Estudio de eficacia, capacidad, optimización y resultado del ensamblado de modelos de aprendizaje simples

Daniel Díaz Nogales
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
UVUS: dandianog - Contacto: daniel0diiaz@gmail.com

Francisco Fernández Angulo
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
UVUS: fraferang - Contacto: francsico.fer.an@gmail.com

***Resumen*—El objetivo principal del trabajo es la realización y estudio de modelos de aprendizaje supervisados formados a partir de la composición secuencial de otros modelos de aprendizaje. Se busca estudiar la eficacia del ensamble de diferentes modelos de aprendizaje simples en un mismo modelo cuya capacidad sea mayor.**

Durante el estudio se ha desarrollado un prototipo de ensamble de diferentes modelos de aprendizaje de tipo regresión, y se ha probado con diferentes conjuntos de datos proporcionados. Se comprueba que la capacidad predictiva del modelo de aprendizaje ensamblado disminuye o aumenta en función de la cantidad de modelos de aprendizaje simples que lo componen, además de otros hiperparámetros que podremos especificar para cada modelo de aprendizaje supervisado.

***Palabras clave*—modelo de aprendizaje supervisado, ensamble, arboles de decisión, regresión**

I. Introducción

Buscando desarrollar un **modelo de aprendizaje supervisado fuerte** [7], y con el objetivo de aumentar el rendimiento de modelos de aprendizaje débiles, estudiamos en este proyecto cómo garantizar mejor resultado de predicción. Para ello, implementaremos **ensamblado de modelos de aprendizaje secuenciales**, que aumentará la capacidad del modelo final de entender conjuntos de datos más complejos y ofrecer mejores predicciones.

En los modelos de aprendizaje débiles, las predicciones normalmente no coinciden con el valor esperado aunque se aproximan (y es más frecuente a mayor aumente la complejidad del conjunto de datos). **Podemos reducir esta diferencia** entre los resultados reales y predictivos (a partir de ahora denominaremos *residuo* a esta diferencia) **uniendo diferentes modelos de aprendizaje que aprendan a optimizar el residuo de un modelo anterior** para así lograr que cada nueva iteración devuelva el menor posible.

Los modelos de aprendizaje permiten satisfacer dos tipos de problemas: **problemas de clasificación**, donde la variable objetivo tiene valores discretos, o **problemas de regresión** [1], donde la variable objetivo tiene un valor continuo. Nos centraremos en este último tipo de problemas en nuestro estudio. Además, se pueden utilizar diferentes tipos de modelo de aprendizaje de tipo regresión para el ensamble, e incluso combinarlos entre sí. Realizamos el estudio haciendo uso de diferentes modelos de aprendizaje, como el árbol de decisión o vecinos cercanos (*knn*) [5].

El **ensamble de modelos de regresión** [2] propuesto funcionará sobre un conjunto de datos dado, donde se seleccionarán las **columnas atributos** y **columna objetivo**. Se dividirá dicho conjunto de datos en 2 partes, formando un **conjunto de entrenamiento** y un **conjunto de evaluación**. Se entrentará un primer modelo con el conjunto de entrenamiento, y devolverá una primera predicción inicial evaluada con una matriz de valores medios de cada columna del conjunto de datos. Nos permitirá obtener nuestro primer residuo, con el que podremos entrenar nuestro siguiente modelo secuenciando la creación de sub-modelos, que aprenderán sobre el cálculo del residuo del modelo anterior.

Finalmente nuestro ensamble **devolverá una lista de modelos entrenados y la predicción final**. Ésta la utilizaremos para compararla con la predicción real y obtener una puntuación de calidad global.

La información del artículo ha sido clasificada en diferentes apartados. Una vez contextualizado el objeto de estudio, y la dinámica que seguirá el prototipo de ensamble propuesto, se profundiza más sobre cada concepto en los siguientes índices.

II. Preliminares

Se dispone de 3 conjuntos diferentes de datos [8]–[10] con los que se realizarán los experimentos en nuestro estudio. Son conjuntos de datos públicos, y consisten simplemente en columnas de atributos que nos servirán para entrenar nuestro ensamble de algoritmo, y una columna objetivo con las variables de respuesta. Los 3 conjuntos de datos utilizados se llaman **adultDataset** [8], **titanic** [9] y **BreastCancerDataset** [10].

A. Métodos empleados

Tras el estudio de los fundamentos de diferentes modelos de aprendizaje de regresión [5], [6], [15], y los problemas de **tipo regresión** [1], [2], planteamos el prototipo de forma simplificada como un problema dividido en 3 fases.

En la fase 1 se busca tener un primer modelo de aprendizaje entrenado con los conjuntos de datos proporcionados. Para ello, tendremos que elegir primero qué columnas serán los atributos de datos, y qué columna será la objetivo. Después, se preprocesará las columnas de atributos y objetivo con el fin de codificar [11], [12] los valores para que el algoritmo de aprendizaje del modelo pueda trabajar con ellos.

Con los datos codificados, **se dividen los conjuntos de atributos de datos y objetivos codificados en conjuntos de entrenamiento y de prueba** con el método *train_test_split* [13]. **Se define el primer modelo de aprendizaje** de clase *DecisionTreeRegressor* [2] o *KNeighborRegressor*, que construye el árbol con el método *fit* [16] a partir del conjunto de atributos y objetivo de la muestra de entrenamiento.

Se realiza la predicción con el método *predict* [14] del modelo de aprendizaje seleccionado cuyo conjunto de evaluación serán los valores medios de todas las columnas del conjunto de datos. Esta primera predicción nos servirá para la fase 2, donde se comenzarán a calcular residuos y los sub-modelos entrenarán a partir de éste.

La fase 2 se centra en crear nuevos sub-modelos que aprendan del residuo de anteriores. Iteraremos un algoritmo de aprendizaje, a partir de ahora será llamado meta-algoritmo, tantas veces como modelos queramos secuenciar en nuestro prototipo de ensamblaje. Este meta-algoritmo primero calculará el residuo de la predicción del árbol anterior. Luego obtendrá una muestra aleatoria sin reemplazamiento del conjunto de evaluación (de proporción hiperparámetro sample_size) y su correspondiente residuo. Estos se utilizarán para entrenar un nuevo sub-modelo de aprendizaje, con el que se evaluará en una predicción el conjunto de evaluación.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Fig. 1. Clasificación de los valores de la predicción en el rango [Xmin, Xmax] para que pertenezcan a las clases 1 o 0

En la fase 3 se busca recolectar los datos obtenidos por los modelos para poder estudiarlos. Del meta-algoritmo se obtiene una lista de árboles entrenados y una puntuación llamada *BalancedAccuracyScore* [17]. Esta puntuación es creada a partir de la diferencia entre los valores reales del conjunto de evaluación y la predicción del último modelo, y nos dará una idea de la calidad predictora del ensamble de modelos. Como nuestra predicción no nos dará valores exactos, y necesitamos clasificar sus clases, aplicamos a la predicción el método classify_prediction (sigue la función de clasificación del a figura 4.16. Con esto ejecutaremos diferentes pruebas a los diferentes conjuntos de datos. La figura 2 muestra la evolución de diferentes evaluaciones del ensamble cambiando la cantidad de modelos secuenciales.

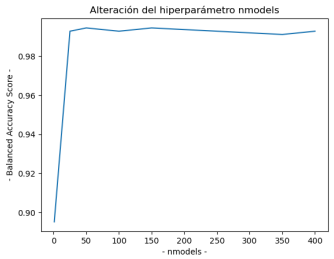


Fig. 2. Diferentes BalancedAccuracyScores del ensamblado cambiando hiperparámetro nmodels en el conjunto de datos Titanic.

Dentro de los ejemplos del estudio, utilizamos las **funciones elementales de Python y numpy** [18] para condicionamiento y procesado de datos, la librería **matplotlib** [19] para diferentes gráficas, varios **métodos de librerías de sklearn** [1]–[4], [11]–[17] para la definición de clases de modelos de aprendizaje y desarrollo, **pandas** [20] para el trabajo de ficheros.

III. METODOLOGÍA

Se han preparado dos clases en dos ficheros externos que son importados como módulo. Dicha clase se construye con los hiperparámetros del problema, y se ejecuta con su método start(X, y), cuyos parámetros son los conjuntos de datos de un dataset. Se explica la metolodología detalladamente por fases tal y como el anterior apartado.

Antes de entrar en la fase 1, definiremos las columnas de atributos y la columna objetivo que procesará nuestro ensamble. Se definirá un objeto de tipo *SequencialModelAlgorithm* y se le pasarán los hiperparámetros de ejecución.

1. Constructor de SequencialModelAlgorithm

Entrada:

- **nmodels**: número de modelos secuenciales
- **sample_size**: proporción de datos a coger en la muestra sin reemplazamiento del meta-algoritmo
- **max_depth**: profundidad en los árboles de decisión para devolver solución
- **lr**: factor de aprendizaje de los modelos secuenciales
- **min_samples_leaf**: mínimo de ejemplos por hoja en el modelo por árboles de decisión
- **max_features**: número máximo de características a considerar
- **min_weight_fraction_leaf**: fracción mínima ponderada de la suma total de pesos de las hojas

Salida:

- Objeto de tipo SequencialModelAlgorithm

Algoritmo:

1) Almacena los parámetros como atributos de la clase

Fijados estos parámetros, y entrando en la primera fase, se inicializa un método para comenzar el entrenamiento del primer modelo de aprendizaje y el meta-algoritmo.

2. Método start

Entrada:

- **attributes_cols**: columnas de atributos del conjunto de datos
- **objetive_col**: columna objetivo del conjunto de datos
- **random_state**: semilla para la muestra del conjunto de datos
- **ftest_size**: proporción para la división del primer conjunto de datos en conjunto de entrenamiento y de evaluación

Salida:

- Conjunto de modelos de aprendizaje débiles entrenado
- Puntuación de calidad *BalancedAccuracyScore*

Algoritmo:

1) Preprocesado de atributos y codificación con método *attributes_preprocess*
2) Preprocesado del objetivo y codificación con método *objetive_preproccess*
3) División de los atributos y objetivo en conjunto de entrenamiento y evaluación
4) Selección y definición del modelo de aprendizaje
5) Entrenamiento del modelo de aprendizaje con el conjunto de entrenamiento
6) Primera predicción evaluando el valor medio de las columnas de atributos
7) Ejecutar *nmodels* veces el método meta_algorithm y acumular cada árbol
8) Calcular la puntuación BalancedAccuracyScore de la última predicción con respecto a los datos de respuesta reales

Los submétodos utilizados en el método start para el preprocesado y codificación de las variables de los conjuntos de datos son:

3. Método attributes_preprocess

Entrada:

- **attributes_cols**: columnas de atributos

Salida:

- Columnas de atributos preprocesadas y codificadas

Algoritmo:

1) Para cada columna de atributos:
    a) Leer el primer valor del primer índice de la columna
    b) Si el primer valor de la columna no es de tipo numérico:
        i) Se codifica con el método *OrdinalEncoder*

4. Método objetive_preprocess

Entrada:

- **attributes_cols**: columna de valores objetivo

Salida:

- Columnas de valores objetivo preprocesadas y codificadas

Algoritmo:

1) Leer el primer valor del primer índice de la columna

2) Si el primer valor de la columna no es de tipo numérico:
    a) Se codifica con el método *LabelEncoder*

Finalmente dentro de la función start, tenemos el método **meta-algoritmo**, que recoge el aprendizaje de cada modelo secuencial, y el método classify_prediction, que clasifica la respuesta del cómputo total de predicciones. Esto correspondería con las fase 2 de iteración del meta-algoritmo, y la fase 3 de evaluación de datos

5. Método meta_algorithm

Entrada:
- **X**: columnas de valores atributo
- **y**: columna de valores objetivo
- **prediction**: predicción del modelo anterior

Salida:
- Modelo de aprendizaje entrenado
- Predicción del nuevo modelo entrenado

Algoritmo:
1) Calcula el residuo a partir de la diferencia de la columna objetivo y la predicción del anterior modelo
2) Se recoge una muestra aleatoria de tamaño proporción sample_size de los datos de atributos y objetivo
3) Se define el modelo de aprendizaje con los hiperparámetros
4) Se entrena el modelo con los datos de atributos y objetivo de la *muestra sin reemplazamiento*
5) Se realiza la predicción del actual modelo como la suma de la predicción anterior más la suma de la actual producto con el factor de aprendizaje *lr*

6. classify_prediction

Entrada:
- **pred**: predicción

Salida:
- predicción con valores clasificados

Algoritmo:
1) Calcula el valor máximo continuo que tiene la predicción
2) Calcula el valor mínimo continuo que tiene la predicción
3) Para cada índice de la predicción, realizamos la función de clasificado de la figura 2

Dentro del método meta-algoritmo se desarrollan los siguientes métodos:

7. sample_without_replacement

Entrada:
- **X**: conjunto de atributos de evaluación
- **res**: conjunto de residuos calculado
- **sample_size**: proporción a tener en cuenta para la cantidad de muestra

Salida:
- Muestra de atributos de proporción sample_size
- Muestra de residuos de proporción sample_size

Algoritmo:
1) Calcula el valor máximo continuo que tiene la predicción
2) Calcula el valor mínimo continuo que tiene la predicción
3) Para cada índice de la predicción, realizamos la función de clasificado de la figura 2

IV. Resultados

Se recogen diferentes resultados con diferentes pruebas y evaluaciones en los diferentes Dataset. Estas pruebas se evaluarán con los siguientes valores predeterminados:

- nmodels = 300
- sample_size = 0.65
- max_depth = 10
- lr = 0.1

Se realizan las siguientes evaluaciones para cada conjunto de datos:

- Prueba en función de nmodels: realizamos la prueba fijando los parámetros predeterminados y variando nmodels entre diferentes valores dependiendo del dataset
- Prueba en función de sample_size: se fijan los hiperparámetros predeterminados y se altera sample_size entre 0.1 y 0.9. Como influye poco en el resultado, y la aletoriedad es más fuerte que esta influencia del parámetro, se propone repetir 5 veces la prueba y devolver un valor medio entre las 5.
- Prueba en función de max_depth: se fijan los hiperparámetros predeterminados y se altera max_depth entre 1 y 20
- Prueba en función de lr: se fijan los hiperparámetros predeterminados y se altera lr entre 0.1 y 0.9. También se propone repetir 5 veces la prueba y devolver un valor medio entre las 5.
- Prueba en función de nmodels utilizando Vecinos Cercanos: se fijan los hiperparámetros predeterminados y se altera nmodels entre 1 y 350.
- Prueba en función de n_neighbors utilizando Vecinos Cercanos: se fijan los hiperparámetros predeterminados y se altera n_neighbors entre 1 y 5.

Además, según los datos obtenidos para estas diferentes pruebas, elegimos los valores que mejor determinen nuestro

resultado de predicción en nuestro ensamble de modelos. Evaluaremos este resultado en función de los recursos que necesite el algoritmo, el tiempo de ejecución y la certeza en su solución.

Para ver cómo el algoritmo funciona según la complejidad de los conjuntos de datos proporcionados, se divide en subsecciones con diferentes comparaciones de las evaluaciones en los diferentes conjuntos de datos.

A. Alteración del hiperparámetro nmodels

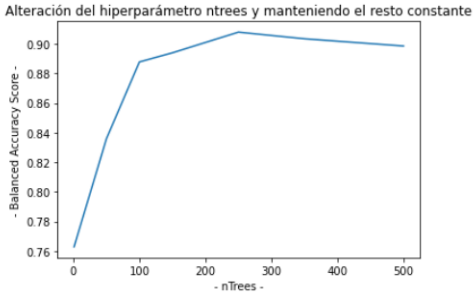
Variaremos la cantidad de modelos de aprendizaje que ensamblan el prototipo propuesto, y lo aplicaremos a diferentes datasets con el fin de sacar conclusiones sobre cómo interfiere la cantidad de modelos aplicados según la complejidad del conjunto de datos.

1. nmodels en adultDataset Árboles de Decisión

Hiperparámetros:

- **nmodels** = [1, 50, 100, 150, 250, 350, 500]
- **sample_size** = 0.65
- **max_depth** = 10
- **lr** = 0.1

Gráfico



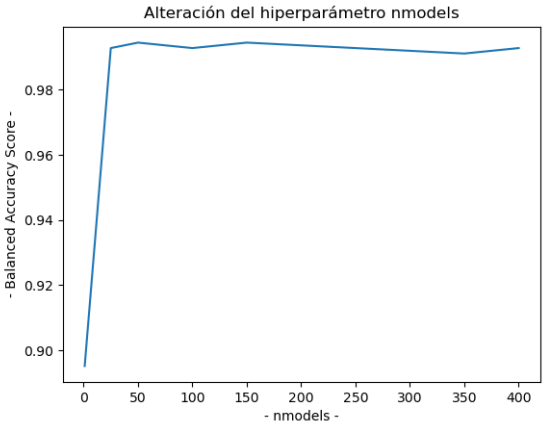
Comentarios adultDataset necesita entre 200 y 250 modelos de aprendizaje secuenciales para devolver la mejor puntuación. Es una función logarítmica, por lo que intuimos que no obtendremos mucha mejor puntuación a cambio de más modelos.
Por optimización podríamos decir que un buen valor para nmodels en adultDataset sería 250

2. nmodels en titanic.csv por Árboles de Decisión

Hiperparámetros:

- **nmodels** = [1, 50, 100, 150, 250, 350, 500]
- **sample_size** = 0.65
- **max_depth** = 10
- **lr** = 0.1

Gráfico



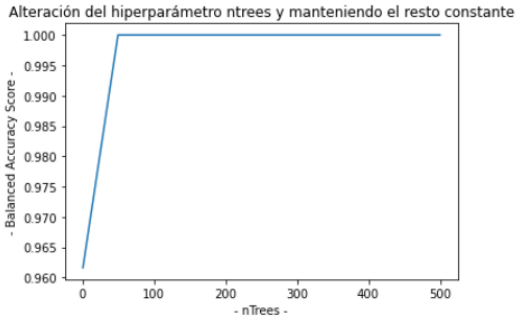
Comentarios titanic necesita aproximadamente 50 modelos de aprendizaje secuenciales para devolver la mejor puntuación. Es función logarítmica aún más marcada que en el anterior dataset, por lo que intuimos que no obtendremos mucha mejor puntuación a cambio de más modelos.
Por optimización podríamos decir que un buen valor para nmodels en titanic sería 50

3. nmodels en BreastCancerDataset Árboles de Decisión

Hiperparámetros:

- **nmodels** = [1, 50, 100, 150, 250, 350, 500]
- **sample_size** = 0.65
- **max_depth** = 10
- **lr** = 0.1

Gráfico



Comentarios BreastCancer necesita 50 modelos de aprendizaje secuenciales para converger en una puntuación de predicción máxima en 1. Por optimización podríamos decir que un buen valor para nmodels en titanic sería 50

Se repite el experimento haciendo uso de modelos de aprendizaje por vecinos más cercanos. Estas ejecuciones son mucho más largas según la cantidad de modelos en modelos más complejos como AdultDataset que usando árboles de

decisión.

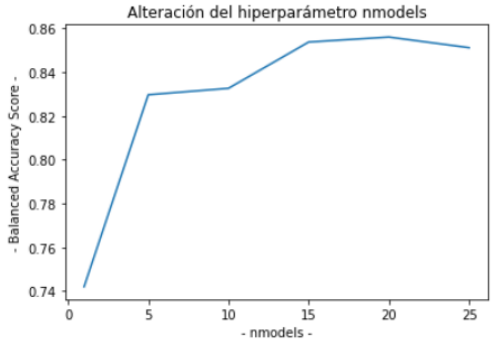
Por consecuente, para AdultDataset se limita nmodels a un máximo de 25 modelos, mientras que en el conjunto titanic o BreastCancer llegaremos a 350 modelos.

4. nmodels en adultDataset Vecinos Cercanos

Hiperparámetros:

- **nmodels** = [1, 5, 10, 15, 20, 25]
- **n_neighbors** = 1

Gráfico



Comentarios No podemos confirmar una función logarítmica en el modelo de aprendizaje por vecinos cercanos, pero la ejecución es mucho más lenta y aumenta según la cantidad de modelos de forma exponencial.

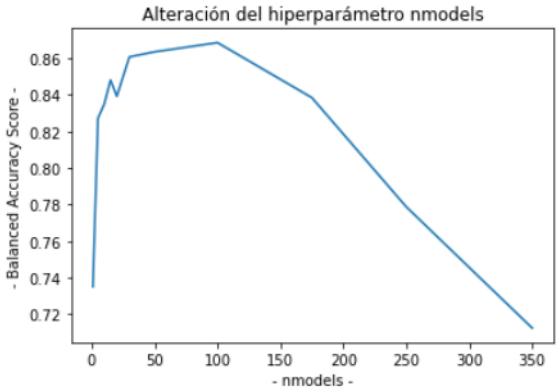
La base de datos titanic tiene menor complejidad, por lo que nos permite hacer ejecuciones según nmodels menos pesadas.

5. nmodels en titanic Vecinos Cercanos

Hiperparámetros:

- **nmodels** = [1, 5, 10, 15, 20, 30, 50, 100, 175, 250, 350]
- **n_neighbors** = 1

Gráfico



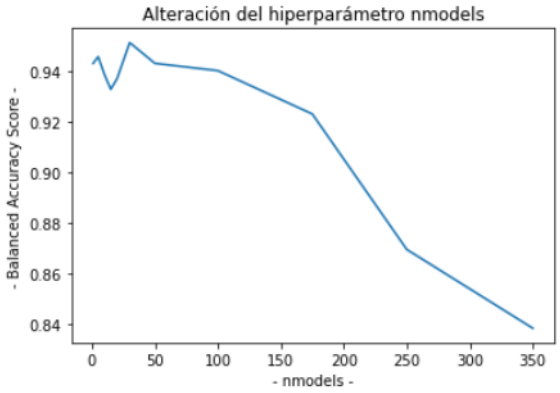
Comentarios La función nos muestra que el valor óptimo de modelos de aprendizaje secuenciales estaría entre 50 y 100. Llega un momento en el que cuanto más modelos peor resultado.

6. nmodels en BreastCancerDataset Vecinos Cercanos

Hiperparámetros:

- **nmodels** = [1, 5, 10, 15, 20, 30, 50, 100, 175, 250, 350]
- **n_neighbors** = 1

Gráfico



Comentarios La función nos muestra que el valor óptimo de modelos de aprendizaje secuenciales estaría entre 50 y 100. Llega un momento en el que cuanto más modelos peor resultado.

B. Alteración del hiperparámetro sample_size

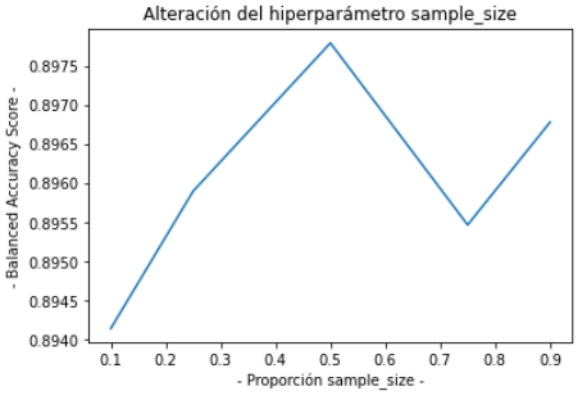
Se modifica la proporción de muestra que se utiliza para entrenar cada submodelo de aprendizaje secuencial. Estas pruebas se realizan 5 veces para mostrar en gráfica una media de los resultados con la intención de reducir el impacto que provocan resultados influidos por la aletoriedad.

7. sample_size en adultDataset Árboles de Decisión

Hiperparámetros:

- **nmodels** = 300
- **sample_size** = [0.1, 0.25, 0.5, 0.75, 0.9]
- **max_depth** = 10
- **lr** = 0.1

Gráfico



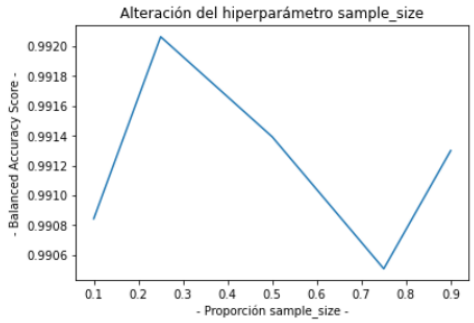
Comentarios La proporción de la muestra óptima está en 0.5

8. sample_size en titanic Árboles de Decisión

Hiperparámetros:

- **nmodels** = 300
- **sample_size** = [0.1, 0.25, 0.5, 0.75, 0.9]
- **max_depth** = 10
- **lr** = 0.1

Gráfico



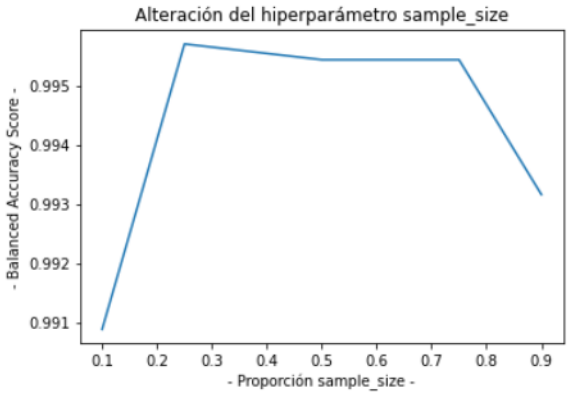
Comentarios La proporción óptima de la muestra está en 0.2

9. sample_size en BreastCancer Árboles de Decisión

Hiperparámetros:

- **nmodels** = 300
- **sample_size** = [0.1, 0.25, 0.5, 0.75, 0.9]
- **max_depth** = 10
- **lr** = 0.1

Gráfico



Comentarios La proporción óptima de la muestra está entre 0.2 y 0.8

C. Alteración del hiperparámetro max_depth

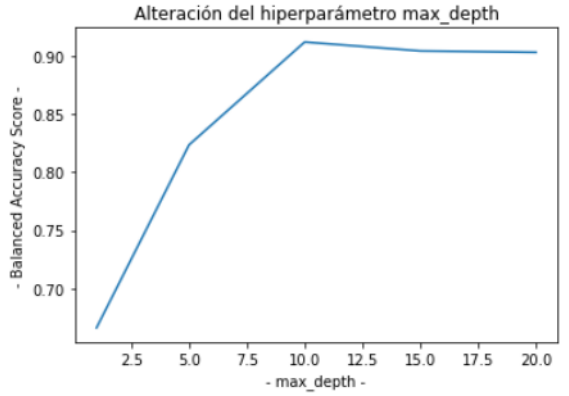
Cambia la profundidad en la que se recorren los árboles de decisión. Esto será muy influyente en el resultado ya que precisa mayores soluciónes por modelo de aprendizaje aunque hace el proceso más pesado.

10. max_depth en adultDataset Árboles de Decisión

Hiperparámetros:

- **nmodels** = 300
- **sample_size** = 0.65
- **max_depth** = [1, 5, 10, 15, 20]
- **lr** = 0.1

Gráfico



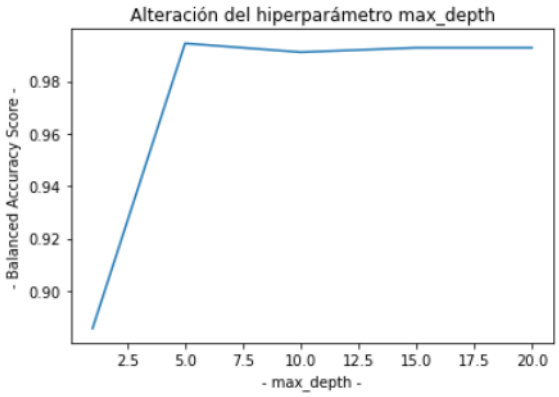
Comentarios El valor óptimo para el parámetro para la profundidad de árboles, max_depth, es 10.

11. max_depth en titanic Árboles de Decisión

Hiperparámetros:

- **nmodels** = 300
- **sample_size** = 0.65
- **max_depth** = [1, 5, 10, 15, 20]
- **lr** = 0.1

Gráfico



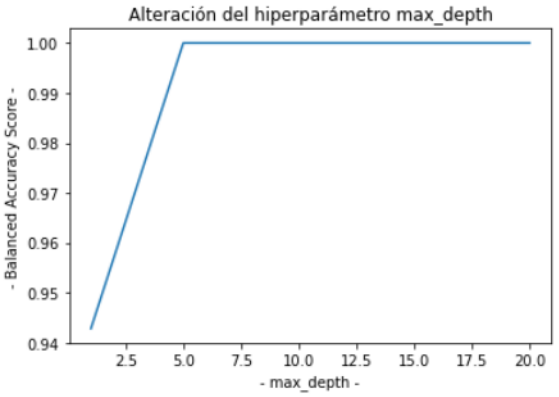
Comentarios El valor óptimo para el parámetro para la profundidad de árboles, max_depth, es 5.

12. max_depth en BreastCancer Árboles de Decisión

Hiperparámetros:

- **nmodels** = 300
- **sample_size** = 0.65
- **max_depth** = [1, 5, 10, 15, 20]
- **lr** = 0.1

Gráfico



D. Alteración del hiperparámetro ir

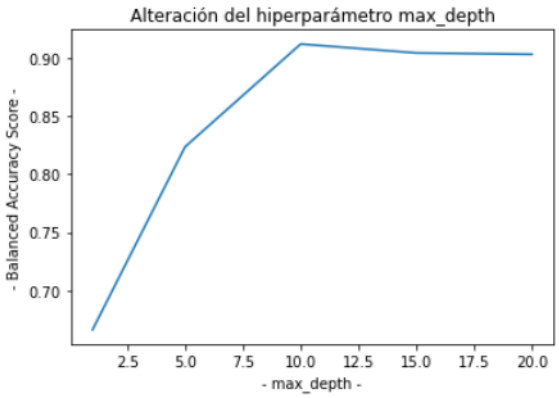
Se altera el valor de la proporción o factor de aprendizaje. Estas pruebas se realizan 5 veces para mostrar en gráfica una media de los resultados con la intención de reducir el impacto que provocan resultados influidos por la aletoriedad.

13. max_depth en adultDataset Árboles de Decisión

Hiperparámetros:

- **nmodels** = 300
- **sample_size** = 0.65
- **max_depth** = [1, 5, 10, 15, 20]
- **lr** = 0.1

Gráfico



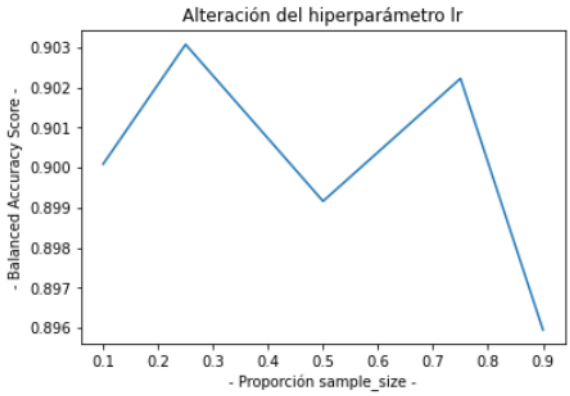
Comentarios El valor óptimo para el parámetro para la profundidad de árboles, max_depth, es 10.

14. lr en adultDataset Árboles de Decisión

Hiperparámetros:

- **nmodels** = 300
- **sample_size** = 0.65
- **max_depth** = 10
- **lr** = [0.1, 0.25, 0.5, 0.75, 0.9]

Gráfico



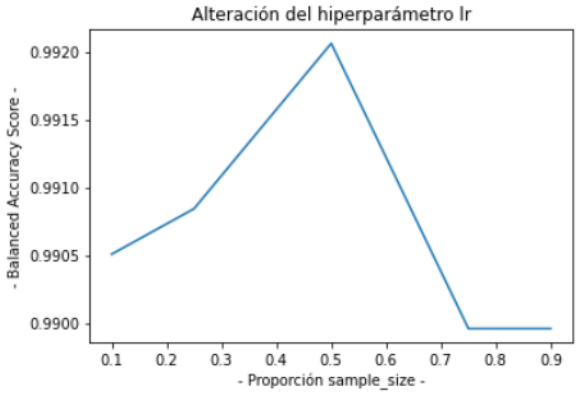
Comentarios Aunque influye mucho la aletoriedad sobre el ejemplo, el valor medio mejor evaluado sería 0.2 como factor de aprendizaje

15. lr en titanic Árboles de Decisión

Hiperparámetros:

- **nmodels** = 300
- **sample_size** = 0.65
- **max_depth** = 10
- **lr** = [0.1, 0.25, 0.5, 0.75, 0.9]

Gráfico



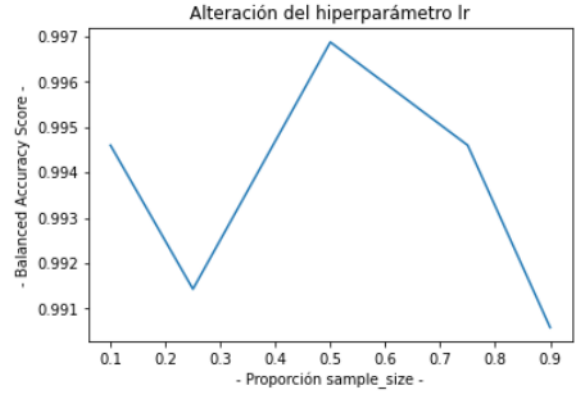
Comentarios Aunque influye mucho la aletoriedad sobre el ejemplo, el valor medio mejor evaluado sería 0.2 como factor de aprendizaje

16. lr en BreadCancerDataset Árboles de Decisión

Hiperparámetros:

- **nmodels** = 300
- **sample_size** = 0.65
- **max_depth** = 10
- **lr** = [0.1, 0.25, 0.5, 0.75, 0.9]

Gráfico



Comentarios Aunque influye mucho la aletoriedad sobre el ejemplo, el valor medio mejor evaluado sería 0.5 como factor de aprendizaje

E. Exploración de otros hiperparámetros

Se han implementado pruebas para otros hiperparámetros en los árboles de decisión y vecinos cercanos en las pruebas del prototipo en forma de cuaderno Jupyter.

V. Conclusiones

Tras el estudio concluimos que el ensamble de modelos de aprendizaje influye directamente en la calidad de predicción, tomando como valor de puntuación **BalancedAccuracyScore**.

El ensamble de modelos permite evaluaciones certeras en conjuntos de datos más complejos, como en *AdultDataset* donde llegamos a una puntuación de 0.92 con la secuenciación de 250 árboles de decisión frente a los 0.68 que resuelve un solo árbol de decisión. Además, en conjuntos de datos menos complejos como *titanic* o *BreastCancer*, conseguimos con apenas 50 secuenciaciones puntuaciones de 0.99 o incluso 1.

Experimentando con los hiperparámetros, se concluye primero que influimos en la puntuación de calidad en función logarítmica a nmodels y max_depth. Además son los parámetros con ajuste más preciso, se entiende fácilmente qué valor es mejor asignar a nuestro ensamble. Con el factor de aprendizaje, pero aún más con sample_size, tenemos ejemplos muy influidos por la aletoriedad del experimento, por lo que hemos tratado de hacer la prueba varias veces y recoger los valores medios que devuelva. Aún así no se aprecia una función de influencia tan directa en la predicción del algoritmo de aprendizaje.

Referencias

[1] Regresión en modelos de aprendizaje con sklearn https://scikit-learn.org/stable/modules/svm.html#regression
[2] Árboles de decisión con Regresión en sklearn https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html
[3] Método Shuffle de sklearn https://scikit-learn.org/stable/modules/generated/sklearn.utils.shuffle.html
[4] Función ColumnTransformer de sklearn https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html
[5] Árboles de decisión https://www.cs.us.es/cursos/rac-2018/temas/tema-05.pdf
[6] Árboles de decisión y algoritmo CART https://www.uv.es/mlejarza/actuariales/tam/arbolesdecision.pdf
[7] Modelos de aprendizajes fuertemente aprendedores vs modelos de aprendizajes débilmente aprendedores por ensamble https://machinelearningmastery.com/strong-learners-vs-weak-learners-for-ensemble-learning/
[8] Conjunto de datos AdultsDataset https://archive.ics.uci.edu/ml/datasets/adult
[9] Conjunto de datos titanic https://www.kaggle.com/c/titanic
[10] Conjunto de datos AdultsDataset https://archive.ics.uci.edu/ml/datasets/adult
[11] Codificación de objetivo LabelEncoder https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
[12] Codificación de atributo https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html
[13] Separación de conjunto de entrenamiento y de prueba en sklearn https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[14] Método predict de Árboles de Decisión de tipo Regresión https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.tree.DecisionTreeRegressor.predict

[15] Vecinos cercanos con Regresión https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html
[16] Métodos fit https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.tree.DecisionTreeRegressor.fit https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html#sklearn.neighbors.KNeighborsRegressor.fit
[17] Balanced Accuracy Score https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html
[18] Numpy https://numpy.org/
[19] Simple graphic plot with matplotlib.pyplot https://jakevdp.github.io/PythonDataScienceHandbook/04.01-simple-line-plots.html
[20] 10 Minutes Pandas https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html