*Simple Random Forest Algorithm*

Tabla de contenido

[1. Introducción 3](#_Toc12565182)

[2. Preliminares 4](#_Toc12565183)

[3. Metodología 6](#_Toc12565184)

[4. Resultados 12](#_Toc12565185)

[5. Conclusión 12](#_Toc12565186)

[Referencias 12](#_Toc12565187)

[Figuras 12](#_Toc12565188)

|  |  |
| --- | --- |
| Raimundo Flores Cabra  *dpto. Ciencias de la Computación e Inteligencia Artificial* *Universidad de Sevilla* Sevilla, España Correo electrónico UVUS: [raiflocab@alum.us.es](mailto:raiflocab@alum.us.es) Correo electrónico de contacto: [raimundokarate98@gmail.com](mailto:raimundokarate98@gmail.com) | Vicente López Vázquez *dpto. Ciencias de la Computación e Inteligencia Artificial* *Universidad de Sevilla* Sevilla, España Correo electrónico UVUS: [viclopvaz1@alum.us.es](mailto:viclopvaz1@alum.us.es) Correo electrónico de contacto: [vlopezvazquez3@gmail.com](mailto:vlopezvazquez3@gmail.com) |

*El objetivo de nuestro trabajo ha sido construir una versión simplificada del algoritmo de Bosques Aleatorios. Pero ¿en qué consiste este algoritmo? Un bosque aleatorio es un conjunto de árboles de decisión entrenados a partir de un conjunto de datos de entrenamiento con un conjunto aleatorio de atributos seleccionados para cada uno y tras la aplicación de Bootstrapping como técnica de muestreo. Esto nos permite construir un conjunto de árboles de decisión partiendo de un solo conjunto de datos de entrenamiento, ya que dichos datos son difíciles de conseguir y no muy abundantes. Por esta razón, se aplican técnicas de muestreo, en este caso Bootstrapping con reemplazo, para dar lugar a un nuevo conjunto de datos para cada árbol de decisión.*

*Esta técnica facilita el desarrollo de sistemas de predicción fiables y con mayor capacidad de predicción que un solo árbol de decisión. Debemos destacar que, a lo largo de este proyecto hemos comprendido que los datos con los que trabajamos no siempre son suficientes y tenemos que adaptarnos a ellos; que las técnicas de muestreo nos facilitan el trabajo con conjuntos de datos pequeños y con la construcción de varios árboles de decisión; y que un solo árbol no es tan eficaz como un conjunto promediado de ellos.*

# Introducción

La Inteligencia Artificial [1] tiene numerosos usos actualmente, desde sistemas de planificación automática o sistemas de decisión, a reconocimiento de la escritura o del habla. Nuestro caso trata sobre los árboles de decisión [2], un ejemplo de Aprendizaje Supervisado [3].

Un árbol de decisión es un modelo de predicción construido a partir de un conjunto de datos, de donde se extraen una serie de reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de manera sucesiva, para la resolución del problema. En la siguiente figura podemos ver la estructura de un árbol de decisión:

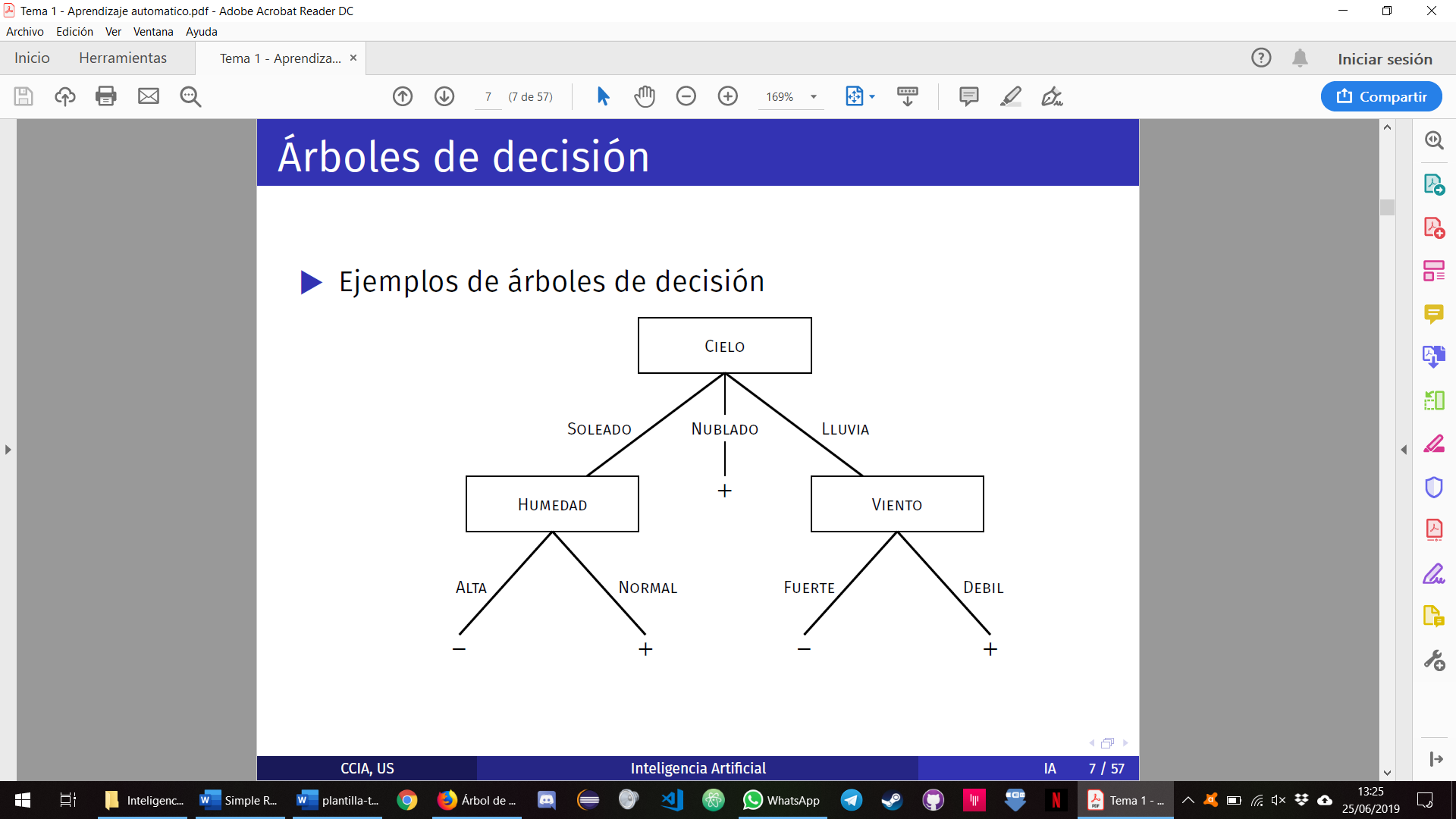
Podemos ver que el árbol clasificará los distintos ejemplos teniendo en cuenta como primera forma de clasificación el atributo “Cielo”. Dependiendo del valor de dicho atributo, el siguiente atributo usado para clasificar el ejemplo será “Humedad” o “Viento” o en caso de que el valor de “Cielo” sea nublado la clasificación es directa.

Figura 1 Árbol de decisión con 3 nodos de decisión.

A grandes rasgos, nuestro proyecto trabaja con dichos árboles de decisión. Pero se complica al introducir técnicas de muestreo como Bootstrapping [4] y Random Subspace Method [5]. Además, trabajaremos con varios árboles con el objetivo de aumentar la precisión calculando la salida promedia. La aplicación de técnicas de muestreo para obtener nuevos conjuntos de datos a partir de uno y el uso de varios árboles entrenados con diferentes conjuntos de datos da lugar a una técnica llamada Random Forest (Bosque Aleatorio) [6].

El uso de Random Forest aumenta la precisión que el uso de un solo árbol de decisión proporciona y facilita el trabajo con conjuntos de datos pequeños o difíciles de conseguir. Por eso su uso está muy extendido y aceptado.

Como conclusión, nuestro proyecto ha consistido en diseñar y construir una forma simple del algoritmo del Random Forest, compararlo con el uso de árboles individuales, profundizar en el uso de las técnicas de muestreo arriba citadas y experimentar con distintos parámetros para llegar a los mejores valores de predicción posibles.

# Preliminares

Antes de entrar de lleno en el trabajo, pasaremos a numerar y explicar en que consisten las técnicas que hemos usado durante el desarrollo del proyecto. Primero hablaremos de la técnica básica en la que se basa nuestro trabajo, para después entrar en las técnicas de muestreo utilizadas y finalmente en la técnica que se crea como conjunto de las anteriores:

* **Técnicas básicas:**
  + **Aprendizaje supervisado [3]:** El aprendizaje supervisado es una técnica usada para clasificar datos o ejemplos a partir de un conjunto de datos de entrenamiento. Los conjuntos de entrenamiento son vectores de datos donde cada dato está formado por unos atributos y por una clasificación o valor numérico.

El objetivo del aprendizaje supervisado es el de dar lugar a una función de decisión que pueda responder correctamente con un margen de error a unos datos de entrada según el entrenamiento dado. Los árboles de decisión son un ejemplo de aprendizaje supervisado.

* + **Árboles de decisión [2]:** Un árbol de decisión es una técnica de aprendizaje automático, y como hemos podido ver arriba, forma parte del aprendizaje supervisado ya que recibe un conjunto de datos de entrenamiento con el fin de entrenar al árbol para que pueda responder con una probabilidad alta de acierto frente a nuevos datos de entrada. Esta técnica es en la que se basa nuestro trabajo, pero la complicamos al aplicar distintas técnicas de muestreo al conjunto de datos de entrenamiento.
* **Técnicas de muestreo:**
  + **Bootstrapping [4]:** Es una técnica de muestreo usada cuando tenemos un conjunto de datos limitado y no podemos obtener más. Dado un conjunto de datos de N filas, crearemos un nuevo conjunto de datos del mismo tamaño a partir de este.

Dicho nuevo conjunto se crea recorriendo el conjunto de datos base; haciendo una selección con probabilidad de determinados ejemplos durante el recorrido; dichos ejemplos seleccionados sustituirlos por una copia de otro ejemplo cualquiera seleccionado con probabilidad. Así hasta generar un nuevo conjunto de datos de tamaño N o de un tamaño dado por el usuario.

Es una técnica muy útil si, como hemos dicho antes, no contamos con un conjunto de datos abundante y tenemos necesidad de más datos que los que podemos conseguir.

* + **Random Subspace Method [5]:** Otra técnica de muestreo usada en el ensamble de modelos. Se basa en la selección aleatoria de determinadas características del conjunto de datos de entrenamiento. El objetivo de esta técnica es que cada modelo de predicción se entrene con un conjunto de características concretas en vez de con el conjunto entero de estas, es decir, que cada modelo entrenado se especialice en un conjunto más pequeño de características para aumentar su precisión finalmente ensamblando todos los modelos y promediando la respuesta.
* **Técnica de ensamble:**
  + **Random Forest (Simplified) [6]:** Es la técnica de ensamble que hemos usado. Hemos desarrollado una versión simplificada de Random Forest o Bosque Aleatorio traducido.

Consiste en: partiendo de un conjunto de datos de entrenamiento y un número de árboles a entrenar, para cada árbol se aplican las técnicas de muestreo arriba descritas al conjunto de datos y se entrena el árbol. Tras entrenarlos todo, esta técnica devuelve un conjunto de árboles (Bosque) entrenados a partir de un mismo conjunto de datos, pero cada uno especializado en un conjunto de características (Aleatorio). Para las predicciones se promediarían las salidas del conjunto de árboles. Esta técnica proporciona una precisión mayor que un único árbol de decisión.

# Metodología

En esta sección de la documentación se mostrarán y realizarán una explicación de todas las funciones realizadas en el trabajo. Primero enseñaremos la entrada y salida de la función, para continuar después con una explicación de la misma y para finalizar la utilidad en nuestro algoritmo.

* ***load\_and\_process\_data***

**Entrada:**

* + El nombre del fichero

**Salida:**

* + DataFrame

**Algoritmo:**

* 1. Asignamos a una variable los datos del archivo que se le pasa como entrada
  2. Se crea un DataFrame que almacenará los datos del archivo una vez se codifiquen
  3. Mientras se tengan datos de la primera variable sin codificar, se le realizar un *LabelEncouder* y un *Fit* para la codificación de ese dato y se introduce en el DataFrame creado anteriormente.
  4. Una vez acabado este bucle, se retorna el DataFrame que tendrá los valores codificados.

**Utilidad:**

* + Este algoritmo es usado para leer los archivos de un fichero y poder codificar sus datos, ya que sino no sería posible usarlos para la realización de los árboles de decisión puestos que estos trabajan con números.
* ***bootstrapping***

**Entrada:**

* + DataFrame

**Salida:**

* + DataFrame

**Algoritmo:**

* 1. Se crea una variable que, mediante el uso de np.random.randint, creará un número aleatorio entre 0 y el tamaño del conjunto de entrenamiento
  2. En el DataFrame de salida se introducirán los valores del conjunto de datos de entrenamiento que la variable anterior, usada como índice, indique

**Utilidad:**

* + Este algoritmo es usado para la modificación del conjunto de entrenamiento para poder tener más valores, aparte de los que nos da el fichero. Esta es la forma en la que implementamos el uso Bootstrapping.
* ***divide\_data***

**Entrada:**

* + DataFrame

**Salida:**

* + Dos DataFrame

**Algoritmo:**

* 1. Se crean un primer DataFrame que contendrá los valores desde 0 hasta el tamaño del DataFrame de entrada menos uno
  2. Se crea el segundo DataFrame que contendrá la última columna del DataFrame de entrada

**Utilidad:**

* + Este algoritmo es usado para realizar la división del conjunto de entrenamiento y poder obtener todas las columnas con sus datos sin la variable de respuesta y, por otro lado, tener un DataFrame con solo la variable respuesta.
* ***random\_subspace***

**Entrada:**

* + DataFrame y un número real

**Salida:**

* + Un DataFrame con las columnas elegidas

**Algoritmo:**

* 1. Se almacena el número de columnas y sus índices en dos variables
  2. Se calcula el número de columnas elegidas, multiplicando el número real de entrada por el número de columnas
  3. Si el valor del real de entrada no está entre 0 y 1, se imprime por pantalla que debe estar entre ese rango
  4. Si se encuentra en ese rango, se vuelve a almacenar el índice de las columnas, pero solo las que se calculen con la función *random.shape* a la cual se le pasa como parámetro los índices de las columnas y las columnas seleccionadas

**Utilidad:**

* + Este algoritmo es usado para seleccionar aquellas columnas del conjunto de entrenamiento de las cuales se van a especializar.
* ***select\_columns***

**Entrada:**

* + Dos DataFrame

**Salida:**

* + DataFrame

**Algoritmo:**

* 1. Crea un diccionario que almacenará las columnas elegidas
  2. Mientras no se acaben de recorrer todos los índices de las columnas que se le pasan como entrada
  3. Se guarda el valor de la columna que indique el índice del bucle y este valor se introduce en el diccionario del paso 1 en la posicion que diga el índice el bucle.
  4. Por último, transforma el diccionario en un DataFrame y lo retonar

**Utilidad:**

* + Este algoritmo es usado para obtener las columnas con las que fueron entrenadas los diferentes árboles.
* ***train\_forest***

**Entrada:**

* + Nombre del Archivo y tres enteros que representan, el número de árboles a entrenar, el número de características a tener en cuenta y la profundidad

**Salida:**

* + DataFrame

**Algoritmo:**

* 1. Se utiliza la función ***load\_and\_process\_data*** y se introduce en una variable
  2. Se crean variables para representar los bosques, los índices elegidos y un árbol de decisión
  3. Se realiza un bucle for entre 0 y el número de árboles
  4. Sobre esta variable que tiene los datos del fichero se realiza la función ***divide\_data*** y se introduce en dos variables
  5. Se usan las funciones ***random\_subspace*** y ***select\_colums*** y se introducen en dos variables, índices de las columnas y columnas seleccionadas respectivamente
  6. Se realiza la función *fit* de *decicionTree* y se les pasa como parámetros las columnas seleccionadas y la columna de respuestas del paso 4. Todo esto se inicializa en una variable
  7. Se introduce en los índices elegidos, las columnas seleccionadas y a su vez en la variable bosque se introduce la variable del paso 6
  8. Una vez finaliza el bucle se retorna el bosque y los índices seleccionados

**Utilidad:**

* + Este algoritmo es usado para entrenar a todos los árboles que se le pasen como parámetros.
* ***forest\_predictions***

**Entrada:**

* + Nombre del Archivo y dos DataFrame, que representan el bosque y los índices escogidos

**Salida:**

* + Dos DataFrame, las predicciones del bosque y la moda de las predicciones

**Algoritmo:**

* 1. Se utiliza la función ***load\_and\_process\_data*** y se introduce en una variable
  2. Sobre esta variable que tiene los datos del fichero se realiza la función ***divide\_data*** y se introduce en dos variables
  3. Se crea un diccionario que contendra las predicciones y se realiza un bucle desde 0 hasta el tamaño del bosque
  4. Se guarda el nombre del árbol y se llama a la ***select\_colums*** pasándole como parámetros el conjunto de entrenamiento sin las respuestas
  5. Se realiza la predicción con la función *predict*, pasándole como parámetro las listas seleccionadas del paso 4, sobre el árbol del bosque que indique el índice del bucle y se introduce en el diccionario donde indique el nombre de la columna
  6. Al finalizar el bucle, el diccionario se convierte en un DataFrame y se realiza la moda de las variables de este DataFrame y por último se retornan estas dos variables

**Utilidad:**

* + Este algoritmo es usado realizar la predicción de los conjuntos de prueba y poder usarlos en las tasas.
* ***calcular\_accuracy***

**Entrada:**

* + Dos DataFrame, respuestas del conjunto de prueba y las predicciones

**Salida:**

* + Tasa de aciertos y Tasa de aciertos balanceada

**Algoritmo:**

* 1. Usando las funciones *accuracy\_score* y *balanced\_accuracy\_score,* pasándoles como parámetros a ambos los DataFrame de entrada, calculan las tasas de acierto y acierto balanceado y retonan

**Utilidad:**

* + Este algoritmo es usado para calcular las tasas de acierto y acierto balanceado.

# Resultados

# Conclusión

Para finalizar nos gustaría comentar que esta metodología usada en esta asignatura de tener un proyecto que entregar al final, una vez ya finalizado todos los exámenes, no ha parecido bastante interesante pues de esta forma realmente vemos el potencial y la funcionalidad de, en nuestro caso, los árboles de decisión y ensamble. Ya que solo viendo teoría y haciendo problemas no sé ve cómo algo con valor fuera de la asignatura y también nos ha abierto una posibilidad de aprender un nuevo lenguaje de programación que nos puede servir en alguna situación.

Como aporte final y como posible mejora, pondríamos alguna serie de ejemplos en los que poder basarnos y no tener que ir buscando a ciegas o depender tanto de las tutorías con los profesores, ya que, aunque siempre nos responden las dudas y están ahí, no todos podemos trasladarnos a la universidad una vez finalizado el curso.

# Referencias

1. <https://es.wikipedia.org/wiki/Inteligencia_artificial>
2. <https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C3%B3n>
3. <https://es.wikipedia.org/wiki/Aprendizaje_supervisado>
4. <https://es.wikipedia.org/wiki/Bootstrapping_(estad%C3%ADstica)>
5. <https://en.wikipedia.org/wiki/Random_subspace_method>

1. <https://es.wikipedia.org/wiki/Random_forest>

# Figuras

1. *Tema 1 – Aprendizaje automático. dpto. Ciencias de la Computación e Inteligencia Artificial Universidad de Sevilla. Sevilla, España.*