CINVESTAV
Unidad Tamaulipas

Implementación del Método de Random Forest para Problemas de Clasificación Binaria

Cómputo Paralelo

por
Hansel Giuseppe Rodríguez Flores

Profesor:
Dr. Mario Garza Fabre

Febrero, 2021

Índice General

1 Introducción

Durante las últimas décadas, uno de los campos con mayor crecimiento y estudio dentro de las ciencias computacionales es el del aprendizaje máquina, o *machine learning* como se conoce en inglés.

Este campo se encarga del desarrollo de técnicas que permitan que una computadora pueda replicar el proceso de aprendizaje llevado a cabo por los seres humanos; es decir, que la máquina sea capaz de realizar una tarea y adquirir una mayor experiencia al realizarla, de tal forma que ofrezca cada vez mejores resultados [1].

De hecho, de igual manera que con los humanos, existen distintas técnicas de aprendizaje que se pueden aplicar dentro del campo de machine learning. Estas técnicas se pueden clasificar dentro de dos grandes grupos o estilos de aprendizaje: (1)*aprendizaje supervisado* y (2) *apendizaje no supervisado*.

En los problemas de aprendizaje supervisado se toma como entrada un conjunto de datos de entrenamiento $D = \{(x^n, y^n),\ n = 1, 2, \cdots, N\}$ de tal forma que la máquina aprenda la relación entre las variables x y y. Una vez realizado esto, al ingresarse un nuevo conjunto de datos, la máquina debe ser capaz de predecir el valor de y^* dado un valor de x^* [2].

Por otro lado, para los problemas de aprendizaje no supervisado, se toma como entrada un conjunto de datos $D = \{x^n,\ n = 1, 2, \cdots, N\}$ de tal forma que la máquina pueda estudiar, por ejemplo, la distribución de la variable x para obtener una forma sintetizada de dichos datos [3].

Otro tipo de problemas que se pueden trabajar dentro de esta área es el de *aprendizaje por refuerzo*. En estos problemas al algoritmo no se le muestra un conjunto de datos que sirva de entrenamiento para encontrar relaciones entre una variable x y una variable y, sino que, a diferencia del aprendizaje supervisado, el algoritmo debe ser capaz de encontrar estas relaciones mediante un proceso de prueba y error [4].

Las técnicas de ML son utilizadas principalmente para las tareas de minería de datos; es decir, el campo de ML proporciona las técnicas para el análisis de los datos en la búsqueda de patrones y regularidades para extraer información que pueda ser útil para diversos propósitos [5].

Para la realización de lo anterior se suelen emplear distintas técnicas tales como redes neuronales, árboles de decisión, redes bayesianas, entre otras; y principalmente se usan para resolver problemas de clasificación, clustering, regresión, entre otros.

Uno de los métodos más conocidos de machine learning es el de *random forest*, el cual es un método basado en los árboles de decisión y que es utilizado principalmente para resolver problemas de clasificación o de regresión.

En el caso de este trabajo, se estudia precisamente el uso e implementación de dicho método para resolver problemas de clasificación en su versión más simple, es decir, problemas de clasificación binaria; los cuales, a pesar de su simplicidad, sirven para aplicarse en una gran cantidad de situaciones en problemas reales. En las siguientes secciones se hace una descripción de este tipo de problemas, así como del funcionamiento de este algoritmo para su resolución, y se estudia, al mismo tiempo, la influencia de distintos parámetros en el funcionamiento de este método.

2
Descripción del Problema

2.1 El Problema de Clasificación

La clasificación se refiere a una tarea en la que el objetivo principal, consiste en asignar a un objeto dentro de una categoría, basada en las diversas características de dicho objeto; si el objeto cumple en general con las características que deberían tener para una cierta categoría, entonces es reconocido como un objeto de algún tipo en específico.

Un ejemplo sencillo de esta tarea es el de, dada una fruta, categorizarla de acuerdo con su tipo; es decir, basándose en sus características tales como el olor, color, tamaño y/o sabor, se puede decir si dicha fruta es una naranja, manzana, pera o de otro tipo.

De manera más formal, el problema de clasificación consiste en tomar como entrada un conjunto de datos compuesto por diversas *instancias*, las cuales se caracterizan por una tupla de *atributos* $(\mathbf{x}, y)$, donde $\mathbf{x}$ corresponde al conjunto de atributos y y se refiere a un atributo especial llamado *clase* [6], el cual indica la categoría a la que pertenece la instancia de acuerdo con los distintos atributos del conjunto $\mathbf{x}$.

Como ya se mencionó, cada uno de los atributos del conjunto $\mathbf{x}$ corresponden a una característica del objeto, y pueden tomar valores numéricos o categóricos, discretos o continuos, nominales u ordinales, etc. Por otro lado, y como una característica muy importante en los problemas de clasificación, la clase y toma valores necesariamente discretos.

Los problemas de clasificación se pueden dividir en dos tipos, dependiendo de las características del conjunto de clases que se tenga: clasificación binaria y clasificación multiclase.

2.1.1 Clasificación Binaria

La clasificación binaria es el problema de clasificación más simple y, al mismo tiempo, el más estudiado y utilizado para el desarrollo y mejora de técnicas de machine learning debido a su simplicidad

[7] [8] [9] [10] [11] [12].

Este problema de clasificación es llamado de esta manera debido a que el conjunto de clases **y** está compuesto solamente por 2 elementos, usualmente $y \in \{0, 1\}$.

Algunos ejemplos de este tipo de problemas pueden ser:

- Decidir si se aprueba un crédito o no a una persona, basándose en su historial crediticio [13].
- Predecir si lloverá o no al analizarse datos como la dirección y velocidad del viento, temperaturas máxima y mínima, etc [14].
- Determinar si un tumor es benigno o maligno con base en las imágenes de radiografías [15].
- Entre otros.

2.1.2 Clasificación Multiclase

La clasificación multiclase podría verse como una extensión del problema de clasificación binaria, sin embargo, en este caso se tiene que $y \in \{1, 2, \cdots n\}$; es decir, el conjunto de clases está conformado por más de 2 elementos.

El problema general es bastante similar al anterior; evaluando el conjunto de atributos x para un objeto, determinar a cuál de las n clases pertenece.

Este tipo de problemas es estudiado en casos tales como la determinación del tipo de cáncer que presenta un paciente [16], determinar la causa de muerte de acuerdo con la información forense [17], o para la detección de fallas y el tipo de falla en pozos de petróleo y líneas de producción [18].

2.2 Proceso General para Resolver un Problema de Clasificación

2.2.1 Generación del Modelo de Clasificación

Para solucionar un problema de clasificación se pueden encontrar una gran cantidad de técnicas de distintos tipos, sean estadísticas [8], de árboles de decisión y random forest [18], de programación genética [10], entre otras técnicas de machine learning e inteligencia computacional [9] [13] [14] [15].

A pesar de la gran variedad de técnicas existentes para esta tarea, se puede establecer un proceso general para encontrar soluciones a los problemas de clasificación [6].

Sin importar la técnica que sea utilizada, para resolver el problema se emplea un algoritmo de aprendizaje, el cual toma un conjunto de datos como entrada (conocido como conjunto de entrenamiento o *training set*) compuesto por diversas instancias, de las cuales se sabe a qué clase pertenecen. A partir de dicho conjunto, el algoritmo debe encontrar un modelo de clasificación que permita relacionar de la mejor manera posible al conjunto de atributos con el conjunto de clases que se tienen.

El modelo generado por el algoritmo de aprendizaje, debe ser lo suficientemente general para que sea capaz de tomar un nuevo conjunto de datos (conjunto de prueba o *test set*) con clases desconocidas, y predecir, precisamente, a qué clase pertenece cada instancia al evaluar su conjunto de atributos.

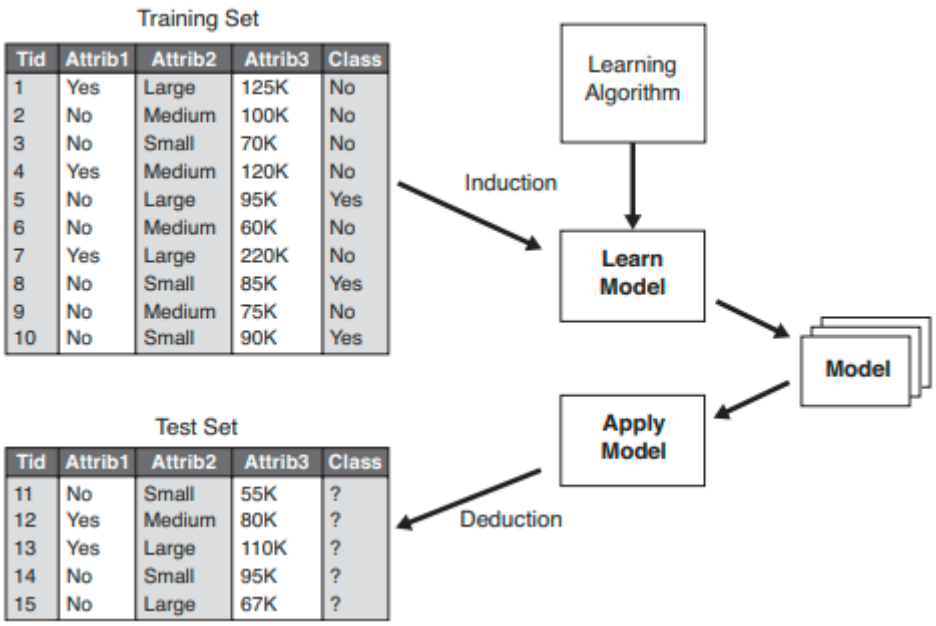

Figura 2-1: Proceso Para la Generación de un Modelo de Clasificación.

En la Figura 2-1 se puede observar el proceso general que se sigue para la generación de un modelo de clasificación; en él se puede observar que se toma al conjunto para la etapa de entrenamiento y se aplica el algoritmo de aprendizaje, el cual genera al modelo de clasificación. Al tomarse un nuevo conjunto con clases desconocidas para la etapa de prueba y validación, se aplica el modelo para predecir a qué clase pertenece cada instancia del conjunto de datos.

2.2.2 Evaluación del Desempeño del Modelo de Clasificación

Una vez generado y aplicado un modelo de clasificación a un conjunto de pruebas, se debe proceder a evaluar el desempeño de dicho modelo el cual se basa en el conteo de predicciones correctas e incorrectas en la etapa de pruebas.

Para realizar este proceso de evaluación, existen dos métricas que pueden ser utilizadas de manera indistinta; la precisión o *accuracy* y la tasa de error o *error rate*, los cuales toman valores entre 0 y 1 y se definen de la siguiente manera

$$\text{Precisión} = \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones}} \tag{1}$$

$$\text{Tasa de Error} = \frac{\text{Número de predicciones erróneas}}{\text{Número total de predicciones}} \tag{2}$$

Medir estos parámetros es importante cuando se trabaja en modelos de clasificación, debido a que lo que se busca es que el valor de la precisión sea lo más alta posible, mientras que el de la tasa de error

sea lo más baja posible. Si se trabaja con varios modelos o técnicas para la solución de los problemas de clasificación, entonces siempre se deberá quedar con aquel(la) que ofrezca el mejor desempeño.

Dentro del gran número de técnicas que pueden ser utilizadas para resolver este tipo de problemas, en este trabajo se estudia en particular el método de random forest. Este método se basa en las predicciones realizadas a partir de árboles de decisión, por lo que resulta fundamental conocer el funcionamiento de este método antes de estudiar el algoritmo de random forest.

En la siguiente sección se explica el proceso general mediante el cual se resuelven los problemas de clasificación a partir de los árboles de decisión, así como el funcionamiento general e implementación del algoritmo de random forest.

3 Algoritmo Secuencial

3.1 Funcionamiento General del Algoritmo

3.1.1 Clasificación Mediante Árboles de Decisión

El uso de árboles de decisión, comúnmente llamados árboles de clasificación y regresión (*classification and regression trees*) es una de las técnicas más comunes para resolver problemas de clasificación, por lo que estos han sido estudiados durante un número importante de años [19].

Un árbol está compuesto por un conjunto de nodos y aristas organizado de forma jerárquica; es decir, los nodos pueden encontrarse en distintos niveles, de tal forma que cuando un nodo apunta a otros nodos en niveles inferiores, entonces se les llama nodos padres e hijos, respectivamente. El nodo padre que no tiene ninguna arista incidente en él, se conoce como nodo raíz; mientras que aquellos nodos que no tienen ningún hijo, son llamados nodos hoja. Todos aquellos nodos que tienen padres e hijos al mismo tiempo, son llamados nodos internos.

En el caso de los árboles de decisión, todos los nodos hoja representan a una clase perteneciente al conjunto de clases del conjunto de datos. Por otro lado, los nodos internos y raíz contienen ciertas condiciones de prueba que permiten identificar y separar a aquellas instancias que tienen ciertas características [6].

El funcionamiento general de los árboles de decisión consiste en evaluar cuáles instancias cumplen con las condiciones establecidas en cada nodo interno, partiendo desde condiciones generales hasta condiciones más específicas; dependiendo del resultado de la evaluación, entonces se procede a evaluar otra condición más específica. Cuando se llega a un nodo hoja, entonces esto indica que se han evaluado las condiciones necesarias que permiten identificar a qué clase pertenecen los objetos evaluados.

Una forma para entender el funcionamiento de esta técnica, podría ser mediante el ejemplo visto en la Figura 3-2, tomado de [19]. En la parte izquierda de la figura se puede observar una gráfica en la que el conjunto de datos está conformado por la tupla $(\mathbf{X}, y)$, donde $\mathbf{X} = \{X_1, X_2\}$ y $y \in \{1, 2, 3\}$. Es decir, cada una de las instancias está conformada por un conjunto de 2 atributos o variables, y puede pertenecer a una de las tres clases distintas que se tienen. Al construirse el modelo de clasificación se

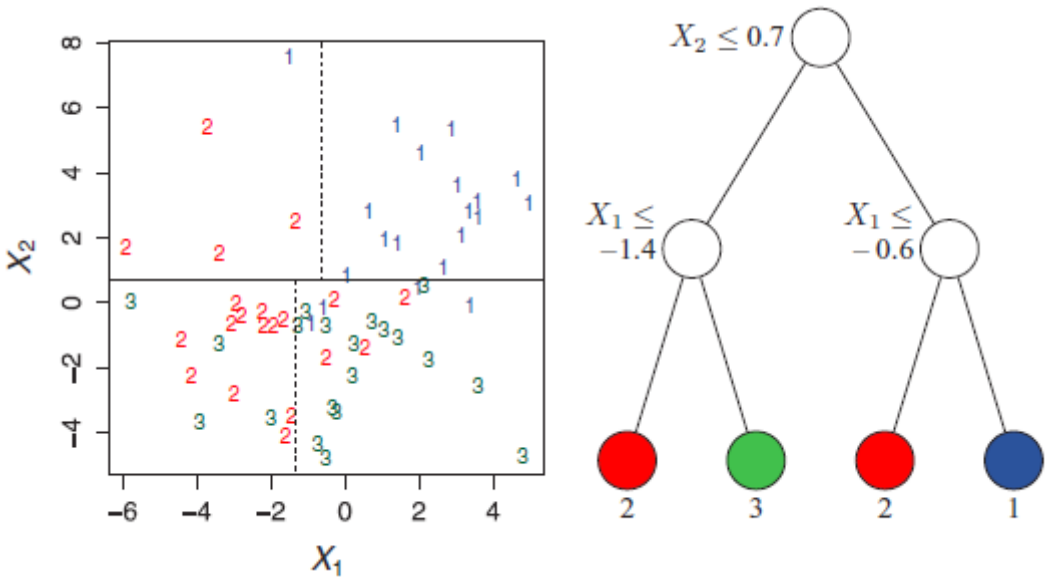

Figura 3-2: Ejemplo del Funcionamiento de un Árbol de Decisión.

encontró que la mayoría de los datos que pertenecen a la clase 1 tienen valores en $X_1 > -0{,}6$ y en $X_2 > 0{,}7$; los elementos que pertenecen a la clase 2 cuentan, en su mayoría, con valores de $X_1 \leq -0{,}6$ cuando $X_2 > 0{,}7$, y de $X_1 \leq -1{,}4$ cuando $X_2 \leq 0{,}7$. Por otro lado, los elementos de la clase 3 comúnmente tienen valores de $X_1 > -1{,}4$ y de $X_2 \leq 0{,}7$. Si se observa la parte derecha de la figura, se puede observar el árbol de decisión construido a partir del modelo de clasificación generado; en él se tiene que el nodo raíz evalúa la condición del valor de X_2. Cuando $X_2 \leq 0{,}7$, entonces los datos son enviados al nodo hijo izquierdo, y si no cumplen con esa condición, entonces son enviados al nodo derecho; posteriormente se procede a evaluar el valor de X_1 de manera similar. Por ejemplo, en dado caso de que un objeto cumpla con la condición establecida en el nodo raíz, entonces se debe evaluar si su valor para $X_1 \leq -1{,}4$; en el caso de que la condición sea verdadera, se envía al nodo izquierdo, de caso contrario, se envía al nodo derecho. Dado que dichos nodos son nodos hoja, entonces el modelo ha evaluado condiciones suficientes para clasificar los datos dentro de las clases 2 y/o 3, dependiendo del valor de X_1.

3.1.2 Clasificación Mediante Random Forests

Una vez que se conoce la forma general en la que funcionan los árboles de decisión, es conveniente comenzar a entender el funcionamiento del método de random forest, el cual es el método estudiado en este trabajo.

Un random forest funciona a partir de las decisiones tomadas por los árboles para clasificación y regresión; estos árboles son, precisamente, aquellos árboles de decisión que funcionan para resolver problemas de clasificación y/o de regresión [19] [20]. La diferencia entre un problema de regresión y uno de clasificación, reside principalmente en que la predicción realizada en un problema de regresión corresponde a datos de tipo continuo [21], mientras que, como ya se mencionó, para los problemas de clasificación las predicciones realizadas son de datos de tipo discreto.

La tarea de clasificación mediante este método se realiza de forma que se tiene un conjunto de

árboles de decisión (bosque), cada uno de los cuales realiza una predicción de manera independiente. Al final, se evalúan las predicciones de cada uno de los árboles y, aquella que se repita un mayor número de veces, es la clase en la que se categoriza el objeto evaluado. Es decir, en un random forest se realiza un conteo de mayoría de votos de las predicciones realizadas por un número predeterminado de árboles de decisión individuales.

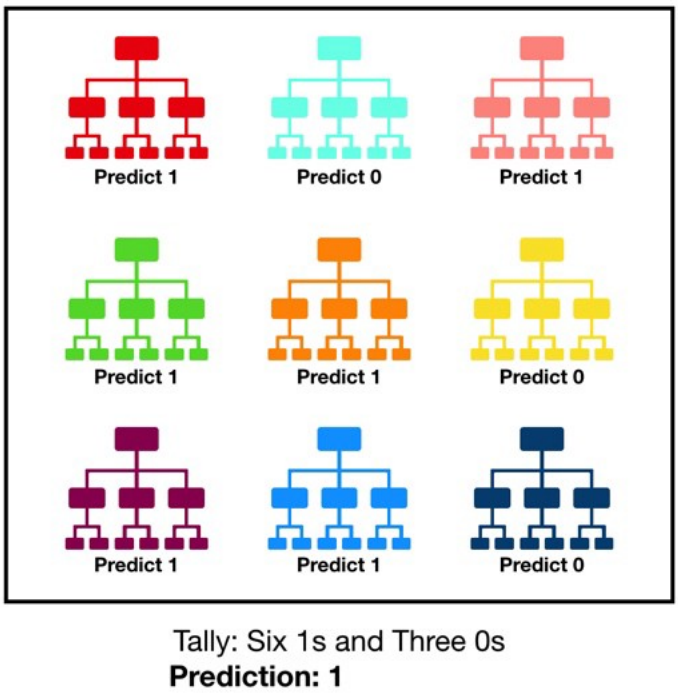

Figura 3-3: Ejemplo de Clasificación Mediante Random Forest.

En la Figura 3-3 se muestra un ejemplo de cómo se resuelve un problema de clasificación aplicando el método de random forest. En dicha figura se puede observar que se cuenta con un conjunto de 9 árboles de decisión, y cada uno realiza la tarea de clasificación de manera independiente, la cual consiste en categorizar una instancia dentro de las clases 0 o 1, realizándose una predicción por cada árbol. Al final, se tiene que 6 de los árboles categorizaron a la instancia dentro de la clase 1, mientras que los 3 restantes lo hicieron dentro de la clase 0; así, al realizarse un conteo de la mayoría de votos, el método clasifica finalmente a la instancia dentro de la clase 1.

3.2 Modelo e Implementación del Random Forest

Tal como se mencionó anteriormente, un aspecto muy importante en el método de random forest es el de la construcción y crecimiento de los árboles de decisión que lo conformarán; para ello existe un gran número de algoritmos, algunos de los cuales se encuentran reportados en [19] y en [6].

En el caso de este trabajo, la implementación general del método se basó en el trabajo realizado por [22]. A continuación se presenta el modelo utilizado para la construcción de los árboles basándose en el procedimiento descrito por [20], así como del funcionamiento del random forest.

3.2.1 Construcción de los Árboles de Decisión

El primer paso para la construcción de los árboles de decisión y, en general, de cualquier método para clasificación, es el de la selección del conjunto de datos que sirve para realizar la etapa de entrenamiento. La manera más sencilla y usada es la de, a partir del conjunto de datos original, tomar un porcentaje determinado de las instancias totales mediante un muestreo aleatorio, el cual servirá para el entrenamiento; mientras que el porcentaje restante se mantiene para la etapa de pruebas.

A partir del conjunto de datos para el entrenamiento, se procede a encontrar el modelo de clasificación que permita resolver este tipo de problemas.

Se debe recordar que el modelo general del funcionamiento de los árboles de decisión consiste en que cada nodo evalúa una condición que permita dividir el conjunto de datos en grupos que compartan ciertas características generales de acuerdo con las variables o atributos de cada instancia, hasta poder identificar las características generales propias de cada clase.

Para poder encontrar estas condiciones, se toma todo el conjunto del entrenamiento y, en el nodo raíz, se comienzan a realizar particiones en dos subconjuntos a partir de una variable seleccionada de manera aleatoria, durante un número determinado de veces; cada una de las particiones generadas se evalúa y se selecciona la que ofrezca una mejor división del conjunto de datos. La idea central es que la división resultante ofrezca una mayor "pureza" en los datos de cada subconjunto. Una forma de medir esta pureza, es mediante el *índice gini*, el cual se define de la siguiente manera

$$I_G = 1 - \sum_{j=1}^{c} p_j^2 \tag{3}$$

Donde I_G se refiere al índice gini, y p_j se refiere a la proporción de las instancias que pertenecen a la clase c. De dicha ecuación se puede observar que lo que se busca es obtener el menor valor posible del índice gini, por lo que esto se puede plantear como un problema de optimización en el que se busca minimizar la función de la ecuación (3). Así, al encontrarse la mejor división del conjunto original, uno de los subconjuntos es enviado al nodo derecho, mientras que el otro se envía al nodo izquierdo. Este proceso se repite de manera recursiva en cada nodo hasta que se cumpla una de las condiciones de parada.

Algorithm 3-1 BuildTree($TrainingSet[0,\cdots,n][0,\cdots,m], maxFeatures, minSamples, maxDepth$)

Input: Conjunto $D = \{(\mathbf{x}, y\})$
Output: Árbol de decisión T
1: Subsets = BestDataSplit($TrainingSet[0,\cdots n][0,\cdots,m], maxFeatures$)
2: Root = CreateNode()
3: T = GrowTree($Root, Subsets[0,1][0,1,\cdots,m], minSamples, maxDepth, 1$)
4: **return** T

Las condiciones de parada son parámetros a los que se debe poner especial atención, ya que lo que se busca es que el árbol de decisión generado ofrezca una alta precisión con el menor tamaño del árbol posible. Esto debido a que, para grandes cantidades de datos, los árboles pueden crecer de forma que se vuelven no sólo computacionalmente ineficientes, sino que también su precisión se puede ver afectada de manera negativa debido a una condición de *sobreentrenamiento*, la cual consiste en que el modelo

generado ofrece una muy buena clasificación de los datos de entrenamiento, pero que no alcanza una generalización que permita obtener una buena precisión con otros datos que no pertenezcan al conjunto con el que fue construido [6].

Algorithm 3-2 BestDataSplit($TrainingSet[0,\cdots n][0,\cdots,m], maxFeatures$)

Input: Conjunto $D = \{(\mathbf{x}, y)\}$
Output: Partición de datos $Subsets[0,1]$

```
1:  max = m − 2
2:  BestGini = 1
3:  leftCount = 0
4:  rightCount = 0
5:  for i = 0; to maxFeatures do
6:      index = rand %(max + 1)
7:      for j = 0; to n do
8:          for k = 0; to n do
9:              if TrainingSet[k][index] < TrainingSet[j][index] then
10:                 Subset1[leftCount] = TrainingSet[k]
11:                 leftCount + +
12:             else
13:                 Subset2[rightCount] = TrainingSet[k]
14:                 rightCount + +
15:             end if
16:         end for
17:         Gini = Gini_Index(Subset1, Subset2
18:         if Gini < BestGini then
19:             Gini = BestGini
20:             ind = index
21:             value = TrainingSet[j][index]
22:             Subset[0] = Subset1
23:             Subset[1] = Subset2
24:         end if
25:     end for
26: end for
```

Para obtener el menor tamaño posible en los árboles generados, en muchas ocasiones se realiza un proceso de poda, el cual consiste en eliminar ciertos nodos de los árboles al evaluar ciertas condiciones, sin que se vea afectada su precisión. En la implementación de este trabajo se consideran las condiciones de parada para evitar realizar este proceso de poda, las cuales son una profundidad máxima y/o un mínimo de instancias en los subconjuntos requerido para realizar el proceso de partición.

Así, la construcción de los árboles se puede resumir tal como se presenta en el Algoritmo 3-1. En dicho pseudocódigo se observan las funciones $BestDataSplit()$ (Alrogitmo 3-2)y $GrowTree()$ (Algoritmo 3-3), que realizan el procedimiento para la generación de la mejor partición y para el crecimiento del árbol, respectivamente.

Algorithm 3-3 GrowTree($Root, Subsets[0,1], minSamples, maxDepth, Depth$)

```
1: LeftNode = CreateNode()
2: RightNode = CreateNode()
3: Depth = 1
4: if Depth ≥ maxDepth then
5:     LeftNode = Class(Subsets[0])
6:     RightNode = Class(Subsets[1])
7:     LeftLeaf = LeftNode
8:     RightLeaf = RightNode
9:     return
10: end if
11: if length(Subset[0]) ≤ minSamples then
12:     LeftNode = Class(Subsets[0])
13:     LeftLeaf = LeftNode
14:     return
15: else
16:     newSubsets = BestDataSplit(Subsets[0], maxFeatures,)
17:     GrowTree(LeftNode, newSubsets[0], minSamples, maxDepth, Depth + 1)
18: end if
19: if length(Subset[1]) ≤ minSamples then
20:     RightNode = Class(Subsets[1])
21:     RightLeaf = RightNode
22:     return
23: else
24:     newSubsets = BestDataSplit(Subsets[1], maxFeatures,)
25:     GrowTree(LeftNode, newSubsets[1], minSamples, maxDepth, Depth + 1)
26: end if
```

3.2.2 Construcción y Funcionamiento del Random Forest

Una vez conocido el proceso de construcción de los árboles de decisión, que son los componentes individuales de un random forest, entonces ya se puede proceder a extender este proceso hasta llegar a la construcción de los bosques.

Algorithm 3-4 BuildRandomForest($TrainingData[0,\cdots,n][0,\cdots,m], NumberOfTrees, maxDepth, minSamples, maxFeatures, ratio$)

```
1: RF[NumberOfTrees] = NULL
2: for i = 0; to NumberOfTrees do
3:     Sample = getSample(TrainingData[0,···,n][0,···,m], ratio)
4:     Tree = BuildTree(Sample, maxFeatures, minSamples, maxDepth)
5:     RF[i] = Tree
6: end for
7: return RF
```

Una característica importante de los random forests, es que cada uno de los árboles es construido con un conjunto de datos ligeramente distinto; por ello, a partir del conjunto de datos de entrenamiento, se toma una muestra aleatoria para la construcción de cada árbol (de aquí proviene el nombre del método, *random forest*).

Algorithm 3-5 getSample($TrainindData[0,\cdots,n][0,\cdots,m], ratio$)

```
1: samplerows = n*ratio
2: count = 0
3: sample[samplerows] = 0
4: while count < samplerows do
5:     max = rows-1
6:     index = rand() % (max+1)
7:     sample[count] = TrainingData[index]
8:     count++
9: end while
```

Una vez tomadas las muestras para el proceso de entrenamiento, se procede a construir cada uno de los árboles de manera independiente; de esta forma, lo que se tiene es en realidad un arreglo de árboles de decisión construidos a partir de una misma fuente, de forma que cada árbol sirve como un clasificador que permite aumentar la precisión en el proceso de predicción.

En el Algoritmo 3-4 se muestra el proceso para la construcción del bosque a partir del conjunto de datos de entrenamiento. El parámetro $NumberOfTrees$ se refiere al número de árboles por el que estará compuesto el bosque; mientras que los parámetros $maxDepth$ y $minSamples$ indican las condiciones de parada en la construcción de los árboles, siendo profundidad máxima y el mínimo de instancias requerido para realizar una división del conjunto de datos en cada nodo. El parámetro $maxFeatures$ indica el número de veces que se pueden generar particiones en cada nodo para su evaluación; es decir, si $maxFeatures = 2$, entonces en cada nodo se seleccionarán dos variables aleatorias a partir de las cuales el conjunto de datos se dividirá en dos, y se procede a la evaluación de las divisiones con el índice gini. Por último, con $ratio$ se indica el porcentaje del conjunto de entrenamiento original que

Algorithm 3-6 RandomForest($TrainingSet[0,\cdots,m][0,\cdots,p]$, $TestingSet[0,\cdots,n][0,\cdots,p]$, $NumberOfTrees, maxFeatures, minSamples, maxDepth, ratio$)

```
1: RF[NumberOfTrees] = BuildRandomForest()
2: predictions[n] = GetPredictions(TestingSet[][], RF[NumberOfTrees])
3: actual[n] = 0
4: for i = 0 to n do
5:     actual[i] = TestingSet[i][p-1]
6: end for
7: correct = 0
8: for i = 0 to n do
9:     if actual[i] == predictions[i] then
10:        correct++
11:    end if
12: end for
13: accuracy = correct/n
```

servirá para la construcción de cada árbol; por ejemplo, si $ratio = 0{,}9$, entonces para la construcción de cada árbol se toma una muestra del $90\,\%$ del conjunto de datos de entrenamiento original de manera aleatoria por medio de la función $getSample$ (Algoritmo 3-5).

Por otro lado, el proceso general del funcionamiento del random forest viene descrito en el Algoritmo 3-6, donde primeramente se toman como entrada los conjuntos para el entrenamiento y para la prueba; con los datos de entrenamiento se construye el random forest con el número de árboles determinado. Posteriormente, a partir del conjunto de datos para la etapa de prueba y con el bosque construido, se realizan las predicciones con la función $GetPredictions()$; asimismo, se toman los valores reales de las clases para cada instancia, y se comparan con las predicciones para determinar la precisión del modelo.

Para realizar las predicciones a partir del random forest (Algoritmo 3-7), se toman cada una de las instancias y se realiza la predicción por cada árbol de manera individual con $Prediction()$, posteriormente, se realiza el conteo de votos para determinar a cuál clase pertenece la instancia. Esta implementación debe recordarse que es para problemas de clasificación binaria, donde las clases son identificadas con 0 y 1.

Las predicciones de cada árbol (Algoritmo 3-8) se realizan al comparar la variable de cada una de las instancias del conjunto de datos de prueba con la que se obtuvo la mejor partición de datos en el nodo raíz, con el valor obtenido durante el proceso de entrenamiento; por ejemplo, determinar si la variable $x_4 < 0{,}4$ en la instancia 2 del $TestingSet$, si es así, entonces se pasa al nodo izquierdo, de lo contrario, se pasa al nodo derecho y se repite el proceso, según el nodo correspondiente. Cuando se llega al nodo hoja, entonces se obtiene la clase a la que pertenece dicha instancia, según el modelo.

3.3 Análisis de Complejidad

La complejidad computacional de un árbol de decisión está relacionada directamente con el número de instancias y el número de características o variables a evaluar. En el caso del algoritmo implementado para la construcción de los árboles, se debe recordar que se establecieron ciertas condiciones para evitar

Algorithm 3-7 GetPredictions($TestingSet[0,\cdots,n][0,\cdots,p], RF[NumberOfTrees]$)

Input: Conjunto de datos de prueba
Output: Predicción de la clase perteneciente

```
1:  for i = 0 to n do
2:      zeros = 0
3:      ones = 0
4:      for j = 0 to NumberOfTrees do
5:          prediction = Prediction(RF[j], TestingSet[i])
6:          if prediction == 0 then
7:              zeros++
8:          else
9:              ones++
10:         end if
11:     end for
12:     if ones > zeros then
13:         return 1
14:     else
15:         return 0
16:     end if
17: end for
```

Algorithm 3-8 Prediction($Tree, Instance$)

Input: Conjunto de datos de prueba
Output: Predicción de la clase a la que pertenece cada instancia

```
1:  if Instance[Tree.ind] < Tree.value then
2:      if Tree.LeftNode ≠ Null then
3:          return Prediction(Tree.LeftNode, Instance)
4:      else
5:          return Tree.LeftLeaf
6:      end if
7:  else
8:      if Tree.RightNode ≠ Null then
9:          return Prediction(Tree.RightNode, Instance)
10:     else
11:         return Tree.RightLeaf
12:     end if
13: end if
```

que el árbol crezca de manera excesiva; así, si se establece n como el número de instancias del conjunto de datos, m como el número máximo de características que se pueden evaluar de manera aleatoria por cada nodo, y p como la profundidad máxima, y dado que se sabe que el número de hojas de un árbol binario está determinado por

$$\sum_{i=0}^{p-1} 2^i \tag{4}$$

entonces el número de evaluaciones máximas que se pueden realizar en la construcción del árbol podría representarse de la siguiente forma

$$G(p) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=0}^{p-2} 2^k \tag{5}$$

Donde se puede observar que el número de evaluaciones $G(p)$, es decir, la cantidad de veces que se debe calcular el índice gini para la construcción del árbol a una profundidad p, para todas las n instancias y seleccionando m características por nodo, está dado en función del total de nodos internos que lo conforman. Así, al resolverse la expresión anterior se tiene

$$G(p) = \sum_{i=1}^{n} \sum_{j=1}^{m} (2^{p-1} - 1) \tag{6}$$

$$G(p) = n \cdot m \cdot (2^{p-1} - 1) \tag{7}$$

Siendo esta última expresión la que indica el número de evaluaciones totales que se pueden realizar en el peor de los casos durante la construcción de un árbol de decisión. Si esto se extiende para todo el bosque con un total de t árboles, entonces esto se repite para cada uno de ellos, así

$$G(t) = \sum_{i=1}^{t} (n \cdot m \cdot (2^{p-1} - 1)) \tag{8}$$

$$G(t) = t \cdot n \cdot m \cdot (2^{p-1} - 1) \tag{9}$$

Observando la expresión anterior, se puede concluir que el tiempo de ejecución del algoritmo dependerá directamente de los valores para todos estos parámetros; por lo que la complejidad en la construcción de un bosque está dada por $O(t^* n^* m^* p)$.

Ahora, para la realización de la tarea de clasificación, y suponiendo que todos los t árboles del bosque tienen una profundidad de p, entonces para cada instancia se realiza una evaluación por cada nivel; es decir, en total se realizan $p - 1$ evaluaciones para determinar la clase a la que pertenece la instancia. Una expresión que permita representar lo anterior sería la siguiente

$$\sum_{i=1}^{n} \sum_{j=1}^{t} \sum_{k=1}^{p-1} 1 \tag{10}$$

Es decir, resolviendo la expresión anterior, en total se realiza un total de $n \cdot t \cdot (p - 1)$ evaluaciones para la tarea de clasificación; por lo que esta tarea en específico tiene una complejidad de $0(n^* t^* p)$.

Todo lo descrito hasta el momento fue implementado y se realizaron distintas pruebas para estudiar la influencia de los valores de los parámetros como $maxDepth$, $maxFeatures$, $NumberOfTrees$, y $ratio$, tanto en el tiempo de ejecución del algoritmo, como en la precisión de los modelos generados

para distintos conjuntos de datos.

4 Resultados

4.1 Descripción de los Casos de Prueba

Para el presente trabajo se realizaron pruebas para observar el comportamiento del modelo principalmente con dos aspectos. El primero, constó en realizar pruebas con una sola base de datos, pero con distintos valores de parámetros tales como la profundidad máxima de los árboles, el número de árboles dentro del bosque, entre otras. Por otro lado, también se estudió el tiempo de ejecución de la implementación del algoritmo para distintos conjuntos de datos con números diferentes de instancias y características o variables cada uno.

En las siguientes secciones se explican a detalle los distintos casos estudiados para cada uno de los aspectos mencionados, así como las características de los conjuntos de datos tomados para las pruebas; todas estas bases de datos fueron tomadas de [23].

4.1.1 Selección de Parámetros

Como ya se ha mencionado anteriormente, para la implementación de este algoritmo es necesario tomar como entrada valores para ciertos parámetros y que, de hecho, influyen en la complejidad computacional del algoritmo para la construcción y ejecución de los árboles de decisión que conforman al bosque.

Debido a lo anterior, es importante estudiar qué tanto influyen estos parámetros tanto en el tiempo de ejecución como en la precisión de los modelos. Por ejemplo, ¿cuántos árboles son necesarios para ofrecer buenos resultados en un tiempo aceptable? ¿Qué profundidad es suficiente para que un árbol tenga una buena precisión?

Para tratar de dar una posible respuesta a las preguntas anteriores, se realizaron pruebas utilizando distintas cantidades de árboles con distintas profundidades máximas; esto con la base de datos *diabetes* de la cual se tomaron distintos porcentajes de las instancias totales para el entrenamiento, y determinar cómo influyen todos estos aspectos en la precisión del modelo.

Esta base de datos está compuesta por información médica de un total de 768 mujeres mayores a los 21 años de edad y 9 características por cada una de ellas. Los primeros 8 atributos indican datos tales como número de embarazos previos, presión sanguínea diastólica, concentración de glucosa en plasma, edad, entre otros; y la última clase indica si estas personas presentan diabetes o no. Este conjunto de datos funciona precisamente para determinar si una persona presenta características de esta enfermedad.

Cuadro 4-1: Valores para las pruebas de los parámetros

Num. of Trees	**Max Depth**	**Ratio for Training**
1 5 10 50 100	5 7 10	0.4 0.8

Con esta base de datos se tomó tanto el tiempo de ejecución como la precisión con distintos valores para los parámetros utilizados, los cuales se encuentran resumidos en la Tabla 4-1. Cada uno de los valores de cada parámetro fue combinado con todos los valores de los parámetros restantes; por lo tanto, al final se realizaron 18 pruebas distintas para determinar cuál combinación resulta más conveniente. El parámetro *Num. of Trees* indica el número de árboles que contiene el bosque; cada uno de esos árboles puede tener una profundidad máxima de cualquiera de los 3 valores indicados por *Max Depth*. Y para la construcción de estos árboles, se puede tomar ya sea el 40 % o el 80 % del conjunto de datos original durante la etapa de entrenamiento. Todas estas pruebas se realizaron durante 31 veces y se tomaron los mejores valores para el tiempo de ejecución y la precisión; además, de la precisión también se tomó el menor valor registrado para cada experimento.

4.1.2 Pruebas con Conjuntos de Datos

Una vez establecidos los mejores valores para los parámetros, se procedió a realizar pruebas de clasificación con distintos conjuntos de datos; cada uno de los cuales representa situaciones distintas y, por lo tanto, cada conjunto cuenta con un número distinto de instancias y características.

Cuadro 4-2: Bases de Datos Probadas.

Dataset	**Num. Instances**	**Num. Features**
diabetes	768	9
sonar	208	61
ionosphere	351	35
banknote	1372	5

Para la realización de estas pruebas se utilizaron, además del conjunto *diabetes*, las bases de datos *sonar*, *ionosphere* y *banknote*, cuyas características generales se resumen en la Tabla 4-2. Estas pruebas fueron realizadas durante un total de 50 veces, tomándose el menor tiempo de ejecución, así como la

mejor precisión registrada.

En el caso de *sonar*, es una base de datos en la que se registraron distintas pruebas de incidencia sonora a rocas y minerales con distintos ángulos; a partir de estos datos, se puede determinar si una muestra representa a una roca o a un mineral.

Por otro lado, *ionosphere* es un conjunto de datos compuesto por 351 instancias de 34 características cada una; y sirve para detectar estructuras en la atmósfera a partir de la reflexión de ondas radiales lanzadas a los electrones libres en la ionósfera. En el caso de que para una instancia se tenga evidencia de la existencia de una estructura, entonces se toma dentro de la clase 1, en caso contrario, se toma dentro de la clase 0.

El conjunto *banknote* está compuesto por un total de 1372 instancias con 4 características tomadas a partir de imágenes digitalizadas de billetes, que permiten determinar si un billete es falso o auténtico.

En la siguiente sección se presentan los resultados obtenidos y un análisis de cada una de las pruebas descritas anteriormente.

4.2 Discusión de los Resultados

4.2.1 Pruebas Para la Selección de Parámetros

Una vez realizados los experimentos descritos en la subsección 4.1.1 y se registraron los tiempos de ejecución así como la precisión en las predicciones realizadas para el conjunto de prueba; un resumen de los resultados obtenidos se puede observar en los Cuadros 4-3 y .

Cuadro 4-3: Resultados de Pruebas para el 40 % de Instancias del Conjunto de Datos Original.

No. Árboles	Prof. Máxima	Precisión				Tiempo
		Min	Max	Promedio	Max - Min	
1	5	0.617	0.726	0.697	0.109	0.111
	7	0.656	0.73	0.689	0.074	0.115
	10	0.636	0.71	0.667	0.074	0.14
5	5	0.702	0.765	0.735	0.063	1.162
	7	0.703	0.763	0.735	0.06	1.283
	10	0.691	0.765	0.712	0.074	1.324
10	5	0.726	0.765	0.742	0.039	1.181
	7	0.719	0.767	0.746	0.048	1.211
	10	0.695	0.767	0.725	0.072	1.282
50	5	0.736	0.769	0.75	0.033	6.082
	7	0.739	0.78	0.755	0.041	6.234
	10	0.713	0.765	0.74	0.052	6.317
100	5	0.732	0.769	0.752	0.037	12.093
	7	0.736	0.758	0.747	0.022	13.34
	10	-	-	-	-	-

Cuadro 4-4: Resultados de Pruebas para el 80 % de Instancias del Conjunto de Datos Original.

No. Árboles	Prof. Máxima	Precisión				Tiempo
		Min	Max	Promedio	Max - Min	
1	5	0.653	0.764	0.719	0.111	0.436
	7	0.66	0.777	0.699	0.117	0.478
	10	0.633	0.732	0.681	0.099	0.474
5	5	0.64	0.758	0.73	0.118	2.367
	7	0.725	0.797	0.768	0.072	2.497
	10	0.647	0.751	0.711	0.104	2.624
10	5	0.699	0.804	0.748	0.105	4.909
	7	0.705	0.804	0.741	0.099	5.621
	10	0.712	0.784	0.756	0.072	6.221

Para el caso de la precisión, se tomaron los mínimos y máximos valores de precisión para cada combinación de parámetros, con el objetivo de observar si existe una mayor o menor variabilidad para estas combinaciones. Asimismo, se registró la precisión promedio para cada experimento, para analizar en qué casos se presentan los máximos valores. En el caso de los experimentos donde se tomó el 80 % del conjunto original para la construcción de los modelos, ya no se pudieron obtener resultados para los casos de 50 y 100 árboles; por lo que los resultados para este caso se encuentran reportados solamente para los 3 primeros parámetros.

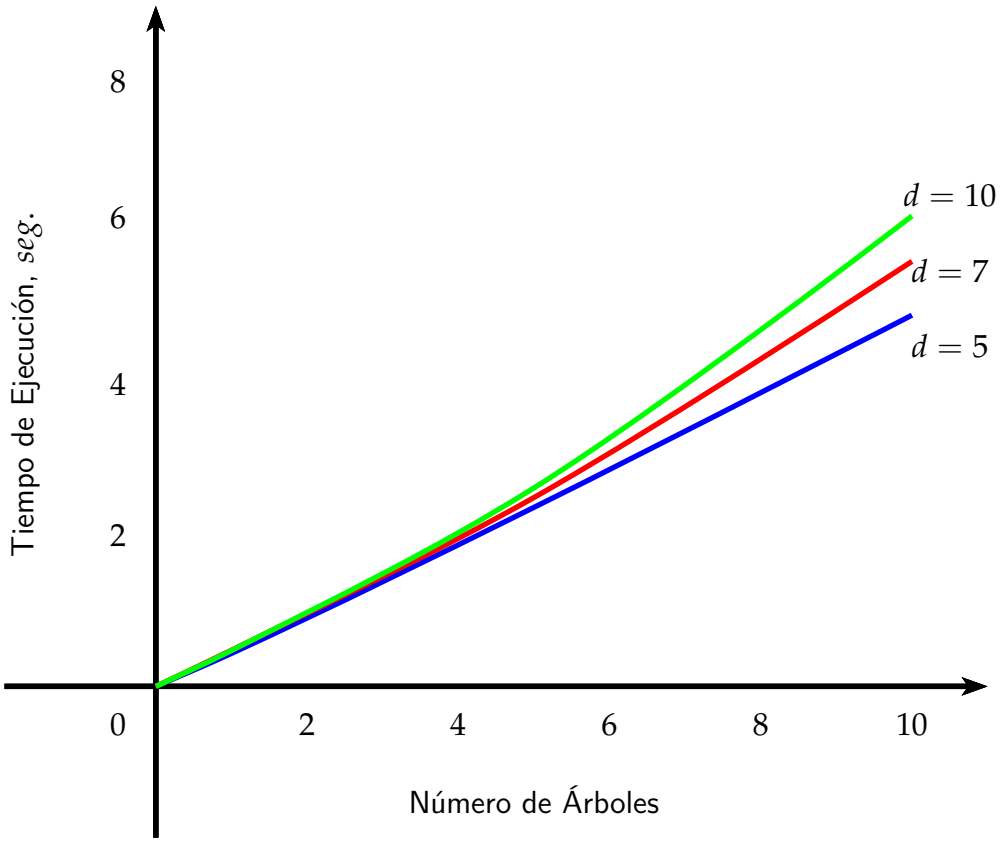

Figura 4-4: Tiempo de Ejecución con Diferentes Números de Árboles con el 40 % de las Instancias

De acuerdo con los resultados, en general existe una menor variabilidad (diferencia entre la precisión

máxima y mínima observadas para cada experimento), en el caso de cuando se tomó el 40 % de las instancias del conjunto original, en comparación con la variabilidad observada de cuando se tomó el 80 % del conjunto y, por lo tanto, los valores promedio son más bajos en este último caso. Sin embargo, esto se debe principalmente a que los valores mínimos observados son relativamente similares en ambos casos, pero los valores máximos de precisión observados son mayores en el segundo caso. Además, se puede observar que en el primer caso solamente el experimento de 50 árboles con una profundidad máxima de 7 niveles supera el 77 % de precisión; mientras que, en el segundo caso, varios experimentos con una considerable menor cantidad de árboles alcanzaron precisiones de alrededor del 80 %.

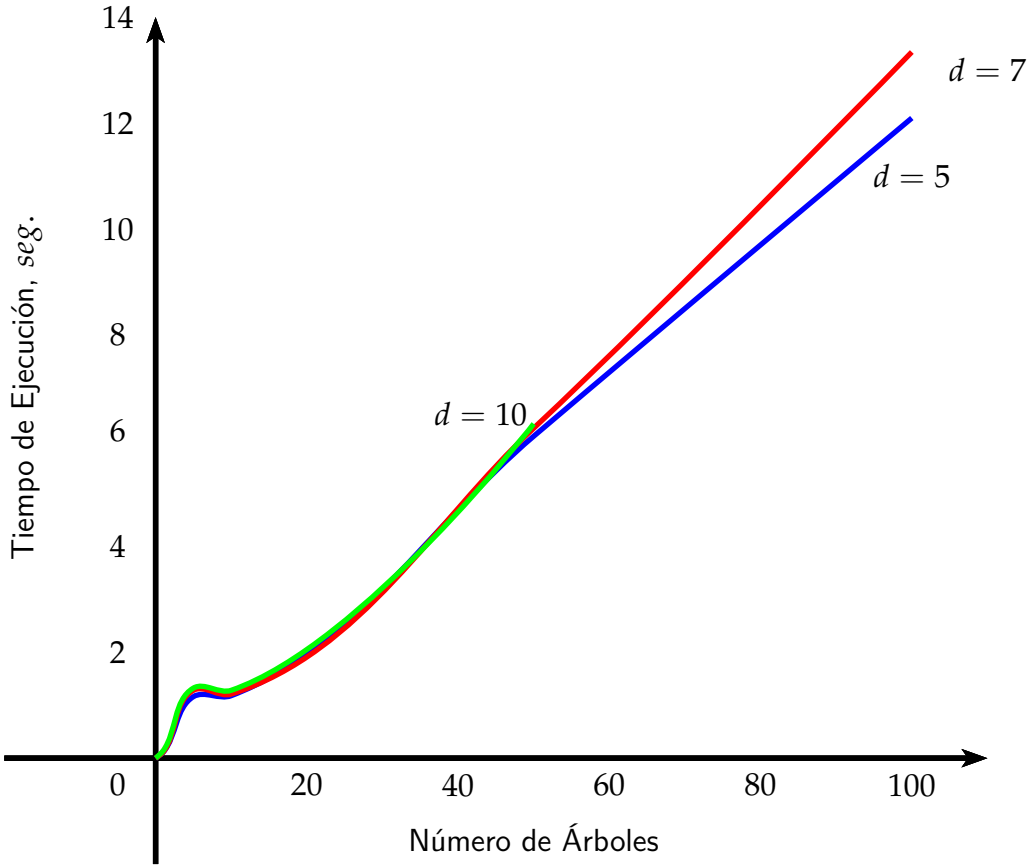

Figura 4-5: Tiempo de Ejecución con Diferentes Números de Árboles con el 80 % de las Instancias

Por otro lado, para el caso de los tiempos de ejecución, los tiempos reportados corresponden al mejor tiempo registrado según cada caso; y a partir de ellos se obtuvo la tasa de crecimiento de los tiempos de ejecución con respecto al número de árboles que conforman al bosque. Por ejemplo, se dividió el tiempo de ejecución para el caso de un bosque con $t = 10$ árboles de un profundidad máxima de $d = 5$ niveles entre el tiempo para un bosque con 5 árboles de la misma profundidad máxima; se realizó lo mismo para todas las d profundidades máximas con un valor de t y se obtuvo el promedio, para observar finalmente el ritmo de crecimiento de acuerdo con el número de árboles (Cuadro 4-5).

En las Figuras 4-4 y 4-5 se muestran los tiempos de ejecución con respecto al número de árboles que conforman al bosque, para los casos donde se toman el 80 % y el 40 % de las instancias del conjunto original, respectivamente; donde se puede observar fácilmente que el crecimiento tiende a ser aproximadamente lineal con respecto al número de árboles para ambos casos. Para las tasas de crecimiento, se puede apreciar que para los bosques compuestos de 10 árboles se tiene un menor orden de crecimiento en los tiempos de ejecución con respecto a los bosques de 5 árboles; además de que es

Cuadro 4-5: Tasas de Crecimiento del Tiempo de Ejecución

		5	7	10	Promedio
40 %	T_5/T_1	10.468	11.156	9.457	10.36
	T_{10}/T_5	1.016	0.943	0.968	0.976
	T_{50}/T_{10}	5.149	5.147	4.927	5.075
	T_{100}/T_{50}	1.988	2.139	-	2.064
80 %	T_5/T_1	5.428	5.223	5.535	5.396
	T_{10}/T_5	2.073	2.251	2.37	2.231

precisamente en esos casos cuando se tiene la menor diferencia entre los valores máximo y mínimo de las tasas de crecimiento registrados para distintas profundidades máximas.

En resumen, de acuerdo con todo lo anterior, se puede observar que los mejores valores de precisión y tiempos de ejecución se presentan, para casi todos los casos, cuando se tienen árboles con profundidades entre 5 y 7 niveles máximo. En cuanto al número de árboles, se puede apreciar que cuando se tienen pocas instancias para realizar el entrenamiento, los mejores resultados se obtienen cuando se tienen bosques conformados por 50 árboles; mientras que, cuando se tiene un mayor número de instancias para la realización del entrenamiento, resultados similares y, en algunos casos mejores, se pueden alcanzar con bosques conformados por solamente 10 árboles.

Para la realización de las siguientes pruebas se determinó que, aunque los mejores valores de los parámetros dependen de las características específicas de cada conjunto de datos, utilizar los mismos valores para todos los casos. Así, a pesar de que en muchos casos se obtuvo una mejor precisión con bosques de 10 árboles con profundidades máximas de 5 niveles, se decidió realizar las pruebas con 10 árboles con profundidades máximas de 7 niveles; debido a que esto depende del número de características de cada conjunto de datos en particular, y pueden existir casos en los que 5 niveles no sean suficientes.

4.2.2 Pruebas con Distintos Conjuntos de Datos

Para las pruebas con distintos conjuntos de datos, se registraron los tiempos de ejecución mínimos observados, así como se determinó la precisión promedio para cada uno de los conjuntos de datos. Un resumen se muestra en el Cuadro 4-6.

Cuadro 4-6: Tiempo de Ejecución y Precisión Promedio para Distintos Conjuntos de Datos

Conjunto de Datos	Tiempo, seg.	Precisión Promedio
diabetes	4.725	0.75
sonar	0.782	0.78
ionosphere	2.648	0.93
banknote	14.772	0.989

De acuerdo con los resultados, no es posible observar una correlación directa entre el número de instancias y/o características, con la precisión alcanzada en las predicciones. Por ejemplo, el conjunto *banknote* cuenta con el mayor número de instancias y el menor número de características a evaluar,

alcanzando una precisión promedio de 0.98; mientras que, el segundo conjunto con mayor número de instancias y menor número de características, es el de *diabetes*, el cual alcanza solamente una precisión de 0.77; un valor mucho menor en comparación con el primer conjunto mencionado.

Por otro lado, los conjuntos *ionosphere* y *sonar*, son los que cuentan con el menor número de instancias y mayor número de características a evaluar; en los cuales se alcanza un 0.93 y 0.78 de precisión promedio, respectivamente. Este comportamiento se debe a que la precisión, aunque sí depende de la cantidad de instancias y de variables que se tienen en el conjunto de datos, también depende mucho de las características propias de estas variables; es decir, existen casos en los que es difícil encontrar buenos niveles en la pureza de los datos al evaluar el índice gini, por lo que al final se terminan obteniendo valores relativamente altos en la cantidad de predicciones fallidas.

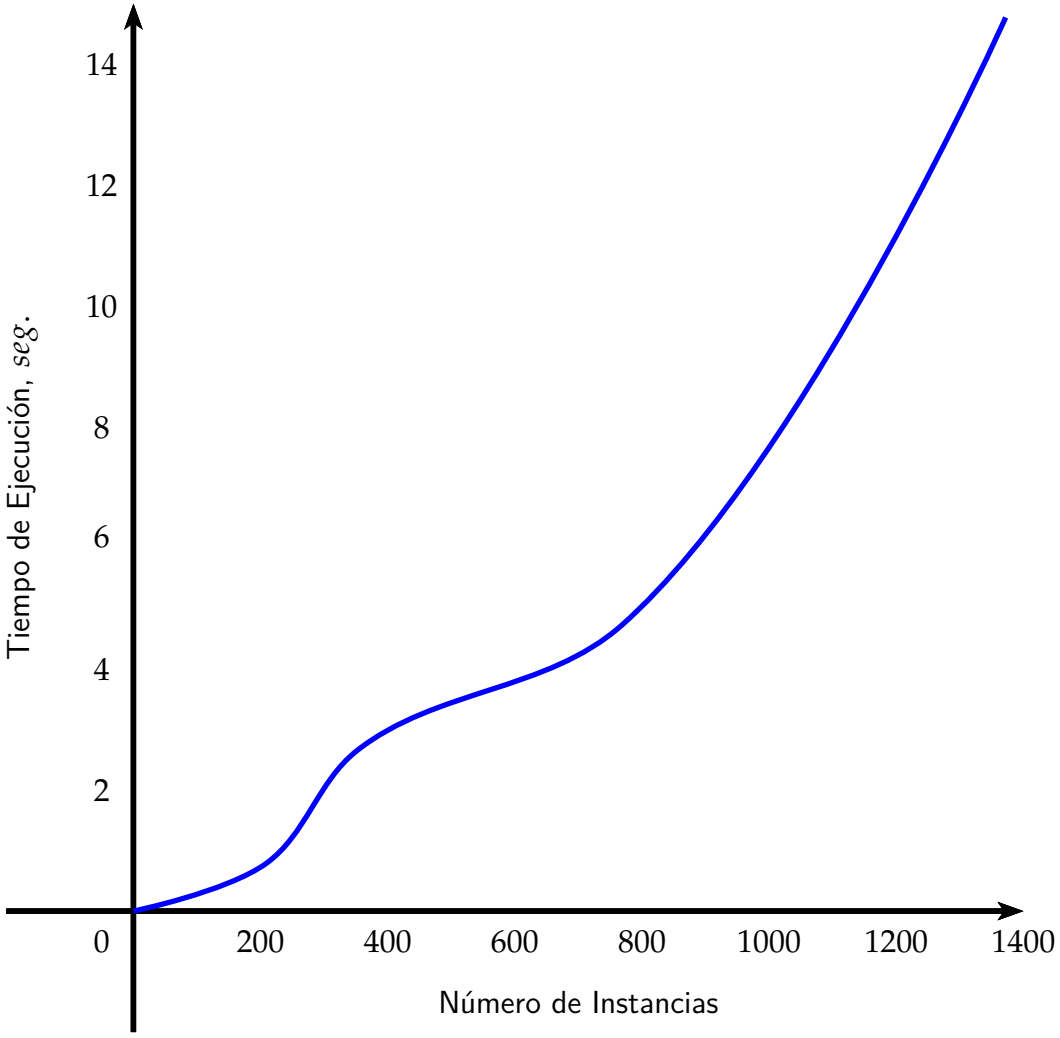

Figura 4-6: Tiempo de Ejecución para Distintos Conjuntos de Datos

En el caso del tiempo de ejecución, mostrado en la Figura 4-6, se puede apreciar que el tiempo de ejecución tiene un crecimiento aproximadamente cuadrático si se compara con el número de instancias que componen al conjunto de datos implementado; además, se puede notar que sigue un comportamiento muy similar al observado en la Figura 4-5 donde se compara con respecto al número de árboles.

5
Conclusiones

En este trabajo se mostró el funcionamiento del método de random forest para distintos números de árboles, profundidades máximas y número de instancias durante la construcción y prueba del bosque.

En general se encontró que la precisión se ve influenciada por los valores de estos parámetros, de forma que a un mayor número de instancias para la etapa de entrenamiento y un mayor número de árboles que compongan al bosque, se puede ver un incremento en la precisión del modelo. En cuanto a la influencia de la profundidad máxima, se pudieron observar casos en los que se alcanza una mayor precisión cuando se tienen profundidades entre 5 y 7 niveles máximos.

Una cosa importante es que se puede observar un límite en la precisión del modelo; es decir, existen casos en los que aunque se siga aumentando el número de árboles que compongan al bosque, la precisión ya no sigue aumentando. Por otro lado, también se puede apreciar que mientras más instancias se consideren para el entrenamiento, es posible que se pueda requerir una menor cantidad de árboles para alcanzar buenos valores de precisión, y viceversa, a menor cantidad de instancias, se necesita un mayor número de árboles.

En otros aspectos, es posible seguir realizando pruebas y experimentos con este trabajo, así como extenderlo a otras situaciones. Por ejemplo, se podría realizar una implementación que permita obtener los mejores valores de profundidad máxima y número de árboles dependiendo de las características del conjunto de datos utilizado para el problema que se pretende resolver. Además, esta implementación funciona solamente para problemas de clasificación binaria, por lo que es posible extenderlo a problemas multiclase o de regresión.

Bibliografía

[1] T. M. Mitchell, *Machine Learning*. McGraw-Hill Science, 1997.

[2] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2005.

[3] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer Science, 2006.

[5] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, 2005.

[6] M. S. Pang-Ning Tan and V. Kumar, *Introduction to Data Mining*. Pearson Education, 2006.

[7] A. Smola and S. Vishwanathan, *Introduction to Machine Learning*. Cambridge University Press, 2008.

[8] S. J. Simske, D. Li, and J. S. Aronoff, "A statistical method for binary classification of images," in *Proceedings of the 2005 ACM Symposium on Document Engineering*, DocEng '05, (New York, NY, USA), p. 127–129, Association for Computing Machinery, 2005.

[9] X. Zhao, L. Niu, and Y. Shi, "Kernel based simple regularized multiple criteria linear programs for binary classification," in *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 03*, WI-IAT '13, (USA), p. 58–61, IEEE Computer Society, 2013.

[10] E. Z-Flores, L. Trujillo, O. Schütze, and P. Legrand, "A local search approach to genetic programming for binary classification," GECCO '15, (New York, NY, USA), p. 1151–1158, Association for Computing Machinery, 2015.

[11] S. W. Purnami, S. Andari, and A. W. Rusydiana, "On selecting features for binary classification in microarray data analyses," ICMLC 2017, (New York, NY, USA), p. 133–136, Association for Computing Machinery, 2017.

[12] V. Estivill-Castro, M. Lombardi, and A. Marani, "Improving binary classification of web pages using an ensemble of feature selection algorithms," in *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '18, (New York, NY, USA), Association for Computing Machinery, 2018.

[13] L. Munkhdalai, T. Munkhdalai, O.-E. Namsrai, J. Y. Lee, and K. H. Ryu, "An empirical comparison of machine-learning methods on bank client credit assessments," *Sustainability*, vol. 11, no. 3, 2019.

[14] N. Oswal, "Predicting rainfall using machine learning techniques," 2019.

[15] C. Arizmendi, A. Vellido, and E. Romero, "Binary classification of brain tumours using a discrete wavelet transform and energy criteria," pp. 1 – 4, 03 2011.

[16] S. Peng, X. Zeng, X. Li, X. Peng, and L. Chen, "Multi-class cancer classification through gene expression profiles: microrna versus mrna," *Journal of Genetics and Genomics*, vol. 36, no. 7, pp. 409–416, 2009.

[17] G. Mujtaba, L. Shuib, R. G. Raj, R. Rajandram, K. Shaikh, and M. A. Al-Garadi, "Automatic icd-10 multi-class classification of cause of death from plaintext autopsy reports through expert-driven feature selection," *PLOS ONE*, vol. 12, pp. 1–27, 02 2017.

[18] M. A. Marins, B. D. Barros, I. H. Santos, D. C. Barrionuevo, R. E. Vargas, T. de M. Prego, A. A. de Lima, M. L. de Campos, E. A. da Silva, and S. L. Netto, "Fault detection and classification in oil wells and production/service lines using random forest," *Journal of Petroleum Science and Engineering*, vol. 197, p. 107879, 2021.

[19] W.-Y. Loh, "Classification and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, pp. 14 – 23, 01 2011.

[20] R. A. O. Leo Breiman, Jerome H. Friedman and C. J. Stone, *Classification and Rgression Trees*. Chapman & Hall, 1998.

[21] A. Criminisi, J. Shotton, and E. Konukoglu, *Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning*, vol. 7, pp. 81–227. NOW Publishers, foundations and trends® in computer graphics and vision: vol. 7: no 2-3, pp 81-227 ed., January 2012.

[22] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[23] D. Dua and C. Graff, "UCI machine learning repository," 2017.