DIPLOMATURA EN MACHINE LEARNING CON PYTHON

Random Forest

Random Forest

Descripción conceptual del método

Paquete sklearn.ensemble.

- RandomForestClassifier
- RandomForestRegressor

Ejemplo de aplicación

Comparación con otras técnicas

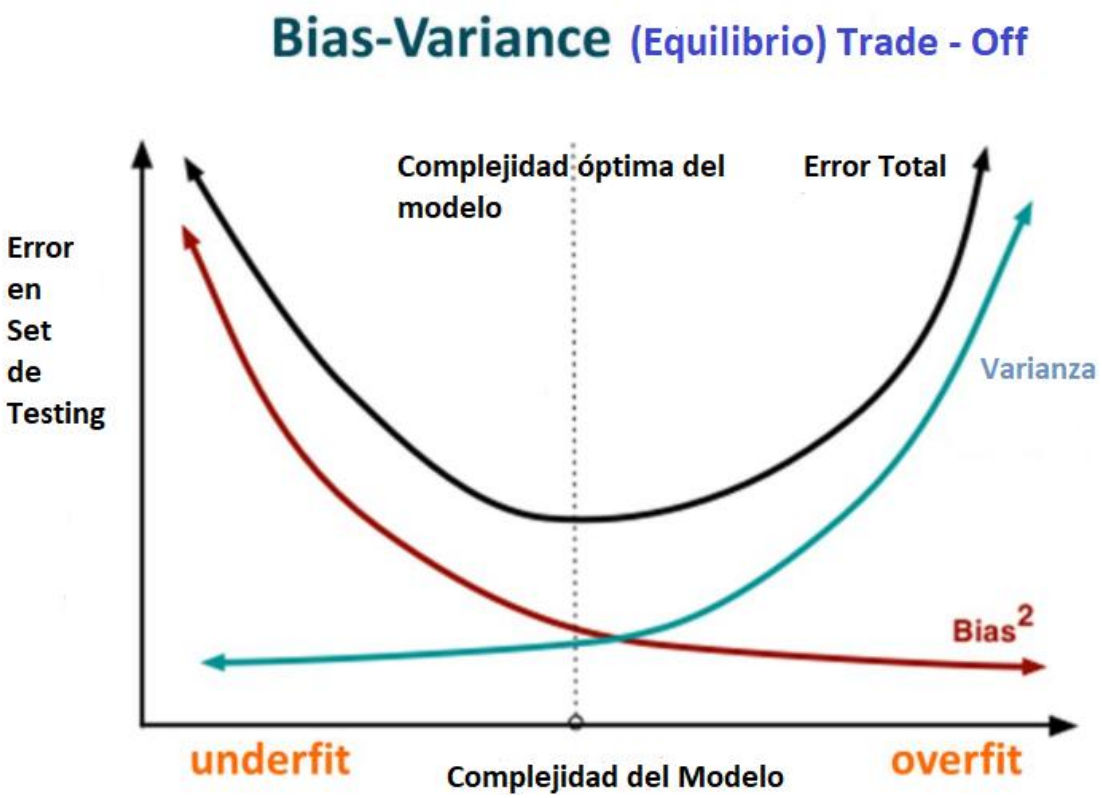
Un poco de contenido teórico:

Uno de los problemas que surge en la creación de un árbol de decisión es que si le damos la profundidad suficiente, el árbol tiende a "memorizar" las soluciones en vez de generalizar el aprendizaje.
Este error cae dentro del concepto de overfitting (sobreajuste). La solución para evitar esto es la de crear muchos árboles y que trabajen en conjunto.
El bosque aleatorio se aprovecha de esto al permitir que cada árbol individual muestre al azar del conjunto de datos **con reemplazo**, lo que resulta en árboles diferentes. Este proceso se conoce como **boostrapping**, lo veremos más adelante.-

Random Forest es un algoritmo de **aprendizaje supervisado** y cae dentro de método del tipo ensamblador. El término ensamblador significa grupo. Los métodos tipo ensamblador están formados de un **grupo de modelos predictivos** que permiten alcanzar una mejor precisión y estabilidad del modelo. Estos proveen una mejora significativa a los modelos de árboles de decisión.

¿Por qué surgen los ensambladores de árboles?

- Así como todos los modelos, un árbol de decisión también sufre de los problemas de sesgo y varianza. Es decir, 'cuánto en promedio son los valores predichos diferentes de los valores reales' (sesgo) y 'cuan diferentes serán las predicciones de un modelo en un mismo punto si muestras diferentes se tomaran de la misma población' (varianza).
- Al construir un árbol pequeño se obtendrá un modelo con baja varianza y alto sesgo. Normalmente, al incrementar la complejidad del modelo, se verá una reducción en el error de predicción debido a un sesgo más bajo en el modelo. En un punto el modelo será muy complejo y se producirá un sobre-ajuste del modelo el cual empezará a sufrir de varianza alta.
- El modelo óptimo debería mantener un balance entre estos dos tipos de errores. A esto se le conoce como "trade-off" (equilibrio) entre errores de sesgo y varianza. El uso de ensambladores es una forma de aplicar este "trade-off".



Ensambladores comunes: Bagging, Boosting and Stacking. Random Forest es del primer tipo

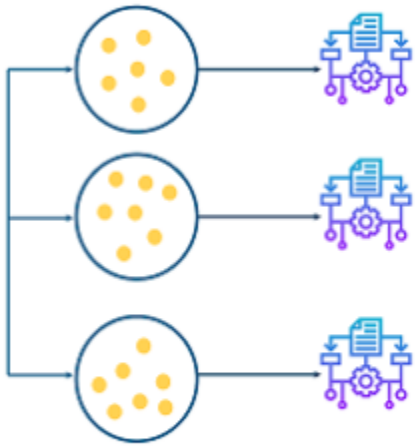
Aunque hay diversas formas de ensamblar o unir algoritmos débiles para formar otros, las más usadas y populares son el **bagging** y el **boosting** (aunque existen otras como el stacking). Cada tipo de algoritmo tiene unas ventajas y unos inconvenientes y pueden ser usados convenientemente según nuestra problemática.

Random Forest es un ejemplo de algoritmo que hace uso de bagging que combina árboles de decisión para formar el Random Forest.

El principal objetivo intrínseco de los algoritmos de bagging es el de la reducción de la varianza

Los métodos de bagging son métodos donde los algoritmos simples son usados en paralelo. El principal objetivo de los métodos en paralelo es el de **aprovecharse de la independencia que hay entre los algoritmos simples,** ya que el error se puede reducir bastante al promediar las salidas de los modelos simples. Es como si, queriendo resolver

un problema entre varias personas independientes unas de otras, damos por bueno lo que eligiese la mayoría de las personas.



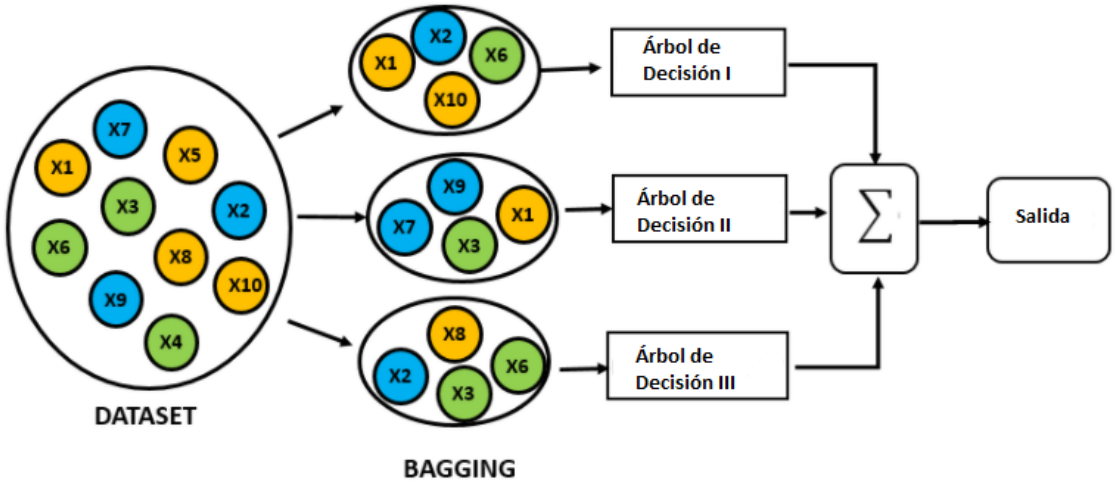
Métodos paralelos de bagging

Bagging proviene del concepto de **agregación de bootstrap**. Un forma de reducir la varianza de las estimaciones es promediando estimaciones de distintos modelos o algoritmos.

Para obtener la agregación de las salidas de cada modelo simple e independiente, bagging puede usar la *votación* para los métodos de clasificación y el *promedio* para los métodos de regresión.

Si lo imaginamos en un proceso podríamos verlo de la siguiente manera. Dado un dataset inicial se van a realizar los siguientes pasos:

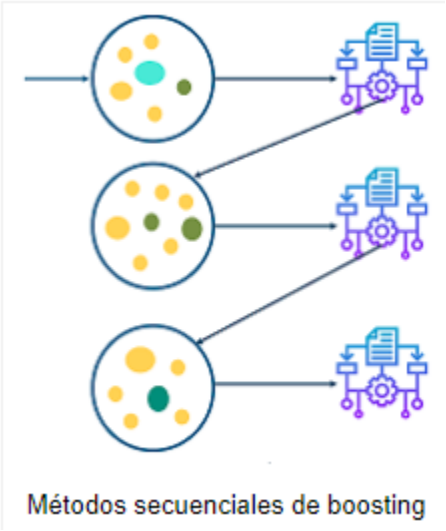
- Crear múltiples subconjuntos de datos
- Construir múltiples modelos
- Combinar los modelos.



Por otro lado, dentro de los algoritmos ensamblados tenemos los algoritmos de boosting. Algunos ejemplos de estos algoritmos son el XGBoost o el AdaBoost.

El principal objetivo de los algoritmos de boosting es el de la reducción del sesgo

En los algoritmos de boosting, los **modelos simples son utilizados secuencialmente**, es decir, cada modelo simple va delante o detrás de otro modelo simple. El principal objetivo de los métodos secuenciales es el de aprovecharse de la dependencia entre los modelos simples.
El rendimiento general puede ser mejorado haciendo que un modelo simple posterior le dé **más importancia a los errores cometidos por un modelo simple previo**. Poniendo un ejemplo, es como si nosotros al resolver un problema aprovechásemos nuestro conocimiento de los errores de otros para mejorar en algo intentando no cometerlos nosotros.



Métodos secuenciales de boosting

Las predicciones de cada modelo simple se combinan por medio de una votación (para problemas de clasificación) o por medio de una suma ponderada (para problemas de regresión) para producir la predicción final. ***La diferencia con el bagging es que en el boosting los algoritmos no se entrenan independientemente, sino que se ponderan según los errores de los anteriores.***

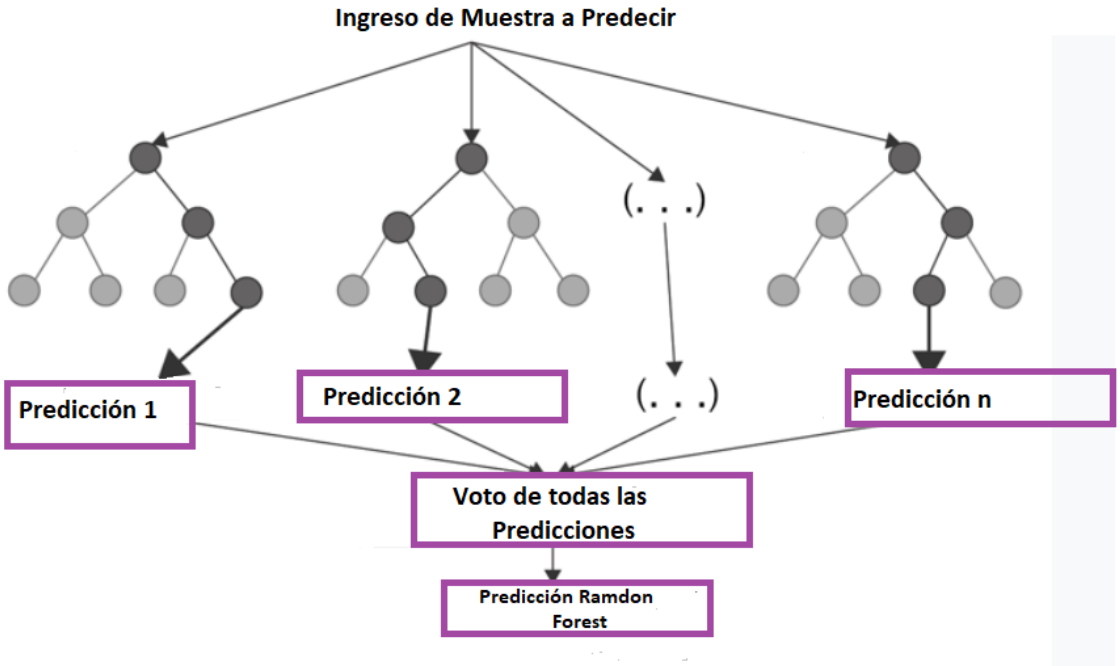
¿Cómo funciona la solución Random Forest?

Random Forest, como su nombre lo indica, consta de una gran cantidad de árboles de decisión individuales que operan como un conjunto.

Cada árbol individual en el bosque aleatorio arroja una predicción de clase y la clase con más votos se convierte en la predicción del modelo, como lo vemos en la figura a continuación.-

Cada árbol crece hasta su máxima extensión posible y **NO hay proceso de poda**.

Nuevas instancias se predicen a partir de la agregación de las predicciones de los x árboles (i.e., mayoría de votos para clasificación, promedio para regresión)



El concepto fundamental detrás del bosque aleatorio es simple pero poderoso: la sabiduría de las multitudes.

En términos de ciencia de datos, la razón por la que el modelo de bosque aleatorio funciona tan bien es:

- Un gran número de modelos (árboles) relativamente no correlacionados que operan como un comité superarán a cualquiera de los modelos constituyentes individuales.

- La baja correlación entre modelos es la clave. Al igual que las inversiones con correlaciones bajas (como acciones y bonos) se unen para formar una cartera que es mayor que la suma de sus partes, los modelos no correlacionados pueden producir predicciones de conjunto que son más precisas que cualquiera de las predicciones individuales.

La razón de este maravilloso efecto es que los árboles se protegen unos a otros de sus errores individuales (siempre que no se equivoquen constantemente en la misma dirección).

Si bien algunos árboles pueden estar equivocados, muchos otros árboles serán correctos, por lo que, como grupo, los árboles pueden moverse en la dirección correcta.

Entonces, los requisitos previos para que el bosque aleatorio funcione bien son:

- Es necesario que haya alguna señal real en nuestras funciones para que los modelos creados con esas funciones funcionen mejor que la adivinación aleatoria.

- Las predicciones (y por lo tanto los errores) realizados por los árboles individuales deben tener bajas correlaciones entre sí.

Out of bag samples y out of bag error

- El proceso de muestreo de los datos con reemplazo se denomina **bootstrap**.
- Un tercio de los datos no se usan para el entrenamiento y pueden ser usados para test.
- Este conjunto se denomina out of bag (OOB) samples.

- El error estimado en estos out of bag samples se conoce como out of bag error (OOB error)
- Usar este conjunto de test (OOB) es tan preciso como si se usara un conjunto de test del mismo tamaño que el de entrenamiento.
- Sería posible no usar un conjunto de test adicional.

¿Qué es el proceso de boostrap?

El bootstrap es un mecanismo propio de la estadística y la econometría que se centra en el re muestreo de datos dentro de una muestra aleatoria o al azar. Su principal uso es hallar una aproximación a la distribución de la variable analizada.

La principal utilidad del empleo del bootstrap es reducir el sesgo dentro de análisis o, en otras palabras, aproximar la **varianza** gracias a la realización de re muestreos aleatorios de la muestra inicial y no de la población. De este modo, se hace más sencilla la construcción de modelos estadísticos mediante la creación de intervalos de confianza y contrastes de hipótesis.

Aunque pueda parecer una práctica muy compleja a priori, el procedimiento en que se basa el bootstrapping es simplemente la creación de un gran número de muestras reposicionando los datos tomando como referencia una muestra poblacional inicial

Dicha técnica resulta especialmente útil en aquellas situaciones en las que las muestras con las que se cuenta son pequeñas o, como se dijo antes, si la distribución es muy sesgada. En ese sentido, ayudan a la resolución de multitud de problemas de probabilidad y estadística aplicada.

Una de las principales características de esta práctica es que supone un re muestreo posterior para poder obtener expresiones cerradas y solucionar la complejidad matemática de estas operaciones. Con el desarrollo de los ordenadores y herramientas tecnológicas en los últimos años, se ha hecho más fácil poder contar con el uso del bootstrapping para re muestreos complejos.

La técnica del re muestreo permite ir más allá a la hora de estudiar muestras de datos de una determinada población. En otras palabras, permite hacer o crear nuevos supuestos sustituyendo valores adicionales de la muestra.

Un aspecto considerado positivo del re muestreo mediante bootstrap es que ha simplificado los métodos estadísticos, en el sentido de que ha sustituido la construcción de modelos matemáticos clásicos y de gran complejidad por el cálculo mediante software específicos, lo que ha mejorado su aplicabilidad o acceso a otros campos o estudios.

Siguiendo esa línea, se considera habitualmente que este mecanismo es mucho más abierto o accesible en comparación con los modelos y supuestos clásicos, lo que le convierte en una herramienta útil para gran número de problemas matemáticos.

Ventajas y Desventajas de Random Forest:

Ventajas:

- Es uno de los algoritmos de aprendizaje más precisos disponibles. Para muchos conjuntos de datos, produce un clasificador de alta precisión.
- Funciona de manera eficiente en grandes bases de datos.
- Puede manejar miles de variables de entrada sin eliminarlas.
- Proporciona estimaciones de las variables que son importantes en la clasificación.
- Genera una estimación interna no sesgada del error de generalización a medida que avanza la construcción del bosque.
- Tiene un método eficaz para estimar los datos faltantes y mantiene la precisión cuando falta una gran proporción de los datos.
- Puede resolver problemas de clasificación y regresión

Desventajas:

- Se ha observado que los bosques aleatorios sobre ajustan para algunos conjuntos de datos con tareas de clasificación / regresión ruidosas.
- Para los datos que incluyen variables categóricas con diferente número de niveles, los bosques aleatorios están sesgados a favor de aquellos atributos con más niveles. Por lo tanto, los puntajes de importancia variable del bosque aleatorio no son confiables para este tipo de datos.

Con respecto a la terminología de cada subárbol se aplican los mismos conceptos que los árboles de decisión.

Terminología de un árbol de decisión

1. **Nodo raíz (nodo de decisión superior):** Representa a toda la población o muestra y esto se divide en dos o más conjuntos homogéneos.

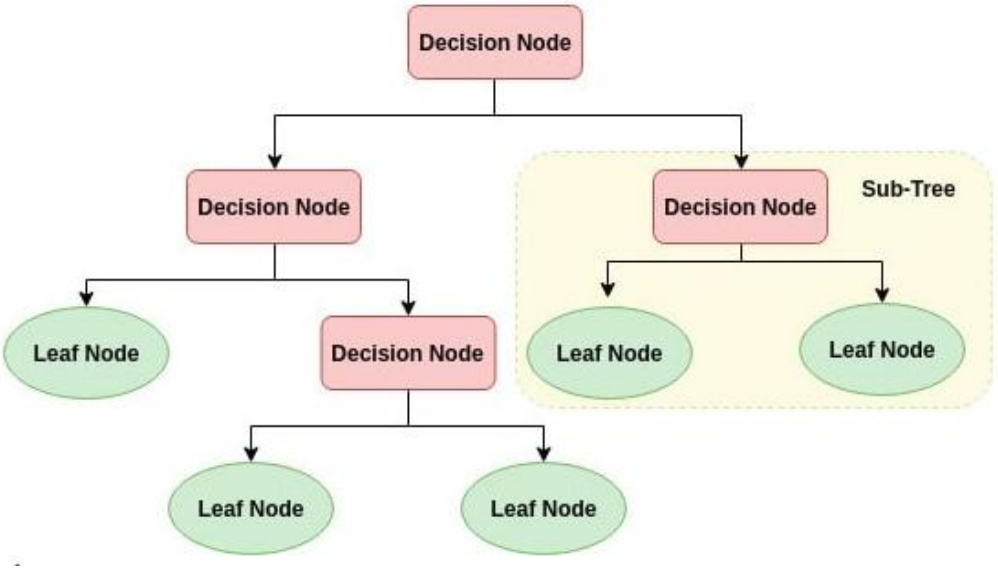
2. **División:** Es un proceso de división de un nodo en dos o más subnodos. Nodo de decisión: Cuando un subnodo se divide en subnodos adicionales, se llama nodo de decisión.

3. **Nodo de hoja / terminal:** Los nodos sin hijos (sin división adicional) se llaman Hoja o nodo terminal.

4. **Poda:** Cuando reducimos el tamaño de los árboles de decisión eliminando nodos (opuesto a la división), el proceso se llama poda.

5. **Rama / Subárbol:** Una subsección del árbol de decisión se denomina rama o subárbol.

6. **Nodo padre e hijo:** Un nodo, que se divide en subnodos se denomina nodo principal de subnodos, mientras que los subnodos son hijos de un nodo principal.



Como utilizo Random Forest en Python – Hiperparámetros importantes

class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*)

class sklearn.ensemble.**RandomForestRegressor**(*n_estimators=100, *, criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None)*

Hiperparámetros importantes

Los siguientes hiperparámetros nos ayudarán a optimizar los resultados del Random Forest.-

- **n_estimators**: será la cantidad de árboles que vamos a generar.

- **max_features**: la manera de seleccionar la cantidad máxima de features para cada árbol.
- **min_sample_leaf**: número mínimo de elementos en las hojas para permitir un nuevo split (división) del nodo.
- **oob_score**: es un método que emula el cross-validation en árboles y permite mejorar la precisión y evitar overfitting.
- **boostrap**: para utilizar diversos tamaños de muestras para entrenar. Si se pone en falso, utilizará siempre el dataset completo.
- **n_jobs**: si tienes multiples cores en tu CPU, se puede indicar cuantos puede usar el modelo al entrenar para acelerar el entrenamiento.

Áreas de Aplicación:

El algoritmo de bosque aleatorio se utiliza en muchos campos diferentes, como la banca, el mercado de valores, la medicina y el comercio electrónico.

En finanzas, por ejemplo, se utiliza para detectar clientes con más probabilidades de pagar su deuda a tiempo o utilizar los servicios de un banco con más frecuencia. En este dominio también se utiliza para detectar a los estafadores que intentan estafar al banco. En el comercio, el algoritmo se puede utilizar para determinar el comportamiento futuro de una acción.

En el ámbito de la salud, se utiliza para identificar la combinación correcta de componentes en medicina y para analizar el historial médico de un paciente para identificar enfermedades.

El bosque aleatorio se utiliza en el comercio electrónico para determinar si a un cliente realmente le gustará el producto o no.