

Técnicas de Aprendizaje Automático

Tema 10. Aprendizaje supervisado. Regresión y clasificación con Random Forest

Índice

Esquema

Ideas clave

10.1. Introducción

10.2. Explotando la diversidad: Bagging y selección de variables

10.3. Interpretación de out-of-the-bag error

10.4. Evolución del número de árboles en función del número de variables por árbol e importancia del número de variables por árbol e importancia de variables

10.5. Cuaderno de ejercicios

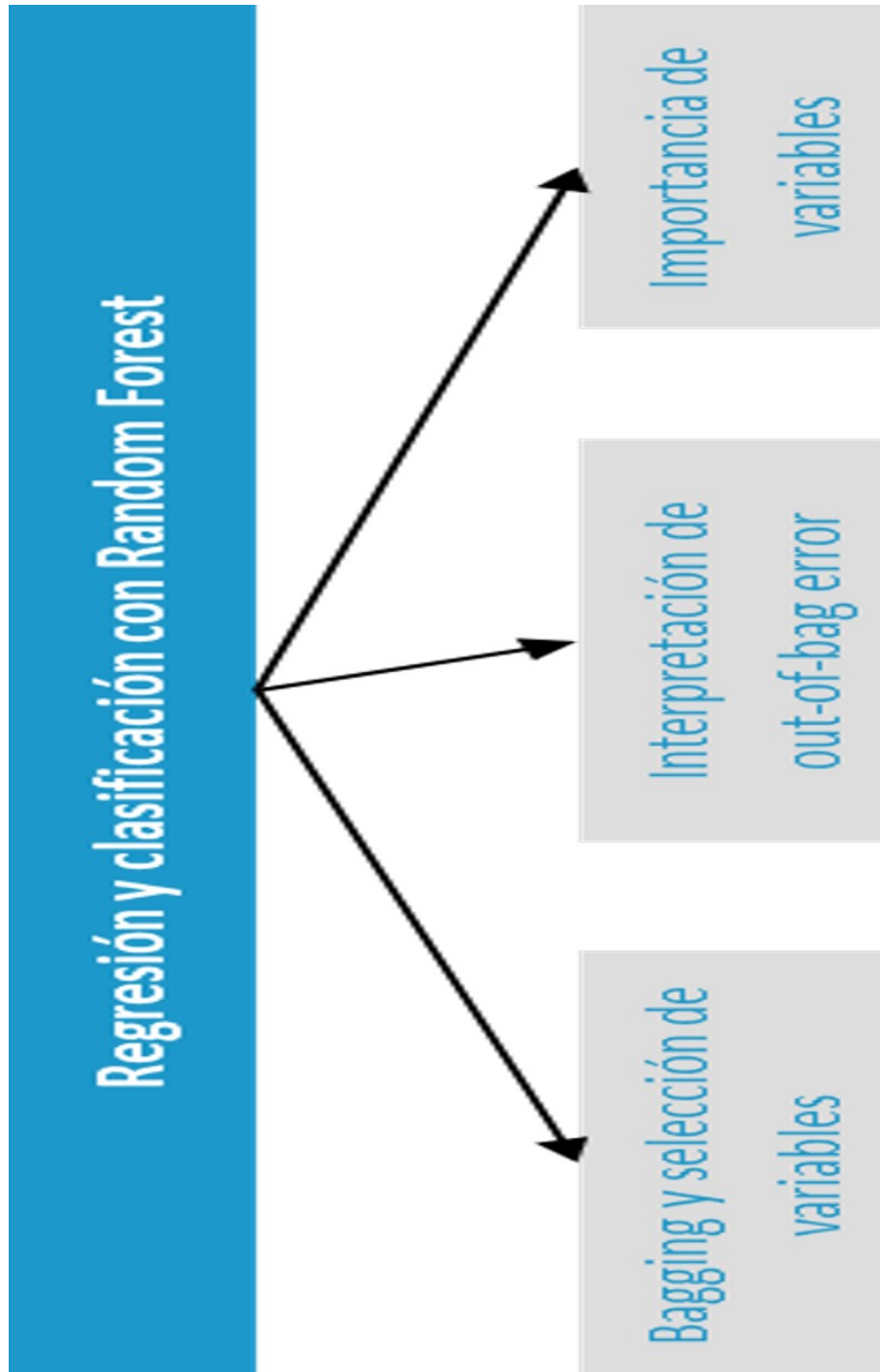
10.6. Referencias bibliográficas

A fondo

Ensamblaje de modelos

Demystifying the random forest

Test



10.1. Introducción

Random Forest es un método que combina la simplicidad de los árboles de decisión, ensamblándolos en un solo modelo más robusto. Fue inventado por Leo Breiman y Adele Cutler en el año 2001 (Breiman, 2001).

En este tema describiremos cómo los modelos de Random Forest explotan la diversidad y son capaces de generalizar y predecir mejor que un árbol de decisión. Esta diversidad se consigue por medio del entrenamiento de los modelos utilizando el método Bagging y la selección de variables aleatoria.

A continuación, discutiremos la interpretación del *out-of-bag error*. Posteriormente, nos enfocaremos en determinar el número de árboles y en evaluar la importancia de las variables.

En este tema nos planteamos los siguientes objetivos:

- ▶ Explicar la teoría del funcionamiento del algoritmo de Random Forest.
- ▶ Analizar las ventajas y desventajas de Random Forest.
- ▶ Aplicar Random Forest en problemas reales.

10.2. Explotando la diversidad: Bagging y selección de variables

Uno de los inconvenientes principales de los árboles de decisión es su **baja capacidad predictiva**. Este inconveniente se puede solventar utilizando un ensamble o combinación de modelos de árboles de decisión. El modelo Random Forest es, en esencia, un ensamble de árboles de decisión. Los ensambles de árboles se pueden combinar utilizando los métodos de Bagging o Boosting. Los modelos de Random Forest se basan en la combinación de los árboles por medio de **métodos Bagging**.

En el caso de los problemas de clasificación en cada observación de test se almacena la clase predicha por cada uno de los B árboles entrenados y la clase final se obtiene por medio del voto mayoritario (problemas de clasificación) o el promedio de las predicciones (problemas de regresión).

El modelo de Random Forest combina los principios de Bagging con selección de variables aleatorias para añadir diversidad a los árboles de decisión. Una vez es generado el ensamble de árboles (Forest) el modelo utiliza el mecanismo de votación o el promedio para generar las predicciones.

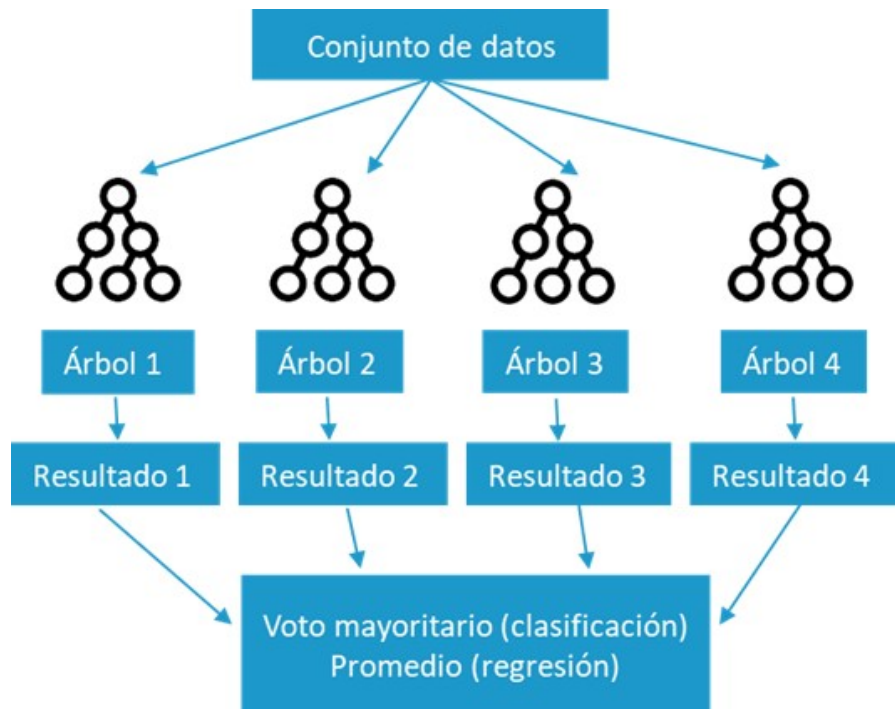


Figura 1. Random forest para clasificación y regresión. Fuente: elaboración propia.

Se trata de un modelo que combina versatilidad y potencia en un enfoque. A la hora de construir cada uno de los árboles se utiliza una porción pequeña y aleatoria de las variables de entrada disponibles; por lo general, este número se define como \sqrt{p} siendo p el **número de variables de entrada**.

Al ser un modelo que genera cada árbol con un subconjunto de los registros de entrada y una selección aleatoria de las variables, puede trabajar con conjuntos de datos bastante grandes y no se encuentra afectado por los problemas de la maldición de la dimensionalidad (*curse of dimensionality*, en inglés). *Curse of dimensionality* se refiere a los desafíos y dificultades que surgen al trabajar con conjuntos de datos de alta dimensionalidad, a medida que la dimensionalidad aumenta, surgen diferentes problemas:

- Los datos disponibles pueden estar muy dispersos y dificultan el análisis.

- ▶ Aumenta la complejidad si hay más dimensiones. El tamaño del espacio de búsqueda aumenta, dificultando la identificación de patrones y relaciones entre variables, llevando a un rendimiento deficiente.
- ▶ Reducción del rendimiento a causa del sobreajuste (*overfitting*) debido a la presencia de características irrelevantes.
- ▶ El procesamiento de altas dimensiones requiere altos recursos computacionales, de tiempo y memoria.

Además, debido a que los árboles son modelos con mucho ruido y muy inestables, se puede ver el beneficio cuando se promedian las predicciones de varios árboles. En cada uno de los cortes de los árboles se elige una serie de variables candidatas de entre todas las posibles.

Este modelo se basa en la utilización de un **gran número de árboles**. La razón de utilizar un gran número de árboles es para que cada variable, de entre todas las posibles, tenga la oportunidad de aparecer en varios modelos.

La ventaja principal de los modelos de Random Forest es que las variables de cada *split* de los árboles se obtienen gracias a la selección de un subconjunto de todas las variables de forma aleatoria. De esta forma, se consigue decorrelar los árboles generados. Además, se reduce la varianza porque el resultado se obtiene a partir del cálculo del promedio de los árboles, y se controla el sobreajuste.

Por tanto, cada árbol de decisión se construye utilizando el muestreo Bootstrap y en cada *split* se utiliza un parámetro que define el número de predictores; a partir de una selección aleatoria de m predictores de un total de p predictores. Finalmente, los distintos árboles se combinan utilizando el voto de la mayoría para clasificación y el promedio para la regresión.

En Random Forest, debido a la selección de variables aleatorias sobre el conjunto de entrada completo, el sesgo de los árboles se suele incrementar (con respecto al sesgo de un único árbol de decisión); no obstante, al promediar los árboles, la varianza se reduce en una proporción mayor que el incremento del sesgo, dando lugar a un mejor modelo. En muchos problemas el rendimiento de Random Forest es similar a los modelos Boosting; como consecuencia, este algoritmo se ha convertido en una técnica muy popular, como podemos observar en la Figura 2. En ella se aprecia el entrenamiento de varios modelos DecisionTree, RandomForest, ExtraTrees (Bosques Extremadamente Aleatorios), AdaBoost en los que se utiliza el conjunto de datos iris, y vemos que el entrenamiento para RandomForest y Adaboost es muy similar.

Clasificadores por subconjunto de características

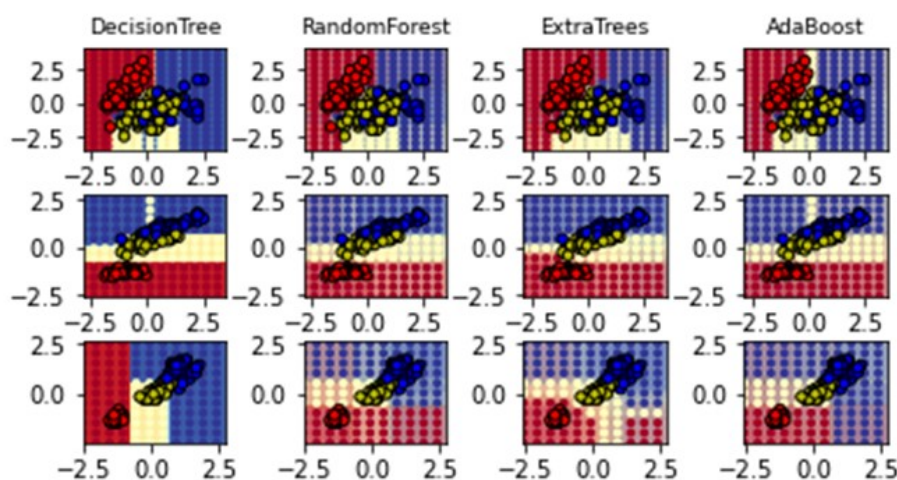


Figura 2. Comparación de modelos DecisionTree, RandomForest, ExtraTrees y AdaBoost. Fuente: Pedregosa *et al.*, 2011.

La fila 1 corresponde a *sepal length* vs *sepal width*; fila 2 a *sepal width* vs *petal length*; y la fila 3 a *petal length* vs *petal width*.

En Random Forest cada árbol crece de la siguiente manera:

- ▶ Se muestrean N instancias de los datos originales, utilizando la técnica de Bootstrap.
- ▶ Si hay p variables de entrada, se especifica un número $m < p$ tal que, en cada nodo, m variables se seleccionan aleatoriamente de las p y la mejor división en estas m se usa para dividir el nodo. El valor de m se mantiene constante durante el crecimiento del bosque.
- ▶ Cada árbol crece lo que más puede. No hay poda.

10.3. Interpretación de out-of-the-bag error

Una característica importante de los modelos Random Forest es el *out-of-bag error*. El *out-of-bag error* o error fuera de la bolsa es una forma sencilla de estimar el error del conjunto de datos de test en un modelo bagged. Se puede demostrar que de media cada árbol construido con un modelo bagged utiliza $2/3$ de las observaciones del conjunto de entrenamiento. El $1/3$ restante de las observaciones de entrenamiento no se utiliza para generar el árbol bagged y se denomina *out-of-bag*.

Si el número de árboles es grande, esta estimación es equivalente a un modelo de tipo *leave-one-out* de validación cruzada. *Leave-one-out* es la forma más rigurosa y extrema de evaluar un modelo de aprendizaje automático.

Un modelo Random Forest se puede entrenar de forma secuencial, no es necesario utilizar validación cruzada, pues las muestras *out-of-bag* proporcionan una estimación similar. Para ello, es necesario observar la evolución del *out-of-bag error* y una vez que este error se haya estabilizado, el algoritmo puede parar el entrenamiento. En la Figura 3 podemos ver un ejemplo de cómo va evolucionando el error *out-of-bag* durante el entrenamiento de un modelo Random Forest con el conjunto de datos de iris. Podemos ver que el error más bajo lo encontramos ya con 20 árboles y ese mismo error estable lo encontramos con aproximadamente 65 árboles, así que podríamos tomar la decisión de parar el algoritmo en 20 o en 65.

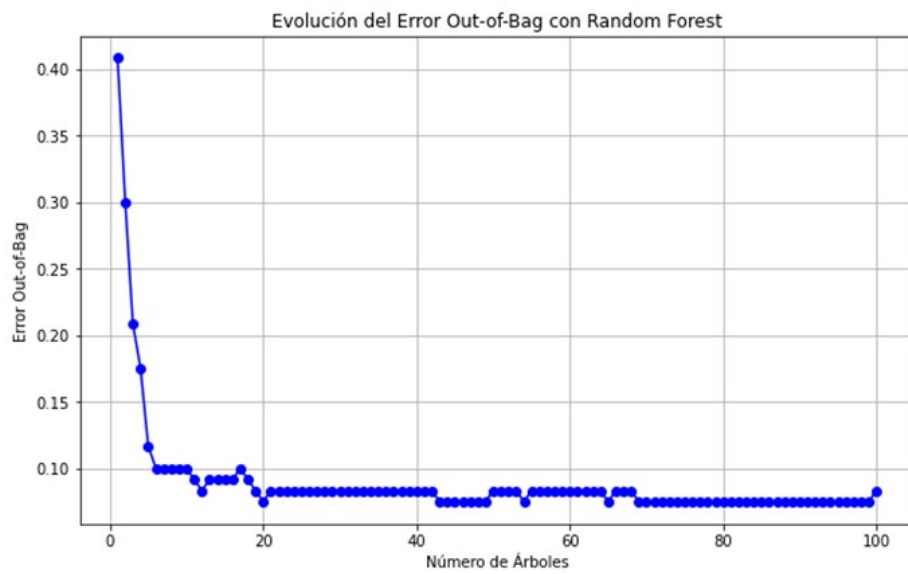


Figura 3. Evolución del *out- of-bag error*. La gráfica muestra la evolución de 100 árboles, pero se observa que 20 árboles serían suficientes. Fuente: elaboración propia.

10.4. Evolución del número de árboles en función del número de variables por árbol e importancia del número de variables por árbol e importancia de variables

Los modelos basados en Random Forest tienen dos parámetros que podemos optimizar: el **número de árboles** y el **número de variables** que se evalúan en cada iteración. Existen otros parámetros como la profundidad de los árboles, pero en la práctica no presentan muchos cambios.

En la Figura 4 se observa la evolución del error en el conjunto de test en función del número de árboles y del número de variables que se prueban en cada *split*. Se observa que el valor de \sqrt{p} proporciona un error menor que los otros valores.

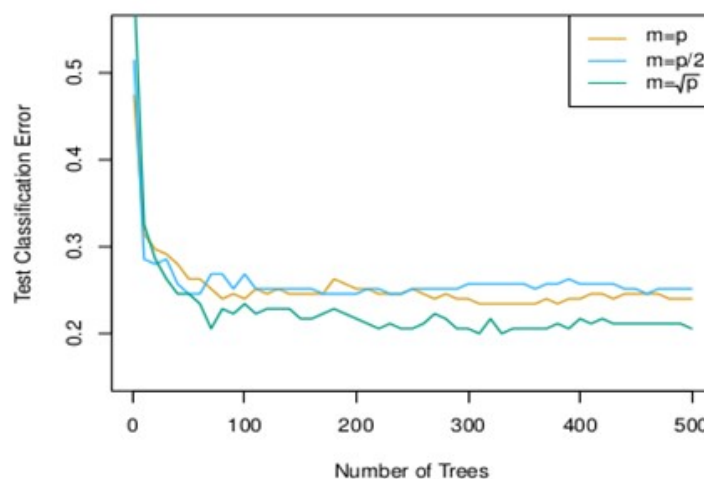


Figura 4. Tres diferentes resultados de un Random Forest para un problema de clasificación de 15 clases y que tiene 500 predictores. Fuente: James *et. al.*, 2017.

Se podría pensar que cuanto más grande sea el número de árboles, mejor. Sin embargo, por lo general, existe un punto óptimo donde, a pesar de añadir más árboles, el error no se reduce.

Una de las ventajas de los Random Forest es que pueden generar buenos resultados sin necesidad de realizar muchos ajustes manuales. Además, permiten obtener la importancia de las variables en el modelo. Bagging mejora la precisión, pero sacrifica la interpretabilidad frente a los árboles de decisión, que son algoritmos muy fáciles de interpretar, pero la precisión no es muy alta. Con Bagging la precisión es mayor, pero no es tan claro cuáles son las características más importantes. Para poder obtener las variables más importantes se calculan las medidas de importancia de las variables, como el índice Gini para la clasificación y la Suma Residual de Cuadrados (RSS, por sus siglas en inglés) para la regresión. De esta forma, podemos saber cuáles son las características más importantes.

A medida que se construyen los árboles de decisión, se puede calcular cuánto cae la función de error para una variable en cada punto de división. Para poder calcular la importancia de las variables utilizando el índice Gini se siguen los siguientes pasos:

- ▶ Cada vez que se divide un nodo en un árbol de decisión, se mide la disminución de impureza Gini para los nodos descendientes y esta es menor que la del nodo padre.
- ▶ La importancia de las variables se calcula sumando las disminuciones de impureza a lo largo de todos los nodos donde se realiza una división de esa variable.
- ▶ Esta suma se normaliza para que la importancia total de todas las variables sea igual a 1.

Ejemplo de código para ver gráficamente la importancia de las variables utilizando el algoritmo de Random Forest.

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.datasets import load_iris

import matplotlib.pyplot as plt
```

```
# Cargar el conjunto de datos Iris

iris = load_iris()

X, y = iris.data, iris.target


# Crear un modelo de Random Forest

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)


# Entrenar el modelo

rf_model.fit(X, y)


# Obtener importancias de las variables

importances = rf_model.feature_importances_


# Obtener nombres de las variables

feature_names = iris.feature_names


# Visualizar importancias de las variables

plt.barh(range(len(importances)), importances, align='center')

plt.yticks(range(len(importances)), feature_names)

plt.xlabel('Importancia Relativa')
```

```
plt.ylabel('Variables')
```

```
plt.title('Importancia de las Variables en Random Forest')
```

```
plt.show()
```

y como resultado obtenemos la Figura 5.

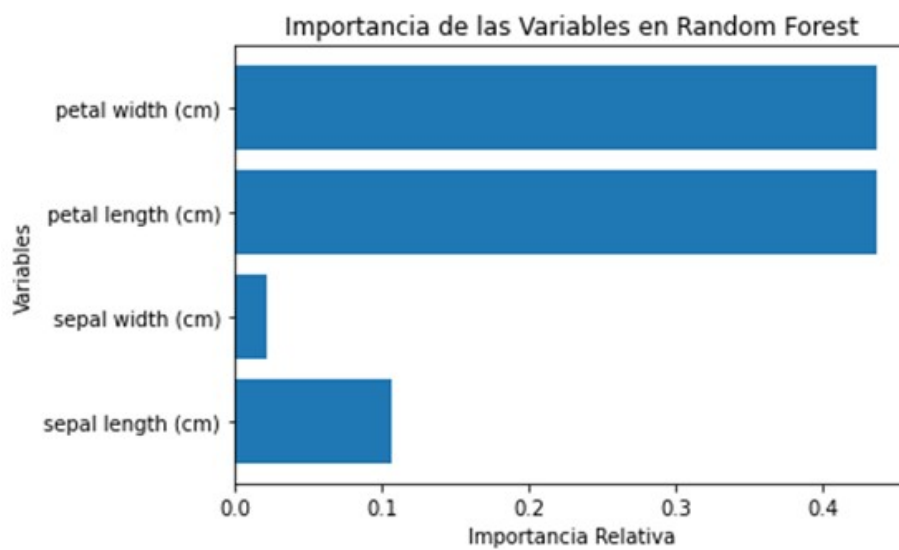


Figura 5. Importancia de variables conjunto de datos iris entrenado con Random Forest.

Fuente: elaboración propia.

Ventajas de los bosques aleatorios o Random Forest:

- ▶ Funciona muy bien con grandes cantidades de datos.
- ▶ Puede manejar un gran número de variables de entrada, sin necesidad de eliminar ninguna.
- ▶ Puede estimar la importancia de las variables dentro de la clasificación.
- ▶ Genera una estimación imparcial interna del error de generalización a medida que avanza la construcción del bosque.

- ▶ Trabaja muy bien ante la existencia de datos faltantes.
- ▶ Es capaz de equilibrar el error en conjunto de datos desbalanceados.
- ▶ Es menos propenso al sobreajuste que se tiene con un único árbol de decisión individual, gracias a la diversidad introducida.
- ▶ Funciona muy bien para problemas de clasificación y regresión.
- ▶ No requiere de un ajuste fino de hiperparámetros complicados y es menos sensible a la elección de estos.

Bosques extremadamente aleatorios (Extra Trees)

Existe una variante de Random Forest denominada Extra Trees, que comparte similitudes, pero también tienen algunas diferencias en el proceso de construcción de los árboles.

Algunas de las **características** que encontramos son:

- ▶ Es un algoritmo de ensamble que construye árboles de decisión y los combina para realizar predicciones más precisas y robustas.
- ▶ La principal diferencia con la implementación básica de Random Forest (RF) radica en la forma en la que se construyen los árboles. Mientras que en RF se elige aleatoriamente un subconjunto de características para dividir en cada nodo, Extra Trees utiliza el total del conjunto de características para cada división; además, las divisiones se eligen de manera aleatoria, sin buscar la mejor división posible.
- ▶ Al igual que los otros métodos de ensamble, Extra Trees busca reducir la varianza al combinar múltiples modelos, pero su enfoque es extremadamente aleatorio en la construcción de árboles, en muchos casos el tiempo de entrenamiento es más eficiente.

Ejemplo de código para ver gráficamente la importancia de las variables comparando Random Forest y ExtraTrees con el conjunto de datos de diabetes.

```
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier

from sklearn.datasets import load_diabetes

import matplotlib.pyplot as plt

import numpy as np

diabetes = load_diabetes()

X, y = diabetes.data, diabetes.target

# Crear un modelo de RandomForestClassifier

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Entrenar el modelo

rf_model.fit(X, y)

# Obtener importancias de las variables

importances_rf = rf_model.feature_importances_
```

```
# Crear un modelo de ExtraTreesClassifier

extra_trees_model = ExtraTreesClassifier(n_estimators=100, random_state=42)


# Entrenar el modelo

extra_trees_model.fit(X, y)


# Obtener importancias de las variables

importances_extra_trees = extra_trees_model.feature_importances_


# Obtener nombres de las variables

feature_names = diabetes.feature_names


# Visualizar importancias de las variables para RandomForest

plt.figure(figsize=(12, 6))


plt.subplot(1, 2, 1)

plt.barh(range(len(importances_rf)), importances_rf, align='center')

plt.yticks(range(len(importances_rf)), feature_names)

plt.xlabel('Importancia Relativa')

plt.ylabel('Variables')
```

```
plt.title('Importancia de las Variables en RandomForest')

# Visualizar importancias de las variables para ExtraTrees

plt.subplot(1, 2, 2)

plt.barh(range(len(importances_extra_trees)), importances_extra_trees, align='center')

plt.xticks(range(len(importances_extra_trees)), feature_names)

plt.xlabel('Importancia Relativa')

plt.ylabel('Variables')

plt.title('Importancia de las Variables en ExtraTrees')

plt.tight_layout()

plt.show()
```

Y como resultado obtenemos la Figura 6. Vemos que en términos generales la puntuación varía muy poco y la importancia de variables para este conjunto de datos es similar en los dos algoritmos.

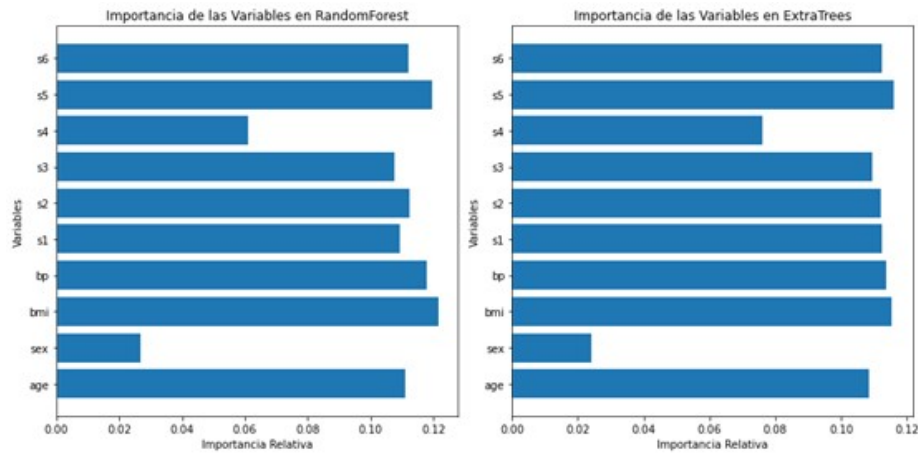


Figura 6. Importancia de variables conjunto de datos diabetes entrenado con Random Forest vs ExtraTrees. Fuente: elaboración propia.

Ahora comparemos la evolución del *out-of-bag-error* para el conjunto de datos de diabetes.

Ejemplo de código para ver gráficamente la evolución del OOB comparando Random Forest y ExtraTrees con el conjunto de datos de diabetes.

```
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier

from sklearn.datasets import load_diabetes

import matplotlib.pyplot as plt

import numpy as np

# Cargar el conjunto de datos Diabetes

diabetes = load_diabetes()

X, y = diabetes.data, diabetes.target
```

```
# Inicializar listas para almacenar el número de árboles y los errores OOB para
RandomForest y ExtraTrees

n_trees_range = range(1, 101) # Ajusta según sea necesario

oob_errors_rf = []

oob_errors_extra_trees = []


# Entrenar modelos con diferentes cantidades de árboles

for n_trees in n_trees_range:

    # Crear y entrenar el modelo RandomForest con n_trees árboles

    rf_model      =      RandomForestClassifier(bootstrap=True,      n_estimators=n_trees,
    oob_score=True, random_state=42)

    rf_model.fit(X, y)


    # Obtener el error OOB y almacenarlo en la lista para RandomForest

    oob_error_rf = 1 - rf_model.oob_score_

    oob_errors_rf.append(oob_error_rf)


    # Crear y entrenar el modelo ExtraTrees con n_trees árboles

    extra_trees_model  =      ExtraTreesClassifier(bootstrap=True,      n_estimators=n_trees,
```

```
oob_score=True, random_state=42)

extra_trees_model.fit(X, y)


# Obtener el error OOB y almacenarlo en la lista para ExtraTrees

oob_error_extra_trees = 1 - extra_trees_model.oob_score_

oob_errors_extra_trees.append(oob_error_extra_trees)


# Visualizar la evolución del error OOB para RandomForest y ExtraTrees

plt.figure(figsize=(12, 6))


plt.subplot(1, 2, 1)

plt.plot(n_trees_range, oob_errors_rf, marker='o', linestyle='-', color='b')

plt.title('Evolución del Error Out-of-Bag con RandomForest')

plt.xlabel('Número de Árboles')

plt.ylabel('Error Out-of-Bag')

plt.grid(True)


plt.subplot(1, 2, 2)

plt.plot(n_trees_range, oob_errors_extra_trees, marker='o', linestyle='-', color='r')

plt.title('Evolución del Error Out-of-Bag con ExtraTrees')
```

```
plt.xlabel('Número de Árboles')
```

```
plt.ylabel('Error Out-of-Bag')
```

```
plt.grid(True)
```

```
plt.tight_layout()
```

```
plt.show()
```

Como resultado obtenemos la Figura 7. Vemos que el ExtraTrees con 20 árboles obtiene un error bajo y a medida que aumenta el número de árboles aumenta nuevamente el error. Se da el caso contrario con la implementación básica de Random Forest, donde el error se estabiliza con 65 árboles aproximadamente.

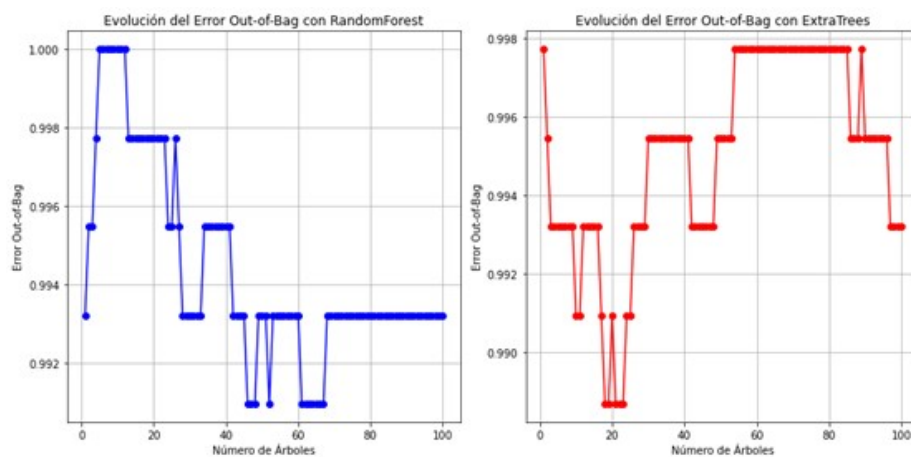


Figura 7. Evolución del OOB conjunto de datos diabetes entrenado con Random Forest vs ExtraTrees. Fuente: elaboración propia.

10.5. Cuaderno de ejercicios

- Dadas las siguientes predicciones de los árboles pertenecientes a un modelo Random Forest, ¿cuál es la predicción final del modelo?

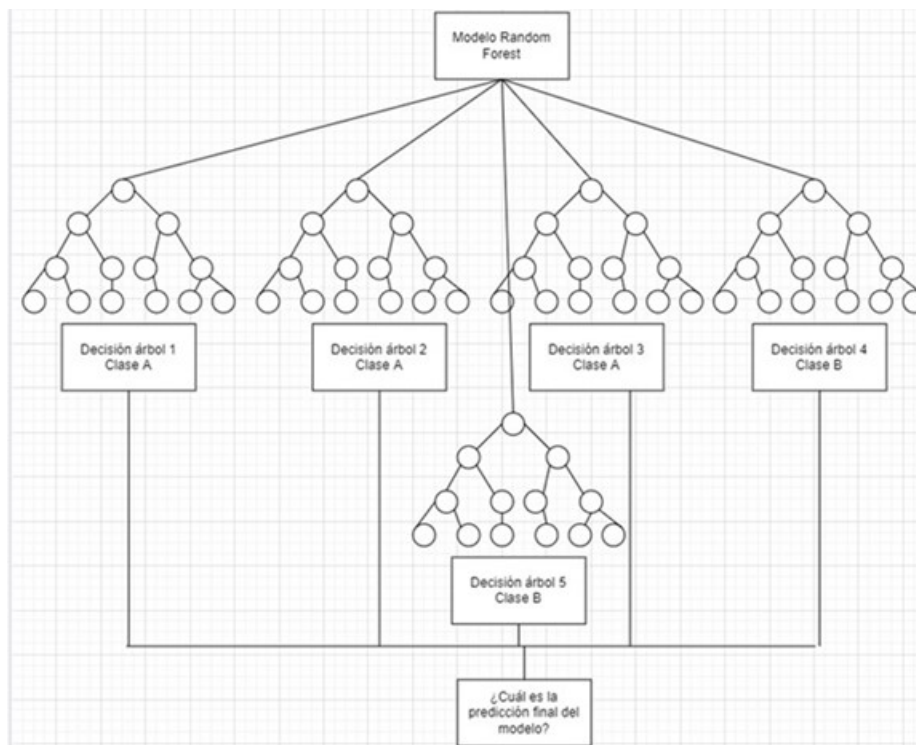


Figura 8. Predicción de cinco árboles de decisión, problema de clasificación. Fuente: elaboración propia.

SOLUCIÓN

La predicción se obtiene por voto mayoritario: Clase A.

- Dadas las siguientes predicciones de los árboles pertenecientes a un modelo Random Forest para un problema de regresión, ¿cuál es la predicción final del modelo?

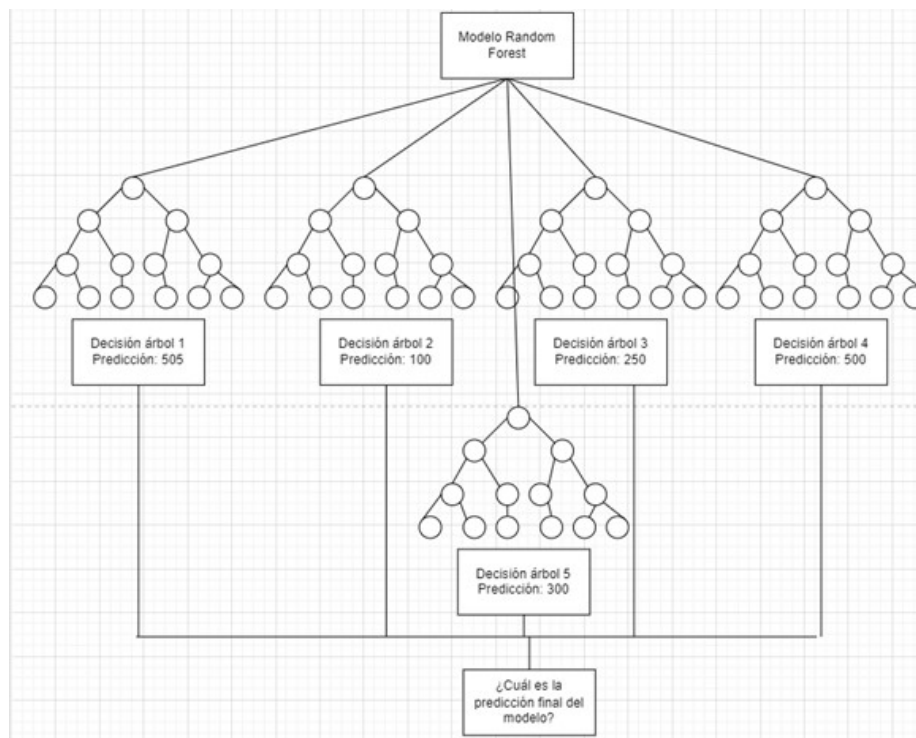


Figura 9. Predicción de cinco árboles de decisión, problema de regresión. Fuente: elaboración propia.

SOLUCIÓN

Es el promedio de las predicciones de los árboles: 331.

- En el gráfico se muestra la matriz de confusión y los resultados de las métricas asociadas a ella. La predicción se realiza con un modelo Random Forest. ¿Es un buen modelo? ¿Qué conclusiones podemos sacar de las métricas?

	Reales	
Predichos	A	B
A	36405	623
B	42	58
Accuracy	0,982089	
Sensibilidad	0,998848	
Especificidad	0,085169	
Recall	0,998848	
Precision	0,983175	
F1	0,990949	

SOLUCIÓN

No es un buen modelo, porque tiene buena sensibilidad y pésima especificidad. Seguramente es porque estamos frente a unos datos desbalanceados.

RECORDEMOS

Precisión $TP/TP+FP$

Nos da la calidad de la predicción: ¿qué porcentaje de los que hemos dicho que son la clase positiva en realidad lo son?

Recall - sensibilidad $TP/TP+FN$

Nos da la cantidad: ¿qué porcentaje de la clase positiva hemos sido capaces de identificar? F1 combina Precision y Recall en una sola medida.

- ▶ Con el conjunto de datos `breast_cancer` de la librería `sklearn`, realiza las siguientes tareas:
 - Divide el conjunto de datos en conjuntos de entrenamiento y prueba.

- Entrena un modelo de Random Forest y uno de ExtraTrees para predecir la presencia o ausencia de la enfermedad.
- Evalúa el rendimiento del modelo en el conjunto de prueba.
- Visualiza la importancia de las variables en el modelo.

SOLUCIÓN

```
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier

from sklearn.datasets import load_breast_cancer

import matplotlib.pyplot as plt

import numpy as np

# Cargar el conjunto de datos Diabetes

breast_cancer = load_breast_cancer()

X, y = breast_cancer.data, breast_cancer.target

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

# Crear un modelo de RandomForestClassifier

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Entrenar el modelo

rf_model.fit(X_train, y_train)

y_pred = rf_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
```

```
print(f'Exactitud del modelo Random Forest: {accuracy}')
```



```
# Obtener importancias de las variables

importances_rf = rf_model.feature_importances_
```



```
# Crear un modelo de ExtraTreesClassifier

extra_trees_model = ExtraTreesClassifier(n_estimators=100, random_state=42)
```



```
# Entrenar el modelo

extra_trees_model.fit(X_train, y_train)
```



```
y_pred = extra_trees_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f'Exactitud del modelo Extra Trees: {accuracy}')
```



```
# Obtener importancias de las variables

importances_extra_trees = extra_trees_model.feature_importances_
```



```
# Obtener nombres de las variables
```

```
feature_names = breast_cancer.feature_names

# Visualizar importancias de las variables para RandomForest

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.barh(range(len(importances_rf)), importances_rf, align='center')

plt.yticks(range(len(importances_rf)), feature_names)

plt.xlabel('Importancia Relativa')

plt.ylabel('Variables')

plt.title('Importancia de las Variables en RandomForest')

# Visualizar importancias de las variables para ExtraTrees

plt.subplot(1, 2, 2)

plt.barh(range(len(importances_extra_trees)), importances_extra_trees,
align='center')

plt.yticks(range(len(importances_extra_trees)), feature_names)

plt.xlabel('Importancia Relativa')

plt.ylabel('Variables')

plt.title('Importancia de las Variables en ExtraTrees')
```

```
plt.tight_layout()
```

```
plt.show()
```

Exactitud del modelo Random Forest: 0.9649122807017544.

Exactitud del modelo Extra Trees: 0.9649122807017544.

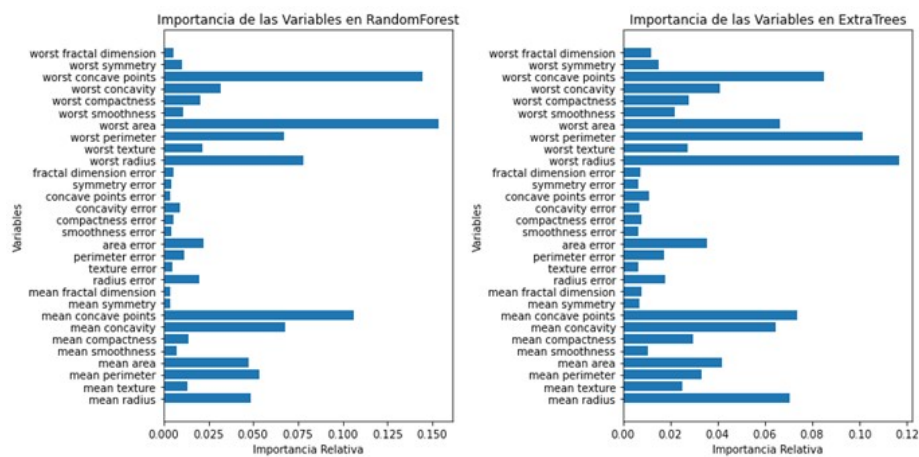


Figura 10. Diagrama de variables más relevantes en Random Forest y ExtraTrees. Fuente: elaboración propia.

- Con el mismo conjunto de datos anterior y con las mismas tareas, entrena los dos algoritmos y compara la evolución del OOB.

```
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
```

```
from sklearn.datasets import load_diabetes
```

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
# Cargar el conjunto de datos Diabetes
```

```
diabetes = load_diabetes()

X, y = diabetes.data, diabetes.target

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


# Inicializar listas para almacenar el número de árboles y los errores OOB para RandomForest y
ExtraTrees

n_trees_range = range(1, 101) # Ajusta según sea necesario

oob_errors_rf = []

oob_errors_extra_trees = []


# Entrenar modelos con diferentes cantidades de árboles

for n_trees in n_trees_range:

    # Crear y entrenar el modelo RandomForest con n_trees árboles

    rf_model = RandomForestClassifier(bootstrap=True, n_estimators=n_trees, oob_score=True,
random_state=42)

    rf_model.fit(X_train, y_train)


# Obtener el error OOB y almacenarlo en la lista para RandomForest

oob_error_rf = 1 - rf_model.oob_score_

oob_errors_rf.append(oob_error_rf)
```

```
# Crear y entrenar el modelo ExtraTrees con n_trees árboles

extra_trees_model = ExtraTreesClassifier(bootstrap=True, n_estimators=n_trees, oob_score=True,
random_state=42)

extra_trees_model.fit(X_train, y_train)


# Obtener el error OOB y almacenarlo en la lista para ExtraTrees

oob_error_extra_trees = 1 - extra_trees_model.oob_score_

oob_errors_extra_trees.append(oob_error_extra_trees)


# Visualizar la evolución del error OOB para RandomForest y ExtraTrees

plt.figure(figsize=(12, 6))


plt.subplot(1, 2, 1)

plt.plot(n_trees_range, oob_errors_rf, marker='o', linestyle='-', color='b')

plt.title('Evolución del Error Out-of-Bag con RandomForest')

plt.xlabel('Número de Árboles')

plt.ylabel('Error Out-of-Bag')

plt.grid(True)
```

```
plt.subplot(1, 2, 2)

plt.plot(n_trees_range, oob_errors_extra_trees, marker='o', linestyle='-', color='r')

plt.title('Evolución del Error Out-of-Bag con ExtraTrees')

plt.xlabel('Número de Árboles')

plt.ylabel('Error Out-of-Bag')

plt.grid(True)

plt.tight_layout()

plt.show()
```

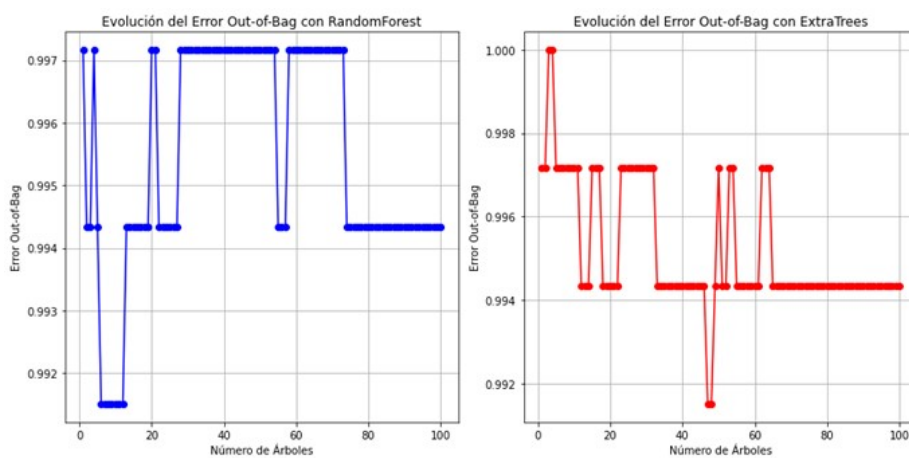


Figura 11. Diagramas de evolución del error en función del número de árboles. Fuente: elaboración propia.

10.6. Referencias bibliográficas

Breiman, L. (2001). Random Forest. *Machine Learning*, 45(1), 5-32.

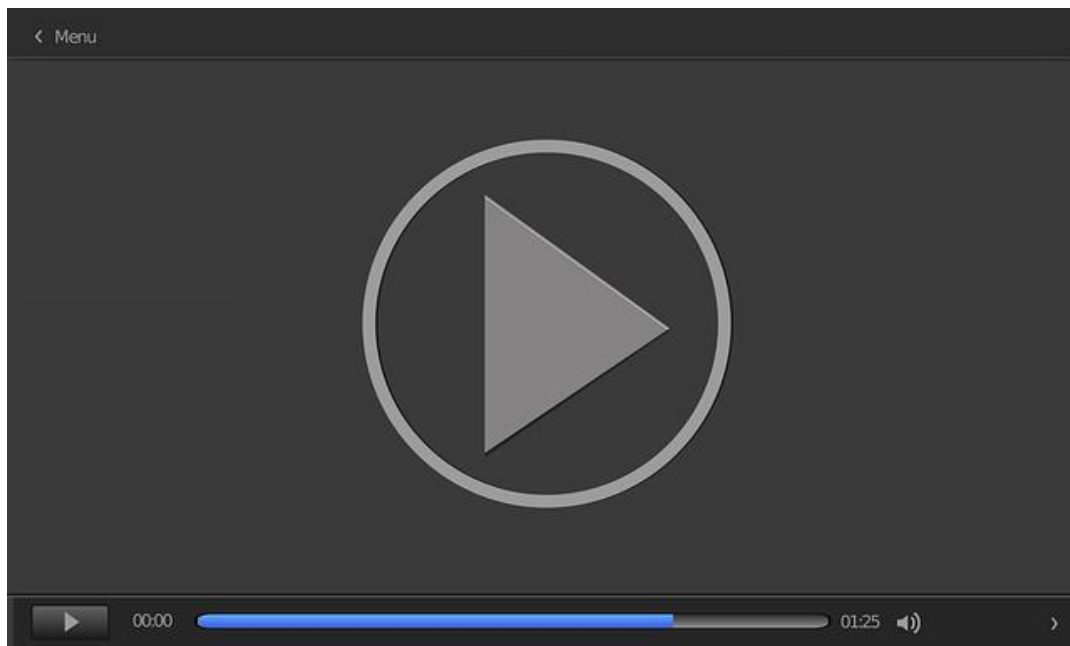
Brownlee, J. (2016). *Master Machine Learning Algorithms. Discover How They Work and Implement Them From Scratch*. Github. <https://github.com/linux08/machine-learning-books/blob/master/Master%20Machine%20Learning%20Algorithms%202016.pdf>

James, G., Witten, D., Hastie, T. y Tibshirani, R. (2017). *An Introduction to Statistical Learning with Applications in R*. Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V. y Thirion, B. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830. <https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf>

Ensamblaje de modelos

Un geógrafo en YouTube. (2023, enero 12). *Clusterización con Random Forest en R* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=4DESM1e46TQ>



Accede al vídeo:

<https://www.youtube.com/embed/4DESM1e46TQ>

Fabio combina técnicas de aprendizaje automático con técnicas SIG para resolver un problema de cartografía climática en Colombia.

Demystifying the random forest

Ramesh, S. (2023). *Demystifying the Random Forest. Deconstructing and Understanding this Beautiful Algorithm.* Medium.
<https://towardsdatascience.com/demystifying-the-random-forest-8a46f4fd416f>

Explica el modelo de Random Forest con un ejemplo codificado en Python paso a paso.

1. ¿Cuál es el propósito principal de Random Forest en aprendizaje automático?
 - A. Reducir el sesgo.
 - B. Reducir la varianza.
 - C. Aumentar el sesgo.
 - D. Aumentar la varianza.

2. En Random Forest, ¿cómo se realiza la toma de decisiones final para un problema de clasificación?
 - A. Votación.
 - B. Promedio.
 - C. Suma ponderada.
 - D. Máxima probabilidad.

3. ¿Cuál es el propósito principal de utilizar múltiples árboles en un bosque de Random Forest?
 - A. Aumentar la complejidad del modelo.
 - B. Aumentar el sesgo.
 - C. Reducir la varianza y generalizar el modelo.
 - D. Reducir el sesgo.

4. En el algoritmo Random Forest, ¿qué es el Out-of-Bag (OOB) error?
 - A. El error en el conjunto de prueba.
 - B. El error en el conjunto de entrenamiento.
 - C. El error en las instancias no utilizadas durante el entrenamiento.
 - D. El error en el conjunto de validación.

5. El algoritmo Extra Trees destaca por:
 - A. Utilizar árboles de decisión poco profundos.
 - B. Seleccionar características de manera más aleatoria durante la construcción de árboles.
 - C. Utilizar técnicas avanzadas de regularización.
 - D. Requiere un mayor número de árboles para converger.

6. ¿Cuál es uno de los beneficios de Extra Trees en comparación con Random Forest?
 - A. Mayor interpretabilidad de los modelos.
 - B. Mayor velocidad de entrenamiento.
 - C. Menor capacidad para manejar datos ruidosos.
 - D. Mayor sensibilidad al sobreajuste.

7. En Random Forest, ¿cómo se calcula la importancia de las variables?
 - A. Suma de las predicciones.
 - B. Suma de las impurezas en los nodos donde se realiza una división.
 - C. Número de nodos en los árboles.
 - D. Número de variables en el conjunto de datos.

8. ¿Qué hiperparámetro en Random Forest controla la cantidad de características consideradas en cada división de un árbol?
 - A. nestimators.
 - B. maxfeatures.
 - C. maxdepth.
 - D. minsamplesplit.

9. ¿En qué contexto sería preferible utilizar Extra Trees en lugar de Random Forest?

- A. Cuando se busca una mayor interpretabilidad del modelo.
- B. Cuando se tiene un conjunto de datos pequeño.
- C. Cuando se desea mayor velocidad de entrenamiento.
- D. Cuando se necesita un modelo altamente preciso.

10. ¿Qué pasa si aumento el número de árboles a un número muy grande en Random Forest en términos de desempeño del modelo?

- A. Aumenta la precisión, pero también la varianza.
- B. Aumenta la precisión y reduce la varianza.
- C. Disminuye la precisión y reduce el sesgo.
- D. Es deseable analizar la disminución del error en términos del número de árboles para ver el desempeño del modelo.