Ensamble de modelos predictivos

Elena Tomás Vela
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
eletomvel@alum.us.es
etomasvela@gmail.com

Tadeo Cabrera Gómez
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
tadcabgom@alum.us.es
tcabgom222@gmail.com

***Resumen*—En este trabajo hemos creado un ensamble de modelos de clasificación con el objetivo de que sus resultados sean mejores que los que se obtendrían con un modelo simple y hemos experimentado con dos conjuntos de datos sus hiper-parámetros para observar cómo varía la evaluación de los resultados y buscar cuáles son los óptimos.**

***Palabras clave*—Inteligencia Artificial, Ensamble, Modelo, Muestreo, Sobreajuste**

I. Introducción

El aprendizaje automático es una rama de la inteligencia artificial que se especializa en el desarrollo de algoritmos y modelos capaces de aprender automáticamente, basándose en la experiencia en vez de instrucciones específicas. Dichos modelos están programados para, dado un conjunto de datos, extraer características relevantes y construir representaciones matemáticas capaces de interpretarlos. [1]

Una de las posibles aplicaciones que se le pueden dar a dichos algoritmos es la construcción de modelos predictivos. En ellos, el modelo intentará asignarle un atributo, al que se le denomina ”atributo objetivo”, a un dato a partir de los ejemplos proporcionados al entrenarlo. Un ejemplo muy conocido es el del filtro de mensajes no deseados de correo electrónico, donde, tras recibir como dato de entrada un mensaje, lo clasificará como legítimo o no deseado, basándose en los ejemplos de correos de ambos tipos que se le proporcionaron en su fase de entrenamiento.

Existen distintas maneras de clasificar los modelos predictivos. Si para los ejemplos de entrenamiento se conoce el atributo objetivo, se dice que el aprendizaje es ”supervisado”. En caso contrario, se dice que es ”no supervisado”. Dentro del aprendizaje supervisado, si el atributo objetivo es discreto, se dice que el modelo es de ”clasificación”, pero si el atributo es continuo, se dice que es de ”regresión”. En este trabajo trataremos modelos supervisados de clasificación, es decir, variable objetivo discreta y conocida en los ejemplos de entrenamiento.

El concepto del ensamble de modelos predictivos consiste en que, en vez de crear y entrenar un único modelo, se construyan varios sobre un mismo problema, combinando los resultados para obtener el modelo final (al que se le denomina meta-modelo), el cual proporcionará resultados más precisos que los de un único modelo. [2] Un esquema simplificado aplicado a modelos de clasificación binaria se representa en la Fig. 1.

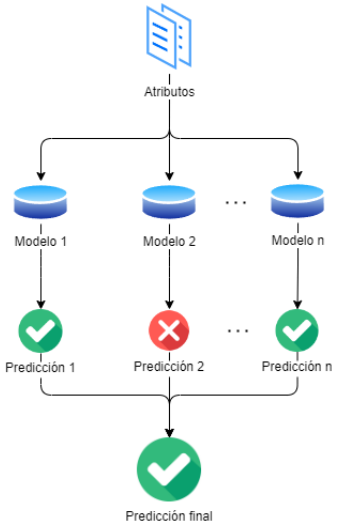

Fig. 1. Esquema de predicción de un ensamble. Imagen creada en draw.io.

Para que el ensamble proporcione buenos resultados, es necesario que cada uno de los modelos que lo componen sean, de alguna forma, diferentes entre sí, ya sea porque usen algoritmos diferentes o conjuntos de entrenamiento diferentes. En este proyecto nos centraremos en la segunda opción, ilustrada de manera simplificada en la Fig. 2. Sin embargo, asignarle un subconjunto del los datos de entrenamiento a cada modelo no es óptimo, ya que estos son limitados. Una posible solución es aplicar técnicas de muestreo sobre estos, y así, a pesar de que cada modelo parte del mismo conjunto, se entrena con datos únicos.

El objetivo del trabajo es construir el ensamble donde cada modelo tiene un conjunto de datos único y estudiar cómo sus meta-atributos (número de modelos, proporción de columnas por modelos, etc.) afectan a los resultados que proporcionan, para así buscar los más óptimos.

II. Preliminares

A continuación se indican las técnicas de muestreo empleadas para que cada modelo del ensamble tenga un conjunto único de entrenamiento:

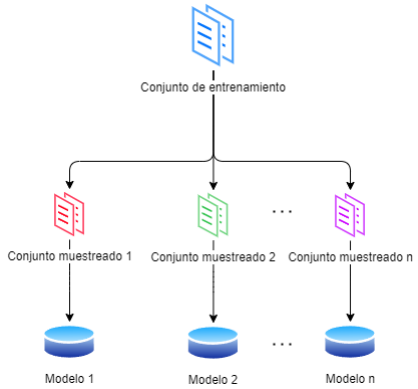

Fig. 2. Esquema de muestreo de datos de un ensamble. Imagen creada en draw.io.

- Muestreo de registros o filas (también denominado Bootstrapping [3]). En este tipo de muestreo, a partir de un conjunto de entrenamiento de tamaño N, se obtienen conjuntos de datos de tamaño N realizando muestreo con reemplazamiento, es decir, escogiendo N veces un elemento aleatorio del conjunto inicial, donde algunos se habrán escogido varias veces y otros ninguna, haciendo que cada conjunto de datos generado sea único.
- Muestreo de atributos o columnas (también denominado Random Subspace Method [4]). En este tipo de muestreo, se filtra un porcentaje de columnas aleatorias del conjunto de datos inicial. De esta manera, cada modelo del ensamble, además de tener datos de entrada únicos, podrá especializarse en una parte de los datos.

En el trabajo realizado, se ha implementado una combinación de los dos tipos de muestreo descritos, es decir, para cada modelo se ha aplicado primero Bootstrapping y luego muestreo por columnas al conjunto de datos de aprendizaje inicial. Como distintos ensambles se pueden crear dados el número de modelos que lo componen y el porcentaje de columnas escogidas en el muestreo por columnas, además de crear el ensamble, investigaremos con qué parámetros de entrada se obtienen mejores resultados.

III. Metodología

Esta sección se dedica a la descripción de la metodología implementada en el trabajo.

- Implementación del algoritmo
  1) *Creación:* Para crear el ensamble hemos creado una clase de Python, la cual hemos denominado `Ensamble`, cuyos atributos son el número de modelos del ensamble, la proporción de columnas por modelo y el modelo plantilla a usar.
  2) *Entrenamiento:* Para entrenar el ensamble, una vez creada la clase, se dividen el conjunto de datos en subconjunto de prueba y subconjunto de evaluación. Los datos de prueba se usarán para entrenar los modelos del ensamble y los de evaluación para evaluar las predicciones de este una vez entrenado. Una

`Ensamble`:
Parámetros:
- Número de modelos: n_m
- Proporción de columnas: P_c
- Modelo base: M_b

testearensamble(D)
Entrada: Conjunto de datos D
Salida: Puntuaciones S_{F1} y S_{BA}
$D \leftarrow$ Datos estandarizados de D
$D_t \leftarrow$ Datos de entrenamiento de D
$D_e \leftarrow$ Datos de prueba de D
ENTRENARENSAMBLE(D_t)
$S_{F1}, S_{BA} \leftarrow$ EVALUARENSAMBLE(D_e)
devolver S_{F1}, S_{BA}

entrenarensamble(D_t)
Entrada: conjunto de datos de entrenamiento D_t
repetir n_m **veces**
    $M \leftarrow$ clonar M_b
    añadir M al ensamble
por cada modelo M **dentro del ensamble**
    $A_M, O_M \leftarrow$ MUESTREAR(D_t)
    entrenar M con A_M, O_M

muestrear(D)
Entrada: conjunto de datos D
Salida: Atributos y valores objetivo A y O
$D_M \leftarrow$ realizar Bootstrapping en D
$A \leftarrow$ atributos sin valor objetivo de D_M
$O_M \leftarrow$ valores objetivo de D_M
$A_M \leftarrow$ realizar Random Subspace Method en A con P_c
devolver A_M, O_M

realizarprediccion(D)
Entrada: Conjunto de datos D
Salida: Lista de predicciones P
$P \leftarrow$ lista vacía
por cada modelo M **dentro del ensamble**
    $P_M \leftarrow$ predicciones con M sobre D
    Añadir P_M a P
$P \leftarrow P$ traspuesto
$P \leftarrow$ lista de valor más frecuente por cada sublista
devolver P

evaluarensamble(D_e)
Entrada: Conjunto de datos de evaluaciónD_e
Salida: Puntuaciones S_{F1} y S_{BA}
$A \leftarrow$ atributos sin valor objetivo de D_e
$O_V \leftarrow$ valores objetivo de D_e
$P \leftarrow$ REALIZARPREDICCION(A)
$S_{F1} \leftarrow$ obtener puntuación F1 a partir de O_V y P
$S_{BA} \leftarrow$ obtener puntuación B.A. a partir de O_V y P
devolver S_{F1}, S_{BA}

Fig. 3. Funciones de la clase `Ensamble`

vez divididos, se crean copias del modelo plantilla hasta obtener el número de modelo deseado. Acto seguido, se le crea a cada modelo del ensamble un conjunto de entrenamiento usando las técnicas mencionadas anteriormente, que usarán cada uno para su entrenamiento individual.

3) *Predicción:* Para realizar predicciones con el ensamble, se le proporciona un conjunto de datos, y para cada uno de ellos se obtiene la predicción dada por cada modelo, de las cuales se seleccionará la que más veces ha aparecido.
Los procesos de predicción, entrenamiento y evaluación se detallan en Fig. 3

- Experimentación

1) *Experimento de Referencia:* Para poder observar cómo al usar un ensamble de modelos predictivos mejoran los resultados obtenidos, se ha creado una función que cree y evalúe un modelo simple del mismo tipo de las que está compuesto el ensamble.
2) *Visualización:* Para visualizar los datos obtenidos, se ha usado la librería `matplotlib` [5], proporcionando los datos obtenidos de los ensambles para crear gráficas sobre su rendimiento.
3) *Mejores Hiper-parámetros del Ensamble:* Al crear un ensamble, se deben especificar cuántos modelos lo componen y la proporción de columnas de datos muestreadas. Para calcular los parámetros que dan mejores resultados hemos seleccionado durante la experimentación el valor del parámetro que resultó en la mayor puntuación. Para ello, hemos partido de un ensamble base, compuesto por 10 modelos y con una proporción de columnas muestreada en cada uno de 70%, sobre el cual hemos realizado varias evaluaciones con distintos parámetros de entrada, para así encontrar el óptimo. Además, para obtener resultados más precisos, todos los experimentos realizados se han realizado 20 veces, promediando los resultados obtenidos.
4) *Mejores Hiper-parámetros de los Modelos:* Al igual que el ensamble, los modelos que los componen tienen hiper-parámetros de entrada, para los cuales hemos buscado sus valores óptimos. En árboles de decisión hemos buscado la mejor profundidad máxima, y para clasificación linear de descenso por el gradiente, el mejor máximo número de iteraciones. También hemos incluido cómo afecta al modelo de referencia dichos hiper-parámetros, aunque hemos despreciado su valor óptimo cuando es distinto al del ensamble, ya que solo nos interesa el de este último.
5) *Resultados con Hiper-parámetros óptimos:* Por último, una vez calculados los hiper-parámetros óptimos para cada conjunto de datos y modelo del ensamble, hemos evaluado un ensamble y el modelo de referencia con dichos hiper-parámetros para comparar si el uso del ensamble mejora los resultados. También hemos comprobado si estos se sobreajustan, usando el conjunto de datos de entrenamiento como conjunto de datos de evaluación y observando si el rendimiento mejora, en cuyo caso significaría que se ha sobreajustado al conjunto de entrenamiento.

IV. Resultados

Como modelos, se han usado durante la experimentación árboles de decisión y clasificadores lineales mediante descenso por el gradiente.
El rendimiento se ha medido usando las medidas `f1_score` y `balanced_accuracy_score`, aunque se hará énfasis sobre la segunda medida.
Por defecto, los ensambles usan 10 modelos, entrenándolos con el 70% de las columnas dadas por el conjunto de datos. Adicionalmente, se crean ensambles con los mismos parámetros y se vuelven a entrenar y evaluar con el objetivo de conseguir una media en las puntuaciones. El número de repeticiones por defecto es 20, y el resultado final es el promedio obtenido tras dichas repeticiones. El modelo de referencia para el algoritmo de clasificador lineal mediante descenso por gradiente, sin embargo, proporciona resultados muy irregulares dependiendo de la semilla. Por ello, este ha sido promediado con 300 repeticiones en un intento de maximizar la precisión de los resultados.

A. Titanic

1) Árboles de decisión: Los resultados de este apartado se pueden observar en la Fig. 4.
El número de modelos causa indudablemente mayor rendimiento cuanto más modelos haya, aunque esta mejoría se hace menos notable a partir de 10 modelos, llegando a puntuaciones alrededor de 0.84.
La proporción de columnas tiene efectos mucho menos oscilantes, y parece estabilizarse a partir del 60% con una puntuación de 0.85, que es bastante cercana a la del modelo de referencia.
La profundidad de árboles tiene un efecto más inestable sobre las puntuaciones, aunque parece que a partir de la profundidad 10 no se obtiene en general ninguna mejora notable, obteniendo puntuaciones alrededor de 0.84.

2) Clasificador lineal de descenso por el gradiente: Los resultados de modificar los hiper-parámetros en ensambles de descenso por el gradiente se pueden observar en la Fig. 5.
Referente al número de modelos, el aumento de estos causa una mejora mayoritariamente constante, aunque no es tan notable a partir de 10 modelos, con puntuaciones aproximadas a 0.785.
Por otro lado, sobre la proporción de columnas, podemos observar que a partir del 50%, el ensamble llega a su rendimiento máximo e incrementar la proporción no resulta en mucha mejoría, estabilizándose a la puntuación de alrededor de 0.77.
Las máximas iteraciones no proveen una mejora clara como otros hiper-parámetros, sino que observamos que los valores

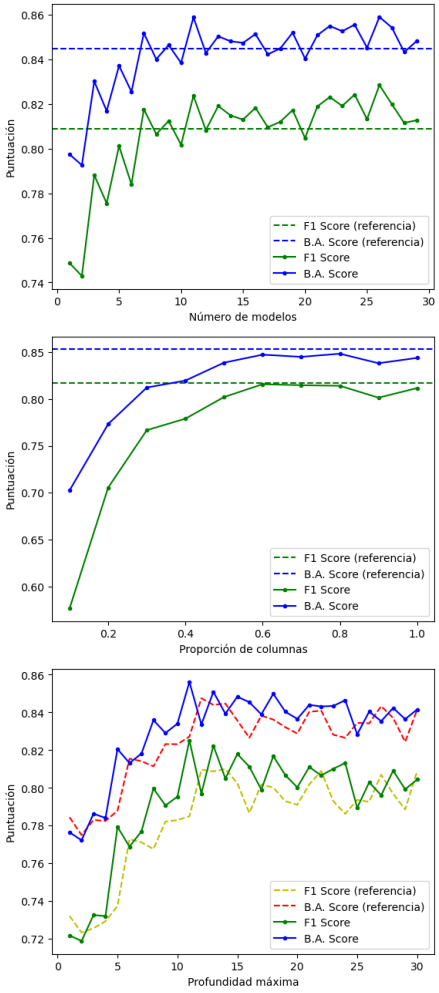

Fig. 4. Efecto de hiper-parámetros sobre ensambles de árboles de decisión. Conjuntos de datos del Titanic.

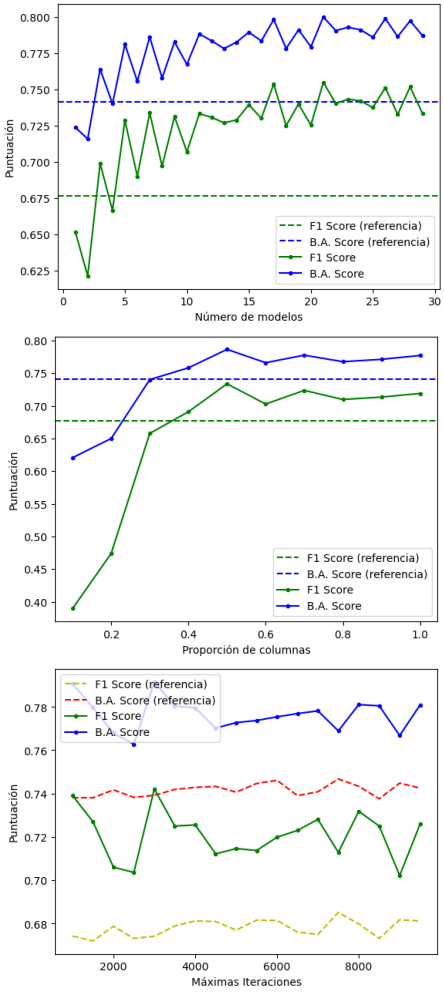

Fig. 5. Efecto de hiper-parámetros sobre ensambles de descenso por el gradiente. Conjuntos de datos del Titanic.

máximos se alcanzan en 1000 y 3000 iteraciones, con una puntuación cercana a 0.79.

B. PCOS

1) Árboles de decisión: Podemos observar los resultados de este apartado en la Fig. 6

En este conjunto de datos, observamos que hay una mejoría constante, aunque más lenta a partir de 5 modelos. Las puntuaciones aquí aspiran al 0.89.

La proporción de columnas vuelve a tener un efecto similar, estabilizándose ligeramente más temprano, alrededor del 40%.

En este conjunto de datos, la profundidad máxima deja de mejorar las puntuaciones mucho más temprano (a partir de profundidad máxima 5), además de tener oscilaciones menores. Las puntuaciones en ese rango ronda el 0.87.

2) Clasificador lineal de descenso por el gradiente: Podemos observar los resultados que causan los distintos hiper-parámetros en este apartado en la Fig. 7.

El número de modelos, como en otros apartados, causa un aumento no muy pronunciado de las puntuaciones, rondando 0.88.

De nuevo, la proporción de columnas tiende a no causar mejorías notables a partir del 40%, obteniendo puntuaciones similares al modelo de referencia con 0.86.

Las máximas iteraciones no causan ninguna mejora observable sobre la puntuación, sino que se mantienen constantes oscilaciones, por lo que las puntuaciones se mantienen alrededor de 0.87.

C. Parámetros óptimos

Una vez obtenidos los hiper-parámetros óptimos, las evaluaciones para cada algoritmo y conjunto de datos de, tanto los ensambles como el modelo de referencia, así como su tendencia al sobreajuste, se documentan en las tablas I,II,III y IV.

V. Conclusiones

A partir de esta experimentación hemos podido realizar las siguientes conclusiones:

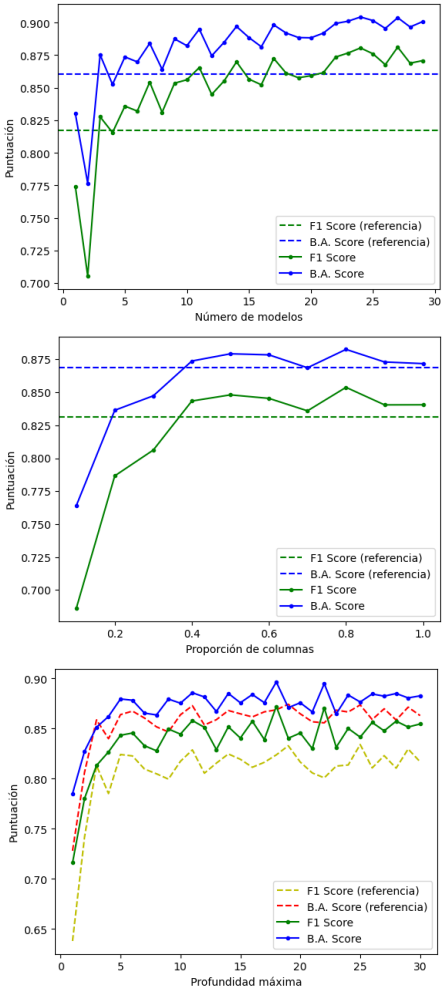

Fig. 6. Efecto de hiper-parámetros sobre ensambles de árboles de decisión. Conjuntos de datos del PCOS.

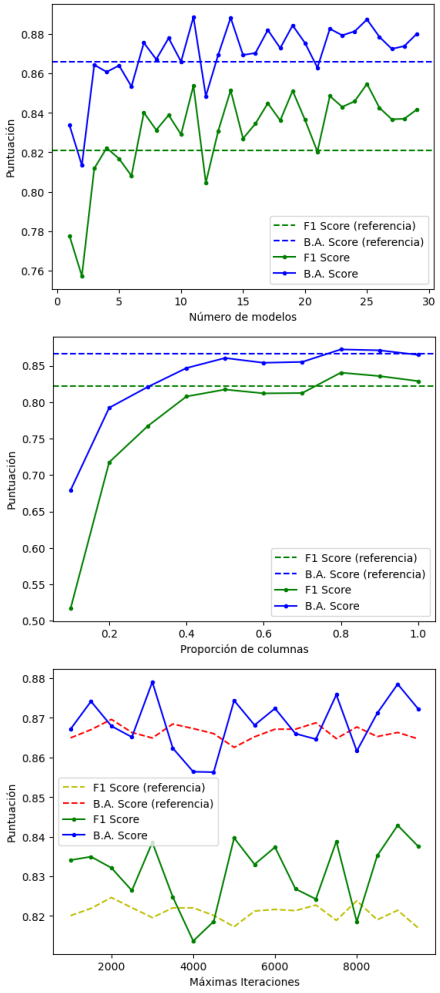

Fig. 7. Efecto de hiper-parámetros sobre ensambles de descenso por el gradiente. Conjuntos de datos del PCOS.

TABLA I
CONJUNTO DE DATOS TITANIC USANDO ÁRBOLES DE DECISIÓN

	Evaluado con conjunto de evaluación	Evaluado con conjunto de entrenamiento
Ensamble	0.8435612032312043	0.9817198771604503
Modelo de referencia	0.8278675705919667	0.9782073560318129

TABLA II
CONJUNTO DE DATOS TITANIC USANDO DESCENSO POR EL GRADIENTE

	Evaluado con conjunto de evaluación	Evaluado con conjunto de entrenamiento
Ensamble	0.7843114781380552	0.8174394559675129
Modelo de referencia	0.7517765624432862	0.7910310592436767

TABLA III
CONJUNTO DE DATOS PCOS USANDO ÁRBOLES DE DECISIÓN

	Evaluado con conjunto de evaluación	Evaluado con conjunto de entrenamiento
Ensamble	0.9134751850461573	0.999907063197026
Modelo de referencia	0.8728986119277169	1.0

TABLA IV
CONJUNTO DE DATOS PCOS USANDO DESCENSO POR EL GRADIENTE

	Evaluado con conjunto de evaluación	Evaluado con conjunto de entrenamiento
Ensamble	0.8838018369582125	0.9795633627577389
Modelo de referencia	0.8721533682998072	0.9721874913725674

1) Las máximas iteraciones en descenso por el gradiente son el hiper-parámetro que menos mejoras a la puntuación ha causado. Por lo cual, este parámetro es prácticamente irrelevante al entrenar ensambles.
2) La proporción de columnas causa peores puntuaciones cuando su valor es bajo y, cuando tiene valores altos, presenta una ventaja casi despreciable respecto al modelo de referencia, aunque se puede recalcar que puede equiparar rendimiento con valores alrededor del 50%.
3) El número de modelos es la manera más confiable de mejorar la puntuación, especialmente en ensambles de descenso por el gradiente, ya que es atributo que causa el mayor aumento en los resultados de la evaluación

respeto al modelo de referencia. Sin embargo, este hiperparámetro es el más costoso de incrementar, ya que el tiempo de entrenamiento incrementa con él.

4) El algoritmo de Árboles de Decisión tiene un sobreajuste muy alto, mientras que el de regresión lineal mediante descenso por gradiente tiene un sobreajuste muy bajo en el primer conjunto de datos, pero alto en el segundo.
5) El algoritmo de Árboles de Decisión proporciona mejores predicciones en ambos conjuntos de datos. Probablemente eso es debido a que el algoritmo de regresión lineal mediante descenso por gradiente es más óptimo para conjuntos de datos más grandes que los proporcionados, de los cuales ninguno superaba 1000 filas.
6) Cuando el ensamble tiene un número par de modelos, este tiende a proporcionar peores resultados, sobre todo cuando tiene una cantidad menor a 10. Probablemente el motivo de esto es que la variable objetivo es binaria en ambos conjuntos de datos y, cuando se produce un empate en las predicciones de los modelos, la predicción seleccionada por el ensamble es arbitraria. Cuantos más modelos tenga el ensamble, menor es la posibilidad de que ocurra un empate, por lo que los resultados son más constantes.

Referencias

[1] Aprendizaje automático. https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico
[2] Ensamble de modelos predictivos. http://www.cs.us.es/~fsancho/?e=106
[3] Bootstrapping: https://www.wikiwand.com/en/Bootstrapping_(statistics)
[4] Random Subspace Method: https://en.wikipedia.org/wiki/Random_subspace_method?oldformat=true
[5] Documentación de la librería `matplotlib` https://matplotlib.org/stable/index.html
[6] Documentación de la librería `pandas` usada para procesamiento de datos. https://pandas.pydata.org/docs/
[7] Documentación de `scikit-learn`, librería usada para modelos de aprendizaje automático. https://scikit-learn.org/stable/index.html
[8] Documentación de librería `tqdm`, usada para mostrar barras de progreso durante la evaluación de ensambles. https://tqdm.github.io