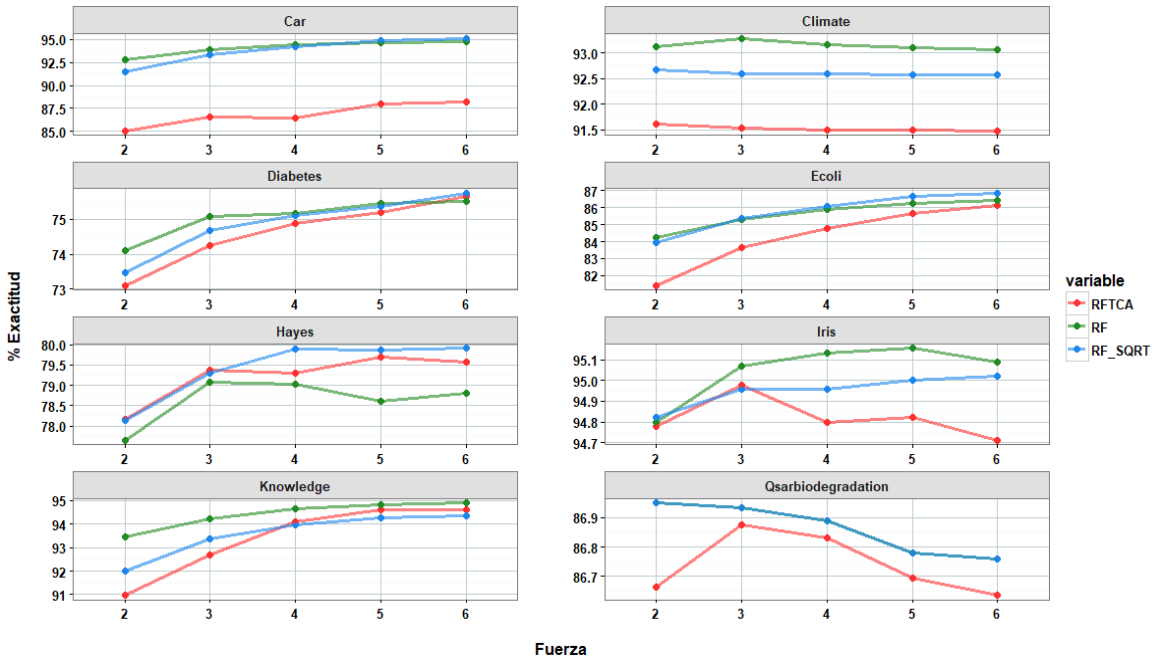

Figura 16. Parte 1 de los resultados más bajos en términos de Exactitud para la propuesta RFTCA

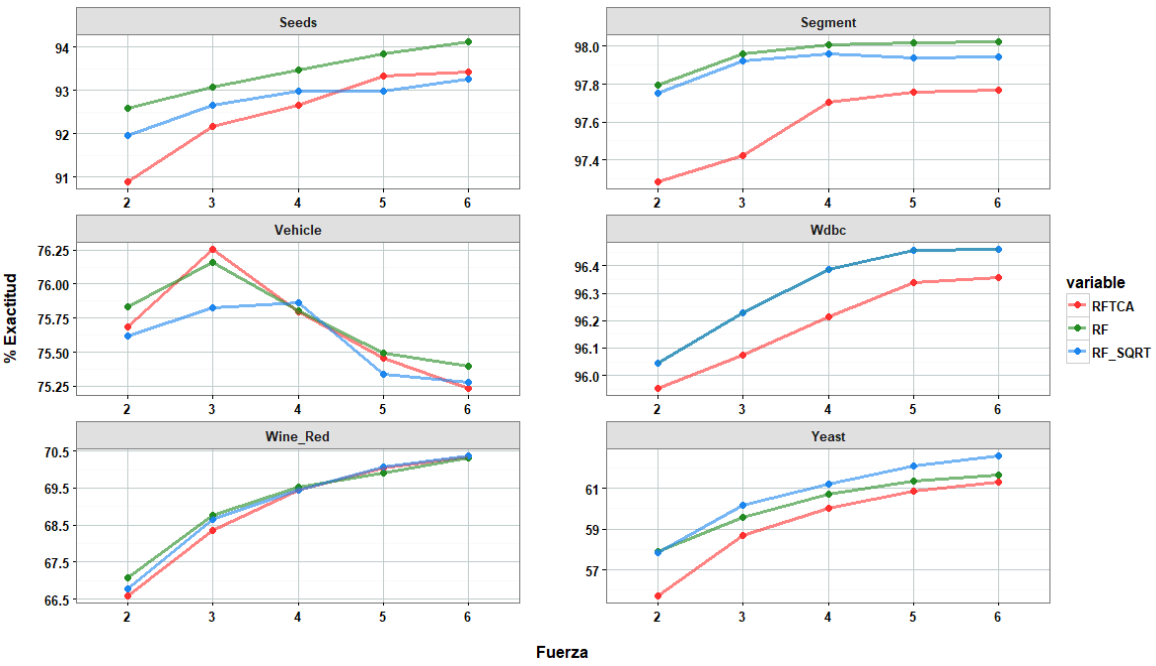

Figura 17. Parte 2 de los resultados más bajos en términos de Exactitud para la propuesta RFTCA

En total se obtuvieron 165 escenarios (33 conjuntos de datos por 5 Valores de Fuerza del CA) de evaluación del algoritmo propuesto con respecto a los del estado de arte. En la **Tabla 5,** 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(165) se observa que el algoritmo RFTCA se desempeña mejor en aproximadamente el 44,85% 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
	RFTCA	RF	RF SQRT	RF y RF_SQRT	RFTCA y RF	
2	13 (39,39%)	13 (39,39%)	5 (15,15%)	2 (6,06%)	0	33
3	15 (45,45%)	12 (36,36%)	4 (12,12%)	2 (6,06%)	0	33
4	15 (45,45%)	8 (24,24%)	8 (24,224%)	2 (6,06%)	0	33
5	16 (48,48%)	8 (24,24%)	7 (21,21%)	2 (6,06%)	0	33
6	15 (45,45%)	8 (24,24%)	7 (21,21%)	2 (6,06%)	1 (3,03%)	33
# Total Escenarios	74 (44,85%)	49 (29,70%)	31 (18,79%)	10 (6,06%)	1 (0,61%)	165

Tabla 5. Resumen de resultados para exactitud en RFTCA y RF Y RF_SQRT

La **Tabla 6** muestra el resumen de resultados tras contrastar la propuesta RFTCA con el algoritmo referencia de Breiman(RF).

Fuerza	Gana		Empata	# Conjuntos de Datos
	RFTCA	RF	RFTCA y RF	
2	14 (42,42%)	19 (57,58%)	0 (0%)	33
3	16 (48,48%)	17 (51,52%)	0 (0%)	33
4	19 (57,58%)	14 (42,42%)	0 (0%)	33
5	19 (57,58%)	14 (42,42%)	0 (0%)	33
6	18 (54,55%)	14 (42,42%)	1 (3,03%)	33
# Total Escenarios	86 (52,12%)	78 (47,27%)	1 (0.61%)	165

Tabla 6. Resumen de resultados para exactitud en RFTCA y RF

La propuesta RFTCA supera a RF con los valores de fuerza 4,5, y 6, destacando principalmente los resultados para fuerza 4 y 5 (57,58%). En cuanto al número total de escenarios (165), el algoritmo RFTCA se desempeña mejor en 86 (52,12%) de los escenarios de evaluación, el porcentaje restante del total de escenarios se reparte entre los dos algoritmos del estado del arte.

La **Tabla 7** muestra el resumen de resultados tras contrastar la propuesta RFTCA con el algoritmo RF_SQRT.

Fuerza	Gana		Empata	# Conjuntos de Datos
	RFTCA	RF_ SQRT	RFTCA y RF_SQRT	
2	16 (48,48%)	17 (51,52%)	0 (0%)	33
3	17 (51,52%)	16 (48,48%)	0 (0%)	33
4	17 (51,52%)	16 (48,48%)	0 (0%)	33
5	20 (60,61%)	13 (39,39%)	0 (0%)	33
6	19 (57,58%)	14 (42,42%)	0 (0%)	33
# Total Escenarios	89 (53,94%)	76 (46,06%)	0 (0%)	165

Tabla 7. Resumen de resultados para exactitud en RFTCA 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 RFTCA supera a RF-SQRT, en especial, al utilizar fuerza 5 (60,61%) y 6 (57,58%). Tomando en cuenta el número total de escenarios (165), el algoritmo RFTCA se desempeña mejor en 83 (53,94%) de los escenarios de evaluación, el porcentaje restante del total de escenarios se reparte entre los dos algoritmos del estado del arte.

De acuerdo a los resultados de la **Tabla 5**,**Tabla 6** y **Tabla 7** se deduce que el valor de fuerza que mejores resultados obtiene en términos de exactitud en el algoritmo RFTCA es la fuerza 5. Asimismo, el valor de fuerza que obtiene los resultados más bajos en el algoritmo RFTCA es la fuerza 2.

Tomando los resultados del test estadístico no paramétrico de Friedman se obtiene el ranking de la **Tabla 8**, en el que se confirma el mejor desempeño de RFTCA 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. Los resultados del test de Wilcoxon para la exactitud presentados en la **Figura 18**, exponen que no se presenta dominancia entre los algoritmos.

Algoritmo	Ranking
RFTCA	**1.9364 (1)**
RF-SQRT	2.0091 (2)
RF	2.0545 (3)
Valor de p	0.55634 (chi-cuadrado con 2grados de libertad: 1.172727)

Tabla 8. Test de Friedman para exactitud en RFTCA y RF Y RF-SQRT

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

Figura 18. Test de Wilcoxon para exactitud en RFTCA 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 RFTCA por un amplio margen, mientras que en Leaf, RFTCA 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 RFTCA (ver el lado izquierdo de la **Figura 19**).

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 TCA. En este conjunto de datos, desde el valor más bajo de fuerza (2) hasta el más alto (7) se presenta una considerable diferencia de RFTCA respecto a RF y RF_SQRT (ver el lado derecho de la **Figura 19**).

En la **Figura 20** 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 TCA genera en promedio un mejor resultado en términos de exactitud.

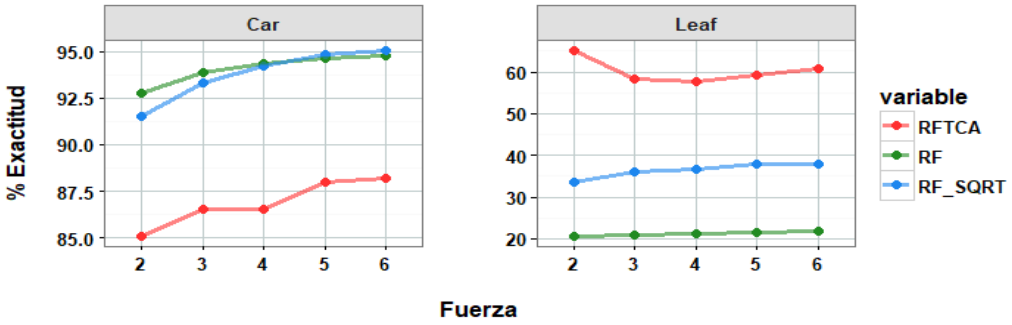

Figura 19. Exactitud en Conjuntos de Datos Car y Leaf

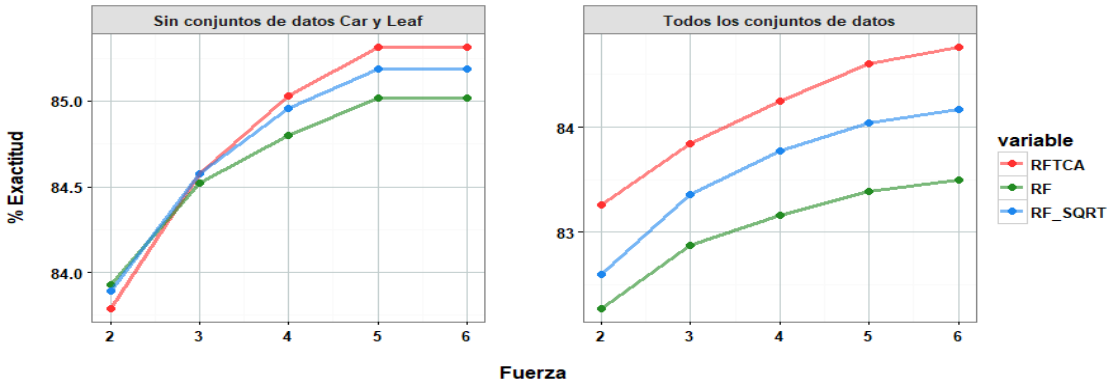

Figura 20. Promedio de Exactitud sobre todas las fuerzas

3.1.2.2 Resultados y análisis para la medida F en RFTCA

Los resultados en términos de medida F presentados en las **figuras 21** y **22** son los conjuntos de datos donde la propuesta RFTCA obtiene mejores resultados con respecto a los algoritmos del estado del arte.

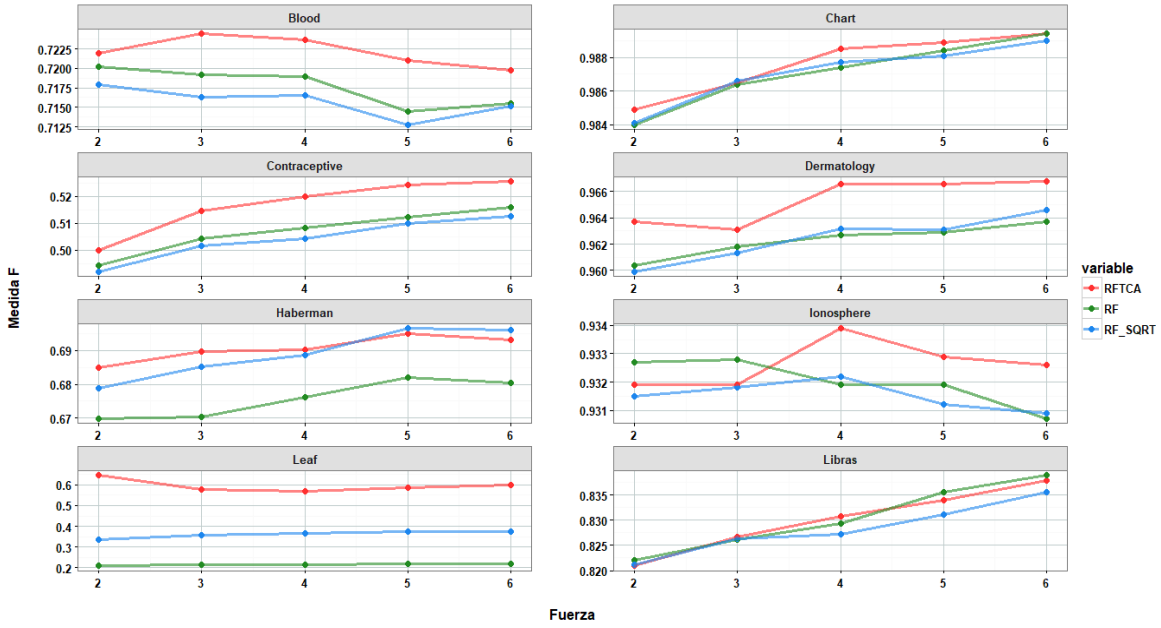

Figura 21. Parte 1 de los resultados más altos en términos de Medida F para la propuesta RFTCA

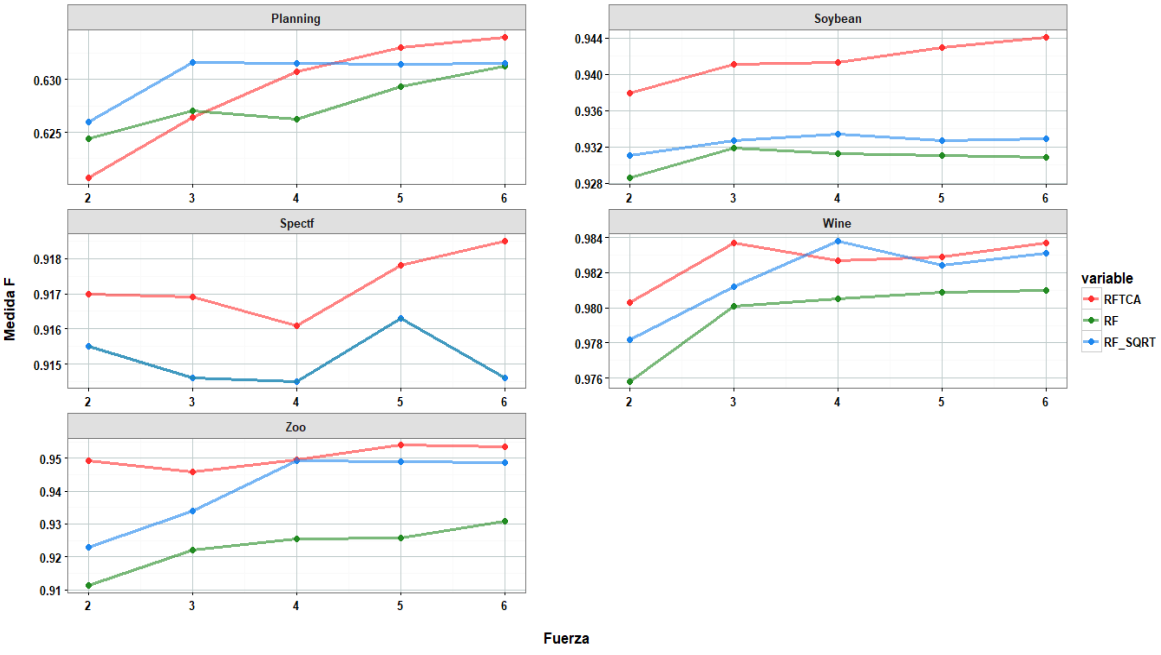

Figura 22. Parte 2 de los resultados más altos en términos de Medida F para la propuesta RFTCA

La **figura 23** corresponde a los conjuntos de datos donde la propuesta RFTCA obtiene resultados similares a los del estado del arte

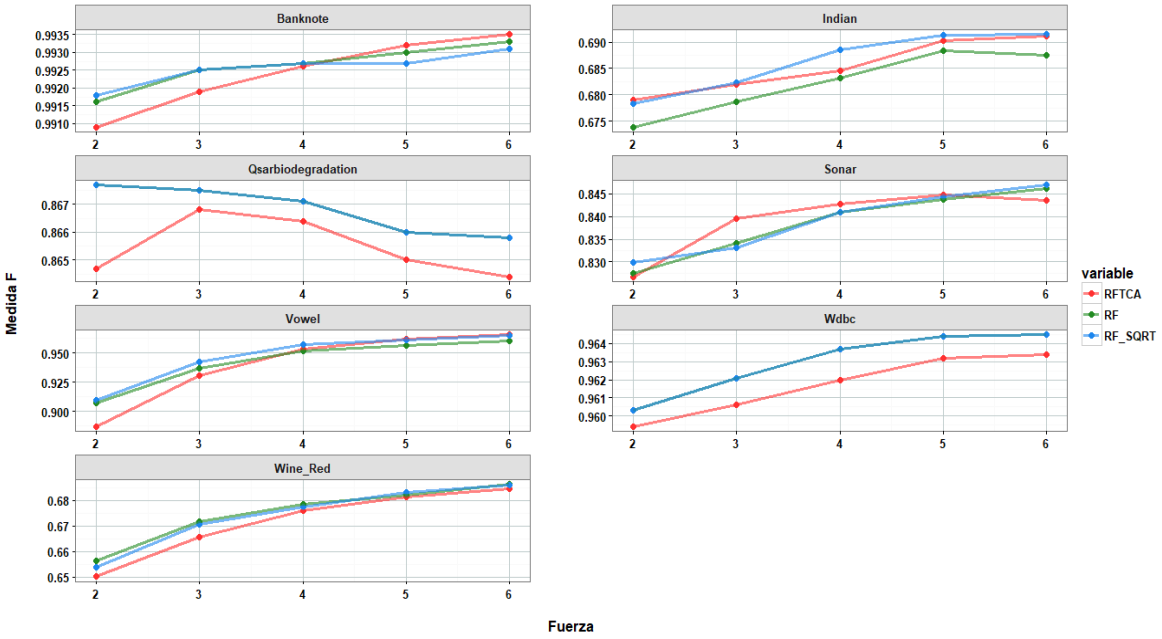

Figura 23. Resultados en términos de Medida F donde la propuesta RFTCA obtiene resultados similares a los del Estado de Arte

Las figuras 24 y **25** son los conjuntos de datos donde la propuesta RFTCA obtiene resultados más bajos

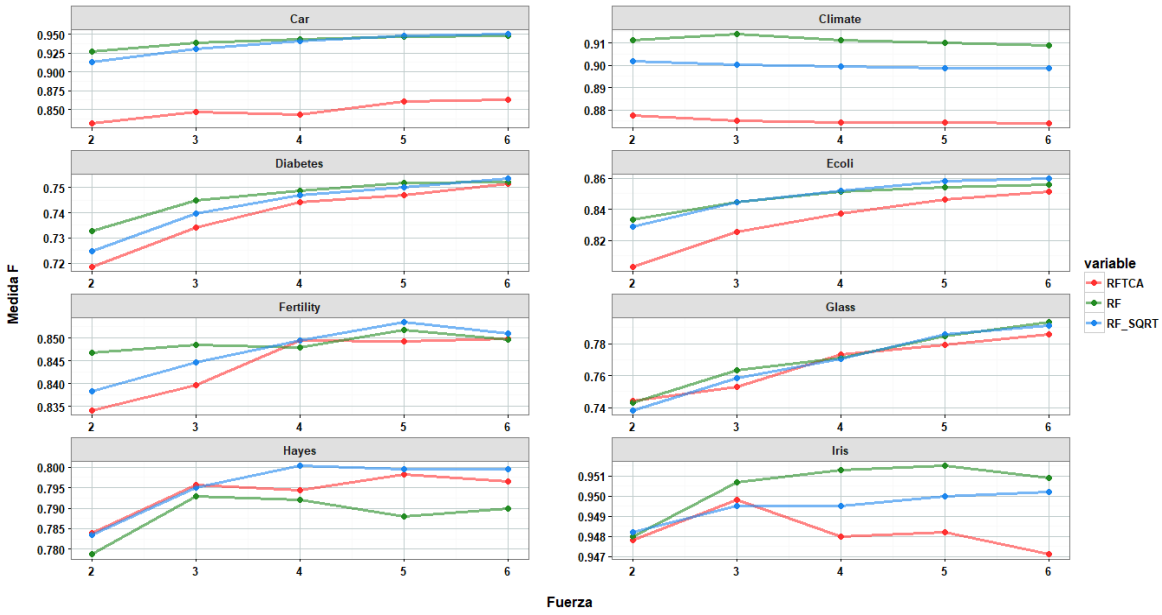

Figura 24. Parte 1 de los resultados más bajos en términos de Medida F para la propuesta RFTCA

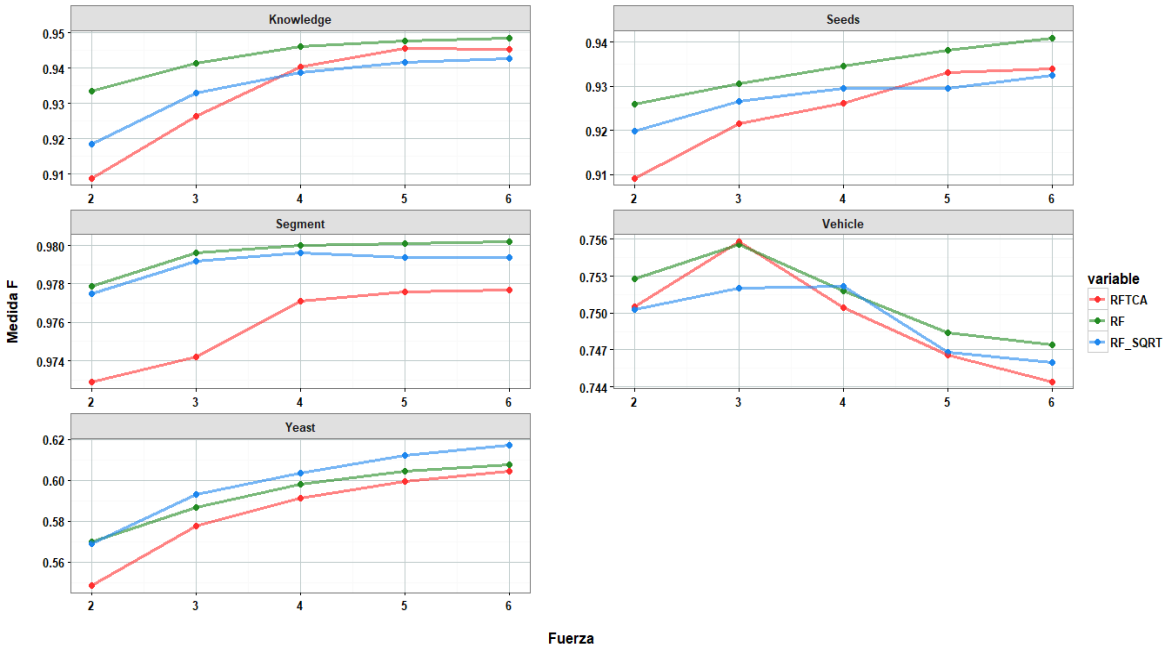

Figura 25. Parte 2 de los resultados más bajos en términos de Medida F para la propuesta RFTCA

En total se tienen 165 escenarios (33 conjuntos de datos por 5 Valores de Fuerza del TCA) de evaluación del algoritmo propuesto con respecto a los del estado de arte. La **Tabla 9** 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(165) se observa que el algoritmo RFTCA se desempeña mejor en 39,39% 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
	RFTCA	RF	RF SQRT	RF y RF_SQRT	RFTCA y RF	
2	13 (39,39%)	13 (39,39%)	5 (15, 15%)	2 (6,06%)	0	33
3	13 (39,39%)	11 (33,33%)	6 (18,18%)	3 (9,09%)	0	33
4	13 (39,39%)	8 (24,24%)	8 (24,24%)	3 (9,09%)	1 (3,03)	33
5	14 (42,42%)	8 (24,24%)	9 (27,27%)	2 (6,06%)	0	33
6	12 (36,36%)	9 (27,27%)	7 (27,27%)	2 (6,06%)	1 (3,03%)	33
# Total Escenarios	65 (39, 39%)	49 (29,70%)	37 (22,42%)	12 (7,27%)	2 (0,12%)	165

Tabla 9. Resumen de resultados para Medida F en RFTCA 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 RFTCA supera a RF-SQRT, en especial, al utilizar fuerza 5 en RFTCA. Por otra parte, RFTCA supera a RF con los valores de fuerza 3,4,5 y 6, destacando principalmente los resultados para fuerza 5.

Por consiguiente, de acuerdo a los resultados de la **Tabla 9** se deduce que, para los resultados en Medida F, el valor de fuerza que mejores resultados obtiene en el algoritmo RFTCA 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 10**, en el cual se evidencia que en términos de la medida F y con respecto a RFTCA y RF , el algoritmo RF_SQRT obtiene los mejores resultados 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 26** 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-SQRT	**1.9545**	**(1)**
RF	2.003	(2)
RFTCA	2.0424	(3)
Valor de p	0.726369 (chi-cuadrado con 2 grados de libertad: 0.639394)	

Tabla 10. Test de Friedman para Medida F en RFTCA y RF Y RF-SQRT

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

Figura 26. Test de Wilcoxon para medida F en RFTCA y RF Y RF-SQRT

3.1.2.3 Resultados y análisis de tiempo de ejecución

Los resultados de tiempos de ejecución se obtuvieron conforme al procedimiento descrito en la sección 5.6 de la Monografía. El tiempo de ejecución de un algoritmo se considera desde el instante que comienza el proceso de creación del modelo de clasificación (RFTCA y algoritmos del Estado del Arte), la medición del tiempo finaliza cuando se competa la construcción del modelo.

La **Tabla 11** muestra el tiempo total de ejecución (suma de los tiempos de ejecución para los 5 valores de fuerza), en milisegundos(Mili-Seg) y segundos(Seg) con cada algoritmo para cada conjunto datos.

Conjuntos de Datos	RFTCA (Mili-Seg)	RFTCA (Seg)	RF (Mili-Seg)	RF (Seg)	RF_SQRT (Mili-Seg)	RF_SQRT (Seg)
Banknote	511,23	0,51	465,63	0,47	363,93	0,36
Blood	409,70	0,41	414,50	0,41	324,47	0,32
Car	189,37	0,19	252,57	0,25	282,13	0,28
Chart	3535,90	3,54	3454,47	3,45	3890,37	3,89
Climate	1158,10	1,16	1009,13	1,01	883,67	0,88

Contraceptive	593,60	0,59	781,50	0,78	709,63	0,71
Dermatology	453,40	0,45	409,83	0,41	413,23	0,41
Diabetes	552,50	0,55	507,17	0,51	319,17	0,32
Ecoli	184,70	0,18	158,20	0,16	127,57	0,13
Fertility	34,40	0,03	30,83	0,03	28,73	0,03
Glass	177,60	0,18	163,50	0,16	135,23	0,14
Haberman	80,30	0,08	68,80	0,07	60,50	0,06
Hayes	28,53	0,03	28,63	0,03	27,03	0,03
Indian	429,47	0,43	411,60	0,41	332,73	0,33
Ionosphere	1172,23	1,17	1125,80	1,13	993,50	0,99
Iris	34,40	0,03	31,07	0,03	26,97	0,03
Knowledge	124,30	0,12	98,90	0,10	88,33	0,09
Leaf	1619,30	1,62	1548,57	1,55	1229,57	1,23
Libras	3437,03	3,44	3260,83	3,26	3925,23	3,93
Planning	232,47	0,23	210,73	0,21	175,23	0,18
QSARBiodegradation	4335,07	4,34	4059,27	4,06	4075,80	4,08
Seeds	104,63	0,10	93,40	0,09	75,07	0,08
Segment	5152,83	5,15	4553,63	4,55	3882,97	3,88
Sonar/Connectionist	970,00	0,97	918,80	0,92	1025,20	1,03
Soy Bean	2583,73	2,58	3727,20	3,73	3634,50	3,63
Spectf	1407,43	1,41	1346,13	1,35	1352,33	1,35
Vowel	1576,27	1,58	1497,50	1,50	1282,43	1,28
Vehicle	1067,63	1,07	1021,20	1,02	827,77	0,83
Wdbc	998,67	1,00	977,03	0,98	986,30	0,99
Wine	106,63	0,11	94,57	0,09	82,53	0,08
Wine Red	1488,43	1,49	1390,60	1,39	1115,60	1,12
Yeast	1352,53	1,35	1267,40	1,27	796,43	0,80
Zoo	58,70	0,06	55,20	0,06	56,07	0,06
Tiempo total de ejecución	36161,08	36,15	35434,19	35,44	33530,22	33,55

Tabla 11. 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, Soy Bean, los cuales tuvieron los valores más altos. También, mediante la **ecuación (3)** se evaluó el porcentaje de diferencia en términos de tiempo total de ejecución en segundos, entre los algoritmos RFTCA y los del estado del arte.

$$Porcentaje\ Diferencia\ = \left(\frac{R_r - R_e}{R_r}\right) * 100\% \quad (3)$$

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

Porcentaje de diferencia Algoritmo RFTCA con respecto al algoritmo referencia de Breiman (RF)

$$Porcentaje\ Diferencia\ = \left(\frac{35{,}44 - \ 36{,}15}{35{,}44}\right) * 100\% = \ -2\%$$

Porcentaje de diferencia Algoritmo RFTCA con respecto al algoritmo RF_SQRT

$$Porcentaje\ Diferencia\ = \left(\frac{33{,}55\ - 36{,}15}{33{,}55}\right) * 100\% = \ -7{,}75\%$$

Los valores en el porcentaje de diferencia del Algoritmo RFTCA con respecto al algoritmo referencia de Breiman (RF) y RF_SQRT fueron de -2% y $-7{,}75\%$ 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, Contraceptive y Soy Bean obtuvo tiempos de ejecución más bajos. En general, los resultados en los tres algoritmos, en términos de tiempo de ejecución son cercanos, y el aumento o disminución de tiempo en los resultados, posiblemente depende más de los aspectos propios de un conjunto de datos, como los son: número de clases, número de instancias, numero de características y tipos de características (discretas, continuas, nominales, ordinales).

4 BIBLIOGRAFÍA

[1] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[2] H. Parmar, S. Bhanderi, and G. Shah, “Sentiment Mining of Movie Reviews using Random Forest with Tuned Hyperparameters,” 2014.
[3] 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].
[4] L. Breiman, “Bagging predictors,” *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, vol. p. 1984.
[6] 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.
[7] P. Geurts, D. Ernst, and L. Wehenkel, “Extremely randomized trees,” *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006.
[8] 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.