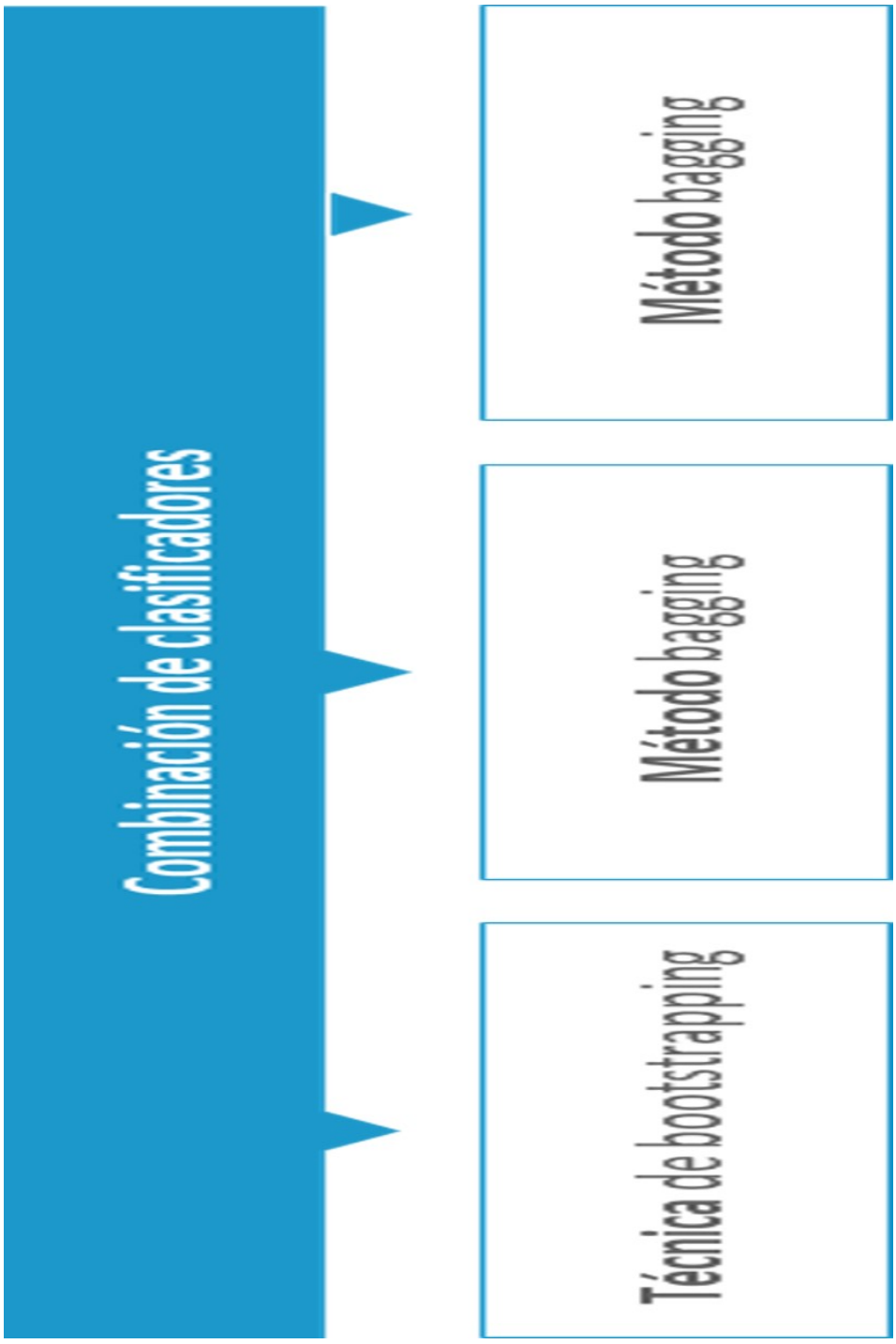
Técnicas de Aprendizaje Automático

Tema 9. Combinación de clasificadores. Bootstrapping, Bagging y Boosting

Índice

Esquema



9.1. Introducción y objetivos

En este tema se van a estudiar los métodos de ensamble. Estos métodos combinan las predicciones de varios estimadores base, que han sido construidos con un algoritmo de aprendizaje automático, con el fin de mejorar la generalización y robustez de un único modelo. El aprendizaje por ensamble de algoritmos combina muchos modelos débiles en un super modelo; sería algo similar a preguntarle a varios expertos sobre un problema en particular y construir una única respuesta.

Por lo general, los métodos de ensemble o combinación de clasificadores se pueden dividir en dos grandes familias:

- **Averaging methods o métodos de media:** los cuales se basan en construir varios estimadores de forma independiente y luego promediar las predicciones. El estimador combinado se suele comportar mejor que cualquier estimador individual porque la varianza se reduce. Los métodos de Bagging funcionan de esta forma.
- **Boosting methods o métodos de refuerzo:** el estimador base se construye de forma secuencial con el objetivo de reducir el sesgo del estimador combinado. La motivación es combinar varios modelos débiles para producir un modelo combinado potente.

Los objetivos que vamos a alcanzar en este tema son:

- Introducir la técnica estadística de Boostraping, que posteriormente se aplicará como método de muestreo en los modelos ensamblados.
- Describir los métodos del Bagging y el Boosting, modelos que llevan a cabo combinación de técnicas débiles con el fin de construir un modelo más robusto.

9.2. Técnica de muestreo Boostrap

La técnica de muestreo Boostrap o Boostrapping es una **técnica estadística** que consiste en un muestreo aleatorio con reemplazamiento. Si tenemos el reto de modelar una población, y se tiene una muestra de dos millones de ejemplos de diez millones que realmente son, decimos que existe una representación de la población.

La duda que surge cuando tenemos la muestra de una población es ¿la muestra representa el promedio real de toda la población? El Boostrapping resuelve ese problema: en lugar de calcular la media una sola vez, lo hace varias veces, y lo hace utilizando un remuestreo con reemplazo de la muestra original.

Boostrapping es la **práctica de estimar las propiedades de un estimador** (así como su varianza) por medio de medir las propiedades al muestrear una distribución aproximada. Una elección habitual para aproximar la distribución es utilizar la función de distribución empírica de los datos observados. En el caso de que se pueda asumir que las observaciones se generan a partir de una población independiente e idénticamente distribuida, se puede muestrear con reemplazamiento.

La idea sobre la que se basa la técnica de Boostrapping es que la inferencia sobre una población a partir de una muestra de datos puede ser **modelada mediante el remuestreo de la muestra,** y llevar a cabo inferencias sobre dicho remuestreo. Como la población es desconocida, el error verdadero en una muestra estadística es también desconocido. En el remuestreo en Boostrapping, la población es en realidad la muestra, la cual es conocida. Por tanto, la calidad de la inferencia de la muestra verdadera a partir de los datos muestreados es medible.

> Ejemplo de la altura de la población de una región.
>
> Como ejemplo, supongamos que nos interesa conocer la media de la altura de la población de una región. Debido a que es complejo medir la

altura de todas las personas, en su lugar podemos muestrear una parte de ellas y medir su altura. Supongamos que el tamaño de esta muestra es N, entonces tenemos la altura de N personas. Con esta muestra solo podemos tener una estimación de la altura media de la población. Para obtener una mejor imagen de la población, necesitamos obtener algún tipo de variabilidad sobre los datos obtenidos. El método de Boostrap más sencillo para obtener esta variabilidad consiste en muestrear con reemplazamiento una muestra de tamaño N. Por ejemplo, si muestreamos sobre [1,2,3,4,5] podríamos obtener [2,5,3,3,1]. Cuando el tamaño N es suficientemente grande para todos los efectos prácticos, existe una probabilidad casi cero de que sea una muestra idéntica a la muestra inicial, por lo que este proceso se repite varias veces. Después, obtenemos la media y la varianza de cada conjunto para finalmente sacar la media de las estimaciones anteriormente calculadas. Se puede decir que la media de la muestra es en realidad la estimación de la media para toda la población.

La estadística Boostrap resuelve el problema cuando no tenemos realmente la población completa. Las estadísticas que se obtienen de una muestra representan las estadísticas de la población general.

9.3. Método Bagging

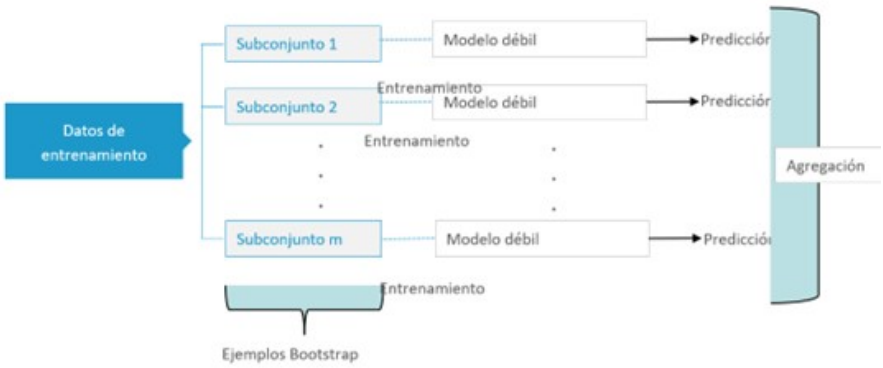


Figura 1. Método Bagging. Fuente: elaboración propia.

El método de Bagging es una clase de algoritmo de ensamble o de combinación de clasificadores, como podemos observar en la Figura 1. El término Bagging proviene de la contracción de *Boostrap aggregation*. Se trata de un procedimiento general que permite reducir la varianza de un método de aprendizaje automático. Es un método para **combinar varias instancias de estimadores** que se han construido sobre muestras aleatorias del conjunto de entrenamiento original, estas muestras se crean utilizando Boostrap; y, posteriormente, se agregan las predicciones individuales para obtener una predicción única.

Dado un conjunto de n observaciones independientes Z1…Zn, cada una con una varianza σ, la varianza de la media Z de las observaciones es σ/n. Es decir, la media de un conjunto de observaciones reduce la varianza. Sin embargo, como lo habitual es no tener muchas observaciones, se suele hacer Boostrap tomando muestras repetidas del conjunto de datos de entrenamiento.

Para ello, se entrenan B conjuntos de entrenamiento distintos y se obtiene la **media de las predicciones**, como se representa en la ecuación (1):

$$\widehat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\widehat{f}^{b}(x)$$

Estos métodos se utilizan como una forma de reducir la varianza de un estimador base; por ejemplo, árboles de decisión, e introducir aleatoriedad en el procedimiento de construcción y, a continuación, construir el ensamble. La probabilidad de sobreajuste disminuye porque la varianza se reduce; algunos ejemplos se repiten en las muestras, lo que hace que la varianza disminuya.

Los métodos de Bagging funcionan mejor cuando se utilizan con modelos complejos, por ejemplo, árboles de decisión profundos, en contraste con los métodos de Boosting que funcionan mejor con modelos simples, por ejemplo, shallow decision trees.

El Bagging también puede reducir la posibilidad de que existan valores atípicos, la probabilidad de tenerlos en nuestro conjunto de entrenamiento es muy baja, ya que son minoría.

Los pasos que seguir en la creación de un modelo utilizando la técnica de Bagging son:

- Tenemos la muestra de entrenamiento y, a partir de esta, creamos muestras Boostrap, datasets de entrenamiento del mismo tamaño. El primer dataset será el input para el primer clasificador o regresor, la segunda muestra será el input para el segundo clasificador o regresor, y así sucesivamente.

- Se combinan todos los clasificadores débiles en uno fuerte y lo probamos con nuestro conjunto de datos de prueba para realizar la predicción. El resultado de la predicción final es la suma de las predicciones de los modelos individuales, dividido entre el número de predictores si estamos frente a un modelo de regresión, y para la clasificación es la clase que más se repite en las predicciones de los modelos individuales o voto mayoritario.

Ideas clave

Existen **diferentes variantes** de los métodos de Bagging, pero dependen principalmente de la forma en que obtienen las muestras del conjunto de entrenamiento:

- Cuando se obtienen muestras aleatorias del conjunto de datos como subconjuntos aleatorios de las muestras, el algoritmo se conoce como *pasting* (Breiman, 1999).
- Cuando se obtienen los conjuntos para entrenar los modelos por medio de muestreo con reemplazo del conjunto de entrenamiento, el método se conoce como *Bagging* (Breiman, 1998).
- Cuando se obtienen subconjuntos aleatorios como subconjuntos del espacio de las variables, el método se conoce como *random subspaces* (Kam, 1998).
- Por último, cuando el estimador base se construye con subconjuntos de muestras y variables, el método se conoce como *random patches* (Louppe y Geurts, 2012).

Out-of-Bag error (OOB)

El error de test en un modelo Bagging se aborda utilizando el OOB, se trata del error medio de predicción de cada muestra de entrenamiento. Este error se calcula usando la muestra de datos que no se utilizan para el entrenamiento. Es decir, se tiene una muestra de entrenamiento de Boostrap, el error OOB se obtiene de los valores faltantes de la muestra original que no se usaron en la muestra Boostrap; esto se denomina error *out-of-bag* o **error fuera de la bolsa**. Por lo anterior, no es necesario utilizar la validación cruzada o retener muestras para obtener una estimación del error, el Bagging facilita esta acción.

Cuando el número de conjuntos de entrenamiento distintos es grande, la estimación del *out-of-bag* error es equivalente al *leave-one-out cross validation* que utiliza un único dato como conjunto de prueba y el resto de datos disponibles como conjunto de entrenamiento. Este proceso se repite hasta que los datos hayan sido utilizados como conjunto de prueba exactamente una vez.

Ejemplo de OOB.

Si se tienen 20 datos de test y el modelo clasifica 5 instancias de manera incorrecta, la exactitud sería: 15/20*100=75 %. Así calcularíamos el error, el OOB es muy intuitivo y fácil de entender.

9.4. Método de Boosting

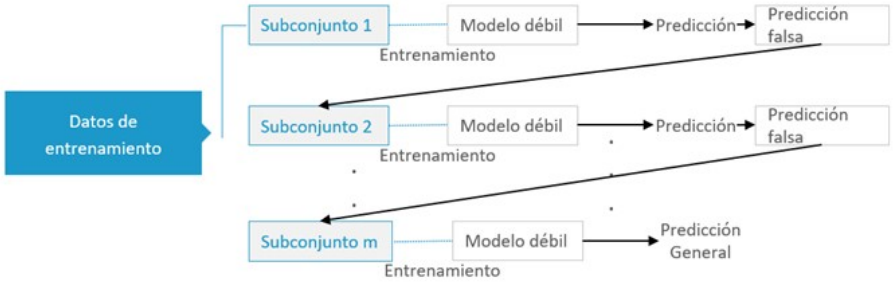


Figura 2. Método Boosting. Fuente: elaboración propia.

Al igual que Bagging, se trata de un método de combinación de modelos que se puede aplicar a los modelos de regresión y de clasificación.

En Bagging se crean múltiples copias del conjunto de entrenamiento original utilizando el muestreo Boostrap y entrenando un modelo con cada copia, para posteriormente combinar todos los modelos en uno solo; cada modelo se construye en un Boostrap data set independiente de los otros. Sin embargo, en el caso de Boosting los modelos se generan de forma secuencial, es decir, cada modelo utiliza información de los modelos anteriores, como podemos ver en la Figura 2.

Boosting es la técnica seguida por los modelos de *generalized boosted models* (GBMs). En estos modelos, en lugar de entrenar un gran árbol de decisión sobre los datos, lo cual es complicado y puede dar lugar a *overfitting* o sobreajuste, se utiliza el enfoque Boosting para aprender poco a poco. Cada uno de los árboles puede ser bastante pequeño, con pocos nodos terminales, pero todos ellos se combinan. Es un modelo que se entrena de manera secuencial. Se siguen los siguientes pasos:

- Se entrena un modelo con las muestras del conjunto de datos original.

- Se evalúan los errores que se cometieron en la predicción del primer modelo, el segundo modelo tomará otra muestra donde los inputs tendrán pesos diferentes, tendrán más peso aquellas instancias que no se clasificaron correctamente en el primer modelo. Así sucesivamente, se van creando modelos con instancias que tienen diferentes pesos dependiendo de la predicción del modelo anterior.

Por lo tanto, cada vez que se añade un nuevo modelo débil, los datos son recalibrados: los ejemplos que estaban correctamente clasificados pierden peso y los ejemplos incorrectamente clasificados ganan peso. El método puede verse como un meta algoritmo de ensamble para reducir sesgo y varianza, se puede aplicar sobre modelos supervisados con el objetivo de convertir modelos simples en modelos más precisos. Está basado en una pregunta formulada por Kearns y Valiant (1989): «¿Puede un conjunto de modelos débiles crear un modelo fuerte?».

La respuesta a la pregunta de Kearns y Valiant ha tenido una gran repercusión en el mundo de *machine learning* dando lugar al desarrollo de los modelos Boosting. Cuando se introdujo esta hipótesis de problemas, Boosting dio lugar a que los algoritmos que consiguieran mejorar por medio de la combinación de modelos débiles fueran llamados Boosting. A pesar de que los algoritmos Boosting no tienen ninguna restricción algorítmica, la mayoría de estos algoritmos consisten en ir aprendiendo modelos débiles iterativamente con respecto a una distribución.

Tipos de algoritmos Boosting

Existen diversos algoritmos de tipo Boosting. El algoritmo original fue propuesto por Robert Schapire y Yoram Singer en 1996, y publicado en el congreso de aprendizaje automático COLT (Conference on Learning Theory), y se titula *Improved Boosting Algorithms Using Confidence-rate Predictions*. El algoritmo original no era adaptativo y no explotaba de forma completa la ventaja de los modelos débiles. Posteriormente, Freund y Schapire desarrollaron el algoritmo AdaBoost, un algoritmo de Boosting adaptativo que ganó el prestigioso premio Gödel.

Ideas clave

La principal variación entre los algoritmos de Boosting es el **método de ponderar los datos de entrenamiento y las hipótesis**. AdaBoost es muy popular e históricamente el primero que fue capaz de adaptarse a los modelos débiles. Sin embargo, hay más algoritmos de Boosting, como el LPBoost, BrownBoost, TotalBoost, LogitBoost y XGboost.

AdaBoost

Este modelo combina clasificadores débiles y aprende progresivamente de las instancias clasificadas en el modelo anterior, por eso se le llama adaptativo. Se puede utilizar con cualquier clasificador, se le suele llamar «el mejor clasificador listo para usarse (*out-of-the-box* classifier)» (Leduc, 2019).

El primer paso de este algoritmo es tener un clasificador débil, luego crear un modelo por cada variable del conjunto de datos. Las muestras clasificadas incorrectamente tendrán más peso para que en la siguiente decisión puedan ser clasificadas correctamente. En este algoritmo también se le asignan pesos a cada uno de los clasificadores, en función de su precisión: precisión alta peso alto. Cuando se asignan pesos a los clasificadores significa que un clasificador nos da mayor precisión que otros; pero como se tienen pesos más altos para algunas muestras, significa que estas muestras están clasificadas incorrectamente. Se repiten estos pasos con los mismos clasificadores por variable hasta que se hayan clasificado todas las instancias correctamente o hasta alcanzar el máximo número de iteraciones.

Ejemplo del algoritmo AdaBoost

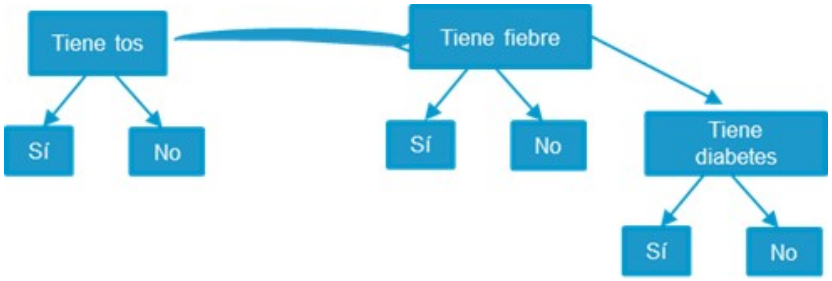


Figura 3. Clasificador AdaBoost para predicción de gravedad del COVID-19. Fuente: elaboración propia.

Tenemos tres clasificadores débiles, uno por cada variable del conjunto de datos. El hecho de tener tos es un síntoma de COVID-19, este clasificador dará una predicción de si el paciente puede estar en riesgo de un COVID-19 grave o no, tener fiebre también es un síntoma de la enfermedad y también dará una predicción. Tener diabetes no es un síntoma de la enfermedad, pero seguramente este clasificador tendrá mayor relevancia a medida que itera el modelo, porque sí es una enfermedad de base que puede agravar el COVID-19.

En el primer paso, cada clasificador dará su predicción por instancia; dependiendo del valor que se tenga por variable, la decisión final será el voto mayoritario de los tres clasificadores. En la siguiente iteración las instancias mal clasificadas tendrán más peso y el clasificador que tenga mejor predicción tendrá más peso. Así se itera hasta llegar al máximo número de iteraciones o hasta que todas las instancias estén clasificadas correctamente, para finalmente tener un clasificador más robusto que combina varios clasificadores débiles.

Veamos ahora la ecuación de AdaBoost (2):

$$\alpha^i = \log \frac{(1 - ErrorTotal)}{ErrorTotal}$$

Alfa es la influencia que tendrá un clasificador débil en la decisión final, es el peso de cada clasificador débil; y t es el clasificador actual. El ErrorTotal es el número de clasificaciones erróneas para ese conjunto de entrenamiento dividido por su tamaño, porcentaje de las muestras clasificadas incorrectamente. Si alfa es un número positivo, significa que la tasa de error disminuye a medida que alfa aumenta, esto va indicando un clasificador débil perfecto. Cuando alfa es positivo, el resultado real y predicho coinciden, es decir, la muestra se ha clasificado correctamente. En este caso, se disminuye el peso de la muestra con respecto al anterior. En cambio, si alfa es negativo significa que el valor predicho no coincide con el real; en este caso, se aumenta el peso para la muestra.

El funcionamiento del AdaBoost se da en función de un valor alfa para cada clasificador débil, y una vez se ha encontrado este alfa utilizando el error total, se actualizan los pesos para cada muestra evaluada con este clasificador, aumentando el peso si está bien clasificada o disminuyendo si está mal clasificada.

- Pseudocódigo

```
Pesos iniciales W: W1, W2 … Wn= 1/n

For i in [1,M]/M clasificadores débiles

    Ajustar clasificadores débiles con los pesos W

    ⟦Error⟧ ^i= (∑_(j=1)^n▒⟦W_j I(C^i (X_j)≠Y_j)⟧ )/(∑_(j=1)^n▒W_j )

    α^i=log ⟦((1-ErrorTotal))/ErrorTotal⟧ +log (K-1) coeficiente para Ci

    W_j=W_j*e^(α^i*I(C^i (X_j )≠Y_j ) ) para cada W_j∈W si error,
incremente el peso

    W=W-media(W) normalizar los pesos
```

La salida del modelo es por voto mayoritario.

Predicción: Ŷ_j=max_k(∑_(j=1)^i▒〖α_i〗 ^ I(C^i (X_j)=k))

Ejemplo en Python

A través de la librería de Scikit Learn existe la implementación de AdaBoost (Zhang, 2019).

```
from sklearn.ensemble import AdaBoostClassifier

from sklearn.datasets import make_classification

X,Y = make_classification(n_samples=100, n_features=2, n_informative=2,

n_redundant=0, n_repeated=0, random_state=102)

clf = AdaBoostClassifier(n_estimators=4, random_state=0, algorithm='SAMME')

clf.fit(X, Y)
```

Parámetros

base_estimator : el algoritmo base para generar los modelos, por defecto tiene el DecisionTreeClassifier(max_depth=1)

n_estimators : es un entero, por defecto está en 50.

learning_rate : tasa de aprendizaje que en este algoritmo es la contribución de cada clasificador. De tipo float, por defecto está en 1.

Random_state : semilla para la aleatoriedad del algoritmo.

Gradient Boosting

Este algoritmo en lugar de ajustar los pesos de los datos se enfoca en la diferencia entre la predicción y el valor real. Para mejorar el modelo minimizamos la función de pérdida, la cual es diferencial y funciona tanto para la regresión como para la clasificación.

- Pseudocódigo

```
Ajuste estimador F1

For i in [1,M]/M estimadores débiles

        〖Loss〗^i=∑_j^n▒〖(Y_j-F^i (X_j))〗^2# Función de pérdida en la ith
iteración

Calcular neg gradiente: - 〖∂L〗^i/〖∂x〗_j =-2/n*(Y_j-F^i (X_j))∀i

ajustar un estimador débil Hi sobre (X,∂L/∂X)

#ρ cambia el tamaño del paso

predicción=
```

> Ejemplo en Python
>
> A través de la librería de Scikit Learn existe la implementación de Gradient Boosting (Zhang, 2019).
>
> ```
> from sklearn.ensemble import GradientBoostingRegressor
>
> model = GradientBoostingRegressor(n_estimators=3,learning_rate=1)
>
> model.fit(X,Y)
>
> # for classification
>
> from sklearn.ensemble import GradientBoostingClassifier
> ```

```
model = GradientBoostingClassifier()

model.fit(X,Y)
```

Parámetros

loss : {'deviance', 'exponential'}, opcional (defecto='deviance').

learning_rate : tasa de aprendizaje, qué tan rápido aprende el modelo. Cada modelo agregado modifica el modelo, la magnitud de esa modificación está controlada por la tasa de aprendizaje. Float, opcional (default=0.1).

n_estimators : int (default=100).

Gradient Boosting no suele caer en sobreajuste, por lo que un gran número de estimadores da un mejor rendimiento.

subsample : float, optional (default=1.0). Se usa para ajustar a los modelos individuales; subsample interactúa con el n_estimators. Elegir una submuestra <1.0 conduce a una reducción de varianza y un aumento del sesgo.

criterion : string, optional (default=" friedman_mse "). Función para medir la calidad de una división. El friedman_mse utiliza el error cuadrático medio. Otra opción que utilizar en lugar del friedman_mse es squared error .

XGBoost

Es una implementación del gradient Boosting diseñado para trabajar a una mayor velocidad y obtener un mejor rendimiento.

Veamos cómo instalar XGBoost en Python y cómo aplicarlo en el conjunto de datos de pima-indians-diabetes. El conjunto de datos proviene del Instituto Nacional de

Diabetes y Enfermedades Digestivas y Renales. El objetivo es predecir el diagnóstico de diabetes, en función de ocho variables de entrada que describen los detalles médicos de los pacientes. Se impusieron varias restricciones a la selección de las instancias de una base de datos más grande. En particular, todos los pacientes aquí son mujeres de al menos 21 años de herencia indígena pima. Es un buen conjunto de datos para aplicar el modelo de XGBoost porque las variables son numéricas y el problema es de clasificación binaria.

Puedes descargar el conjunto de datos a través del siguiente enlace: https://www.kaggle.com/datasets/kumargh/pimaindiansdiabetescsv

Ejemplo de OOB

Instalamos el XGBoost haciendo uso del comando pip install.

```
!pip install xgboost

#importar librerías necesarias

import pandas as pd #pandas nos permite leer los archivos CSV

import numpy as np

import seaborn as sns

import xgboost as xgb

from sklearn.metrics import classification_report

#Leer el conjunto de datos

df_diabetes=pd.read_csv('diabetes.csv')

#Preparar el conjunto de datos de test y entrenamiento y Separar la #etiqueta del dataset
```

```
X= df_diabetes.drop(['Outcome'],axis=1)

Y= df_diabetes.Outcome

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

#Entrenar el modelo

Xgb_model = xgb.XGBClassifier(max_depth=5, learning_rate=0.05).fit(X_train,y_train)

#Predecir y evaluar el modelo

y_pred=xgb_model.predict(X_test)

print(classification_report(y_test, y_pred))
```

9.5. Stacking

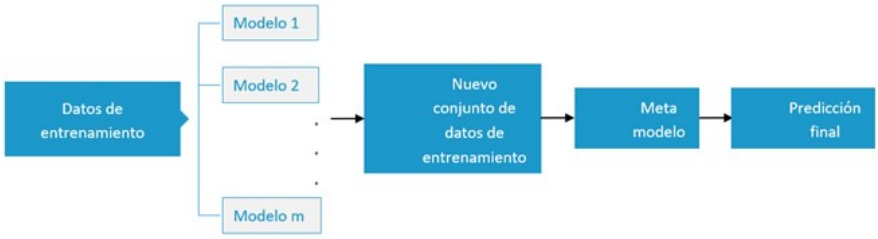


Figura 4. Método Stacking. Fuente: elaboración propia.

El método stacking es una forma muy interesante de combinar modelos. Aquí se usan varios modelos. Los pasos para seguir en este método son los siguientes:

- Con el conjunto de datos de entrenamiento se entrenan m modelos.
- Usando la salida de cada uno de los modelos, se crea un nuevo conjunto de datos de entrenamiento.
- Con el nuevo conjunto de datos, se crea el metamodelo.
- Usando los resultados del metamodelo, se realiza la predicción final.

9.6. Comparación entre modelos de ensamble

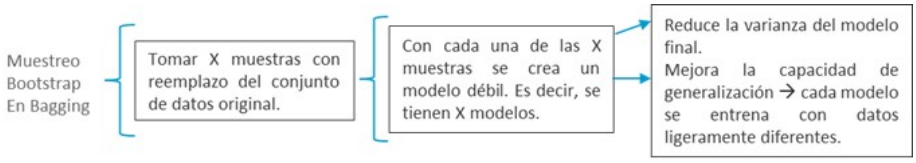
En la Tabla 1 podemos ver la comparación entre los algoritmos de ensamble, su propósito, el aprendizaje base, el entrenamiento base y la forma en que se agregan los modelos.

	Bagging	Boosting	Stacking
Propósito	Tiende a reducir la varianza	Tiende a reducir el sesgo	Mejora exactitud
Aprendizaje Base	Homogéneo	Homogéneo	Heterogéneo
Entrenamiento Base	Paralelo	Secuencial	Secuencial
Agregación	Voto mayoritario, Promedio	Peso promedio	Promedio ponderado

Tabla 1. Comparación de algoritmos de ensamble. Fuente: elaboración propia.

9.7. Cuaderno de ejercicios

- Explica a través de un esquema en qué consiste el muestreo Bootstrap y cómo se aplica en el contexto de bagging. ¿Cuál es su propósito? ¿cómo ayuda a mejorar la robustez y la generalización del modelo?



- Menciona una diferencia entre Adaboost y Gradient Boosting XGBoost.

Adaboost utiliza clasificadores débiles por defecto, como los árboles de decisión, mientras que XGBoost utiliza árboles más profundos.

- Supongamos que tenemos un conjunto de datos con pacientes sanos y pacientes enfermos con una enfermedad rara y se tienen pocos datos de pacientes enfermos. ¿Qué tipo de algoritmo se comportaría mejor, Bagging o Boosting?

Bagging sería más adecuado dado que tenemos un conjunto de datos desequilibrado con una clase minoritaria, Boosting ayuda a reducir el sesgo y mejora la generalización.

- Si para el problema anterior utilizamos árboles de decisión poco profundos, ¿considerarías utilizar Boosting en lugar de Bagging?

Los árboles de decisión son un ejemplo de modelo con bajo sesgo y alta varianza, con lo cual podría ser mejor utilizar Bagging para disminuir la varianza.

- Diseña un experimento que compare el rendimiento de Bagging y Boosting para un mismo conjunto de datos. ¿Qué métricas utilizarías para su evaluación?

```
from sklearn.datasets import load_iris
```

```
from sklearn.model_selection import train_test_split

from sklearn.ensemble import BaggingClassifier, AdaBoostClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score

# Cargar el conjunto de datos Iris

iris = load_iris()

X, y = iris.data, iris.target

# Dividir el conjunto de datos en entrenamiento y prueba

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Bagging con árbol de decisión como modelo base

bagging_classifier =
BaggingClassifier(base_estimator=DecisionTreeClassifier(),

                                        n_estimators=50, random_state=42)

# Entrenar el modelo Bagging

bagging_classifier.fit(X_train, y_train)



# Predicciones con el modelo Bagging

y_pred_bagging = bagging_classifier.predict(X_test)

# Evaluar el rendimiento del modelo Bagging

accuracy_bagging = accuracy_score(y_test, y_pred_bagging)
```

```
print(f'Accuracy con Bagging: {accuracy_bagging}')

# Boosting con AdaBoost

adaboost_classifier =
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1),

                                          n_estimators=50, random_state=42)



# Entrenar el modelo AdaBoost

adaboost_classifier.fit(X_train, y_train)

# Predicciones con el modelo AdaBoost

y_pred_adaboost = adaboost_classifier.predict(X_test)



# Evaluar el rendimiento del modelo AdaBoost

accuracy_adaboost = accuracy_score(y_test, y_pred_adaboost)

print(f'Accuracy con AdaBoost: {accuracy_adaboost}')
```

Las métricas que utilizaremos dependen de si es un problema de regresión o de clasificación. Regresión: MAE, MSE, RMSE, R2…

Clasificación: sensibilidad, especificidad, exactitud, precisión y F1.

9.8. Referencias bibliográficas

Breiman, L. (1998). Bagging predictors. *Machine Learning, 24*(2), 123-140.

Breiman, L. (1999). Pasting Small votes for classification in large databases and online. *Machine Learning, 36*(1), 85-103.

Brownlee, J. (2021). *Extreme Gradient Boosting (XGBoost) Ensemble in Python*. Machine Learning Mastery. https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python/

Freund, Y. y Schapire, R. (1997). A decision-theoretic generalization of online learning and an application to Boosting. *Journal of Computer and System Sciences, 55*(1), 119-139.

Kam, T. (1998). The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, 20*(8), 832-844.

Kearns, M. y Valiant, L. G. (1989). Crytographic limitations on learning Boolean formulae and finite automata. En D. S. Johnson (ed.), *Proceedings of the twenty-first annual ACM symposium on Theory of computing (STOC '89)* (pp. 433–444). Association for Computing Machinery. https://doi.org/10.1145/73007.73049

Leduc, O. (2019). *Boosting and AdaBoost and Gradient Boosting*. Data Action Lab. https://www.data-action-lab.com/2019/07/31/boosting-with-adaboost-and-gradient-boosting/

Louppe, G. y Geurts, P. (2012). Ensembles on Random Patches. *Machine Learning and Knowledge Discovery in Databases*, Volume part I, 346-361.

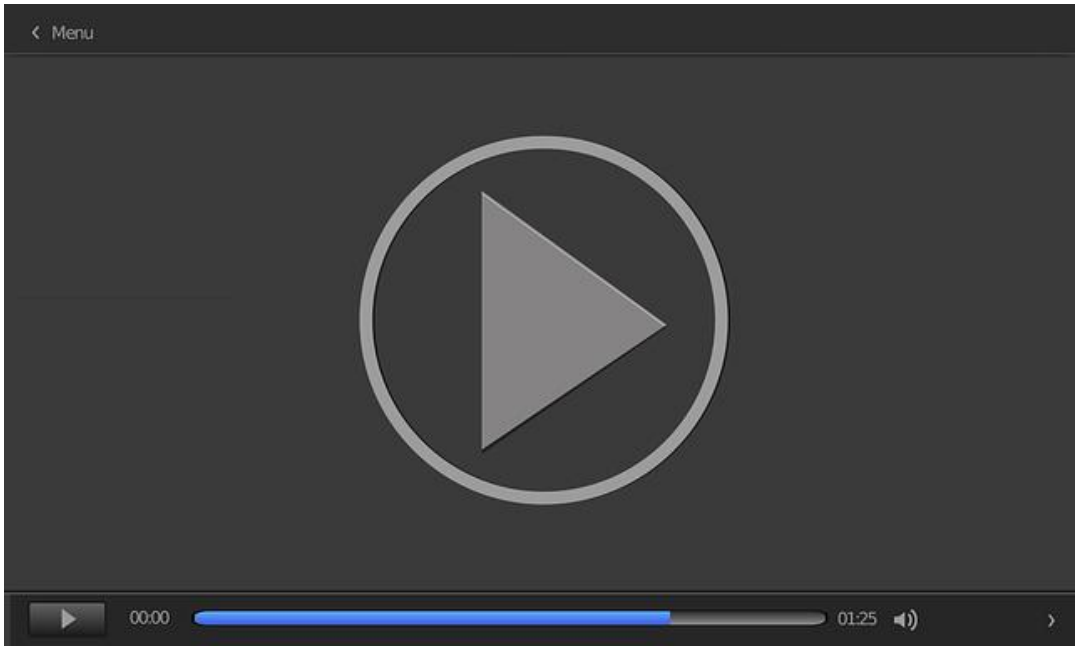
Ideas clave

Schapire, R. y Singer, Y. (1996). Improved Boosting Algorithms Using Confidence-rated Predictions. COLT (Conference on Learning Theory) de 1996. https://sci2s.ugr.es/keel/pdf/algorithm/articulo/1999-ML-Improved%20boosting%20algorithms%20using%20confidence-rated%20predictions%20(Schapire%20y%20Singer).pdf

Zhang, Z. (2019). *Boosting Algorithms Explained. Theory, Implementation, and Visualization*. Medium. https://towardsdatascience.com/Boosting-algorithms-explained-d38f56ef3f30

Ensamblaje de modelos

Sevilla R: Grupo de usuarios de R de Sevilla. (2022, diciembre 7). *Ensamblaje de modelos (stacking, Boosting y Bagging) con R* [Vídeo]. YouTube. https://www.youtube.com/watch?v=SZFbrYFYfo4



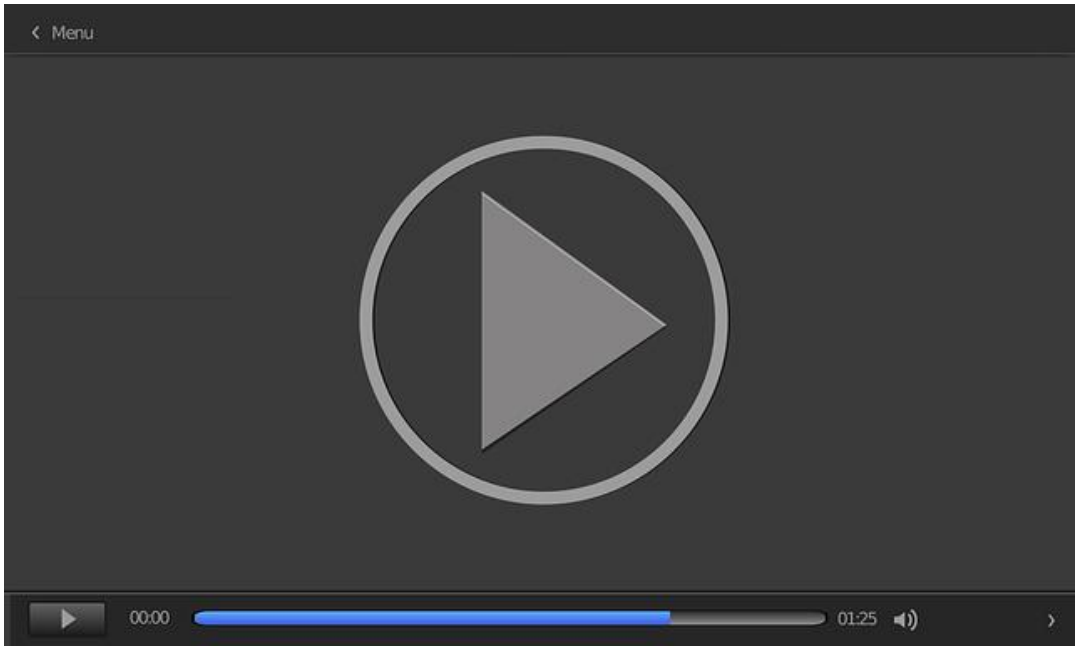
Accede al vídeo:

https://www.youtube.com/embed/SZFbrYFYfo4

Seminario sobre cómo funcionan las técnicas de ensamblaje de modelos y su aplicación para predecir la huella ecológica de los países.

CatBoost

StatQuest with Josh Starmer. (2023, febrero 27). *CatBoost Parte 1: codificación target ordenada* [Vídeo]. YouTube. https://www.youtube.com/watch?v=KXOTSkPL2X4



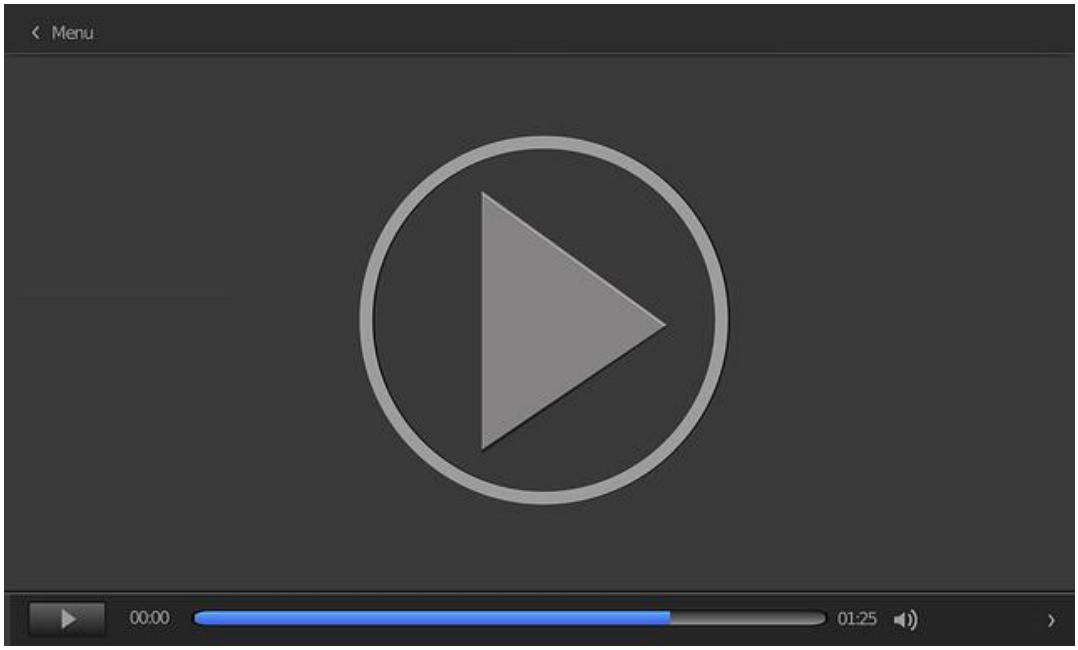
Accede al vídeo:

https://www.youtube.com/embed/KXOTSkPL2X4

Librería open source que provee un framework gradient Boosting. Su objetivo principal es evitar la fuga de datos.

Previsión series temporales con XGBoost

Python España. (2022, octubre 22). *Juan Carlos y Jaume - Regresión al Futuro: Previsión de Series Temporales con XGBoost* [Vídeo]. YouTube. https://www.youtube.com/watch?v=QZoAaauw6fo



Accede al vídeo:

https://www.youtube.com/embed/QZoAaauw6fo

XGBoost implementa eficientemente el gradiente para problemas de clasificación y regresión; es rápido y eficiente en modelos de predicción de series temporales. En este vídeo se muestran las claves para crear un modelo con estas características.

1. ¿Cuáles son los pasos que sigue el método Bagging?

   A. Genera múltiples conjuntos de entrenamiento mediante muestreo con reemplazo, entrena en paralelo tantos clasificadores base como conjuntos de datos generados existan, combina las predicciones de cada clasificador base para obtener la predicción final.

   B. Genera múltiples conjuntos de entrenamiento mediante muestreo con reemplazo, entrena secuencialmente el clasificador base con cada conjunto de datos de entrenamiento generado (hasta no terminar un entrenamiento no puede continuar con el siguiente), combina las predicciones de cada clasificador base a través del voto mayoritario.

   C. Asigna pesos a cada muestra del conjunto de entrenamiento, inicialmente todos tienen el mismo peso. Entrena el modelo débil utilizando el conjunto de datos entrenamiento ponderado con los pesos actuales. Calcula el error del modelo débil. Asigna pesos al modelo débil. Actualiza los pesos de las muestras y repite desde el entrenamiento hasta la actualización de pesos hasta alcanzar un criterio de parada. Combina los modelos débiles para formar un modelo fuerte, que se utiliza para futuras predicciones.

   D. Genera múltiples conjuntos de entrenamiento mediante muestreo con reemplazo. Entrena el modelo débil utilizando el conjunto de datos entrenamiento ponderado con los pesos actuales. Asigna pesos al modelo débil. Actualiza los pesos de las muestras y repite desde el entrenamiento hasta la actualización de pesos hasta alcanzar un criterio de parada. Combina los modelos débiles para formar un modelo fuerte, que se utiliza para futuras predicciones.

2. ¿Cuáles son los pasos que sigue el método Boosting?

A. Genera múltiples conjuntos de entrenamiento mediante muestreo con reemplazo, entrena en paralelo tantos clasificadores base como conjuntos de datos generados existan, combina las predicciones de cada clasificador base para obtener la predicción final.

B. Genera múltiples conjuntos de entrenamiento mediante muestreo con reemplazo, entrena secuencialmente el clasificador base con cada conjunto de datos de entrenamiento generado (hasta no terminar un entrenamiento no puede continuar con el siguiente), combina las predicciones de cada clasificador base a través del voto mayoritario.

C. Asigna pesos a cada muestra del conjunto de entrenamiento, inicialmente todos tienen el mismo peso. Entrena el modelo débil utilizando el conjunto de datos entrenamiento ponderado con los pesos actuales. Calcula el error del modelo débil. Asigna pesos al modelo débil. Actualiza los pesos de las muestras y repite los pasos de entrenamiento, cálculo de error, asignación de pesos y actualización de pesos hasta alcanzar un criterio de parada. Combina los modelos débiles para formar un modelo fuerte, que se utiliza para futuras predicciones.

D. Genera múltiples conjuntos de entrenamiento mediante muestreo con reemplazo. Entrena el modelo débil utilizando el conjunto de datos entrenamiento ponderado con los pesos actuales. Asigna pesos al modelo débil. Actualiza los pesos de las muestras y repite desde el entrenamiento hasta la actualización de pesos hasta alcanzar un criterio de parada. Combina los modelos débiles para formar un modelo fuerte, que se utiliza para futuras predicciones.

3. ¿Cuáles de los siguientes son algoritmos de tipo Boosting?

A. TotalBoost.

B. OptimalBoosting.

C. AdaBoost.

D. ClatsBoost.

4. El algoritmo original de Boosting:

A. Fue propuesto por Freund y Schapire en 1996.

B. Fue propuesto por Freund y Shapire en 1977.

C. Fue propuesto por Schapire y Singer en 1996.

D. Ninguna de las anteriores es correcta.

5. Son características generales del método Stacking:

A. Reduce bias o sesgo, su aprendizaje base es homogéneo, su entrenamiento base es secuencial, y su método de agregación es peso promedio.

B. Mejora exactitud, su aprendizaje base es heterogéneo, su entrenamiento base es secuencial, y su método de agregación es promedio ponderado.

C. Reduce varianza, es homogéneo, el entrenamiento es en paralelo, y la predicción está dada por el voto mayoritario o promedio.

D. Ninguna de las anteriores es correcta.

6. Son características generales del método Boosting:

A. Reduce bias o sesgo, su aprendizaje base es homogéneo, su entrenamiento base es secuencial, y su método de agregación es peso promedio.

B. Mejora exactitud, su aprendizaje base es heterogéneo, su entrenamiento base es secuencial, y su método de agregación es peso promedio.

C. Reduce varianza, es homogéneo, el entrenamiento es en paralelo, y la predicción está dada por el voto mayoritario o promedio.

D. Ninguna de las anteriores es correcta.

7. Son características generales del método Bagging:

A. Reduce bias o sesgo, su aprendizaje base es homogéneo, su entrenamiento base es secuencial, y su método de agregación es peso promedio.

B. Mejora exactitud, su aprendizaje base es heterogéneo, su entrenamiento base es secuencial, y su método de agregación es peso promedio.

C. Reduce varianza, es homogéneo, el entrenamiento es en paralelo, y la predicción está dada por el voto mayoritario o promedio.

D. Ninguna de las anteriores es correcta.

8. ¿Cuál de las siguientes afirmaciones sobre el algoritmo XGBoost es correcta?

A. XGBoost es un algoritmo de clustering ampliamente utilizado en el campo del aprendizaje no supervisado.

B. XGBoost es una técnica de reducción de dimensionalidad que busca proyectar los datos en un espacio de menor dimensión mientras conserva la estructura de la información original.

C. XGBoost es una variante del algoritmo de regresión lineal que se utiliza para predecir valores numéricos continuos.

D. XGBoost es un algoritmo de aprendizaje supervisado basado en árboles que se destaca por su eficiencia y alto rendimiento en competiciones de ciencia de datos.

9. El muestreo Boostrap es una técnica estadística utilizada para:

A. Reducir el sesgo en la estimación de un modelo mediante la agregación de múltiples clasificadores débiles.

B. Estimar la varianza de un modelo mediante la generación de múltiples conjuntos de entrenamiento a partir de reemplazos aleatorios del conjunto de datos original.

C. Seleccionar características importantes de un conjunto de datos para mejorar el rendimiento del modelo.

D. Reducir el sobreajuste en modelos de aprendizaje automático al eliminar datos atípicos.

10. El Out-of-Bag (OOB) error es una métrica utilizada en el método de Bagging que permite:

A. Evaluar el rendimiento de un modelo en el conjunto de datos de entrenamiento, utilizando la técnica de validación cruzada.

B. Medir la varianza del modelo mediante la generación de múltiples conjuntos de entrenamiento a partir de reemplazos aleatorios del conjunto de datos original.

C. Estimar la precisión de un modelo en el conjunto de prueba, utilizando únicamente las muestras que no se utilizaron en el proceso de entrenamiento.

D. Reducir el sesgo en la estimación de un modelo mediante la agregación de múltiples clasificadores débiles.