Modelos de ensamble

Flavio E. Spetale

2023

Método de ensamble

Un ensamble de clasificadores es una combinación de las decisiones individuales de cada uno de ellos, para clasificar nuevas instancias (Dzeroski & Zenki, 2000).

Existen varias razones que justifican el ensamble de clasificadores:

1. Los datos para training pueden no proveer suficiente información para elegir un único mejor clasificador debido a que el tamaño disponible en estos datos es pequeño (Dietterich, 2000).

2. La combinación redundante y complementaria de clasificadores mejora la robustez, exactitud y generalidad de toda la clasificación (Kotsiantis & Pintelas, 2004b).

3. Diferentes clasificadores utilizan diferentes técnicas y métodos de representación de los datos, lo que permite obtener resultados de clasificación con diferentes patrones de generalización.

4. Los ensambles son frecuentemente mucho más exactos que los clasificadores individuales (Dzeroski & Zenki, 2000).

Método de ensamble

Los ensambles de aprendizaje computacional son una técnica de aprendizaje automático que combina varios modelos para obtener un resultado mejor que el de cualquiera de los modelos individuales. Esta técnica se utiliza para mejorar la precisión de los modelos de aprendizaje automático, ya que se combinan los resultados de varios modelos para obtener un resultado más preciso.

Se pueden clasificar en tres tipos principales:

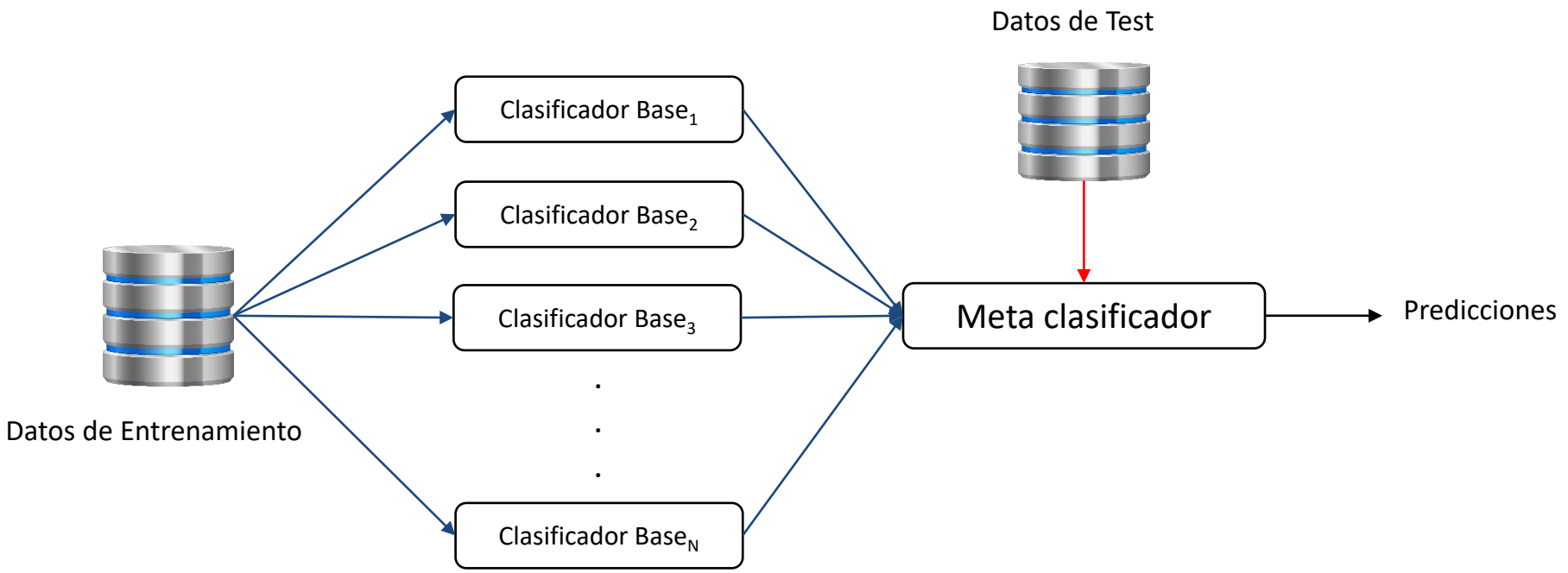
Ensamblado por promedio: Esta técnica promedia los resultados de varios modelos para obtener un resultado más preciso y se utiliza cuando los modelos tienen resultados similares y no hay mucha variación entre los resultados.

Ensamblado por votación: Esta técnica se basa en la idea de que el resultado más preciso se obtiene al contar los votos de los modelos individuales y se utiliza cuando los modelos tienen resultados muy diferentes y hay mucha variación entre los resultados.

Ensamblado por mezcla: Esta técnica combina los resultados de los modelos individuales para obtener un resultado más preciso y se utiliza cuando los modelos tienen resultados similares y hay mucha variación entre los resultados.

Esta técnica se puede aplicar a cualquier tipo de problema de aprendizaje automático, clasificación o regresión.

Método de ensamble



Método de ensamble: Ventajas

Los ensambles tienen varias ventajas sobre los modelos individuales.

- Son menos propensos a sobreajustar los datos, lo que significa que son más precisos. Esto se debe a que los modelos individuales se ajustan a los datos de entrenamiento, lo que puede llevar a resultados no precisos.
- Son más robustos, ya que un modelo individual puede fallar si los datos cambian, pero los ensambles son menos propensos a fallar debido a la diversidad de los modelos.
- Son más fáciles de implementar que los modelos individuales. Esto se debe a que los modelos individuales pueden ser complejos y requerir una configuración específica para obtener resultados precisos.
- Son más fáciles de configurar y entrenar, lo que los hace más accesibles para los usuarios.

Método de ensamble: Desventajas

Requieren una gran cantidad de recursos computacionales para entrenar. Esto significa que los ensambles son costosos de implementar, especialmente si se usan grandes conjuntos de datos.

Pueden ser difíciles de entrenar debido a la complejidad de los modelos.

Pueden ser difíciles de interpretar. Esto significa que es difícil entender qué partes del modelo están contribuyendo a los resultados finales y puede ser un problema si se necesita entender el comportamiento del modelo para tomar decisiones.

Pueden ser propensos a sobreajuste. Esto significa que los modelos pueden comenzar a aprender los ruidos en los datos en lugar de los patrones reales y puede reducir la precisión del modelo y hacer que los resultados sean menos fiables.

Método de ensamble: Desventajas

Existen dos opciones para la construcción de clasificadores ensambles.

1. Combina clasificadores trabajando en paralelo, es cuando todos los clasificadores base utilizan el mismo modelo o algoritmo (p. ej. árboles de decisión), así que será necesario manipular el conjunto de entrenamiento original para generar diferentes clasificadores base y poder tomar una decisión conjunta a partir de la decisión parcial de cada uno de ellos.

2. Consiste en combinar clasificadores base muy diferentes (p. ej. árboles de decisión y redes neuronales) de forma secuencial, de forma que cada clasificador utilice los resultados de un clasificador anterior, intentando capturar alguna característica clave de la naturaleza de los datos o del problema a resolver.

Método de ensamble: Combinación paralela de clasificadores base similares

Se generan una gran cantidad de clasificadores base construidos a partir de la alteración del conjunto de entrenamiento original.

Todos estos clasificadores base son muy parecidos, al estar basados en ligeras variaciones del conjunto de entrenamiento, pero no son idénticos, proporcionando diversidad al clasificador combinado, el cual combina (mediante un esquema de votación) las clasificaciones parciales para tomar una decisión.

Existen dos técnicas básicas para la creación del clasificador combinado, en función de cómo se genera el conjunto de entrenamiento de cada clasificador parcial y del peso asignado a cada uno de ellos llamadas *bagging y boosting*

Bagging

La idea básica del bagging es utilizar el conjunto de entrenamiento original para generar centenares o miles de conjuntos similares usando muestreo con reemplazo, i.e., de un conjunto de N elementos se pueden escoger N' ≤ N al azar, existiendo la posibilidad de escoger un mismo elemento más de una vez (de ahí «con reemplazo»).

Normalmente N' = N, por lo que existirán elementos repetidos. Aunque no es tan habitual, los conjuntos generados también pueden usar una dimensionalidad d' ≤ d, de forma que no todas las mismas variables disponibles existan en los conjuntos generados.

Esto puede ayudar a reducir el grado de colinealidad entre variables, haciendo emerger variables relevantes que pueden quedar siempre descartadas en frente de otra variable.

Una vez se han construido los conjuntos de entrenamiento parciales, se construye un clasificador para cada uno de ellos, siendo los árboles de decisión la opción más típica.

Teniendo en cuenta la naturaleza del clasificador de ensamble, no es necesario crear clasificadores base muy precisos, ya que el objetivo es que los errores cometidos por cada clasificador base queden minimizados en frente de la decisión de la mayoría.

En el caso de árboles de decisión, lo que es habitual es crear clasificadores más pequeños (limitados en profundidad) reduciendo el coste computacional de todo el conjunto.

Bagging

La decisión final se toma por mayoría, dando el mismo peso a todas las decisiones parciales, i.e., la clase resultante del clasificador combinado es aquella que aparece más veces entre las decisiones parciales tomadas por los clasificadores base.

Una manera de medir el error cometido por el clasificador combinado se conoce como *out-of-bag*, dado que el error estimado es el promedio de todos los errores parciales cometidos por cada clasificador base para todos aquellos elementos no usados en el conjunto de entrenamiento usado para construirlo.

De esta manera no es necesario separar los datos de entrada en un conjunto de entrenamiento y otro de test, sino que se usan todos para construir el clasificador de ensamble.

No obstante, si se dispone de suficientes datos, es *siempre recomendable utilizar un conjunto de test para validar el clasificador de ensamble.*

Algoritmo Bagging

El algoritmo de bagging (Leo Breiman, 1996) consta de tres pasos:

Bootstrap: Este método de remuestreo genera diferentes subconjuntos del conjunto de datos de entrenamiento seleccionando puntos de datos al azar y con reemplazo.

Entrenamiento paralelo: Los diferentes subconjuntos son entrenados de forma inde-pendiente y en paralelo entre sí utilizando clasificadores base.

Agregación: Finalmente, dependiendo de la tarea (regresión o clasificación), se toma el promedio o la mayoría de las predicciones generadas por los clasificadores base.

Bagging

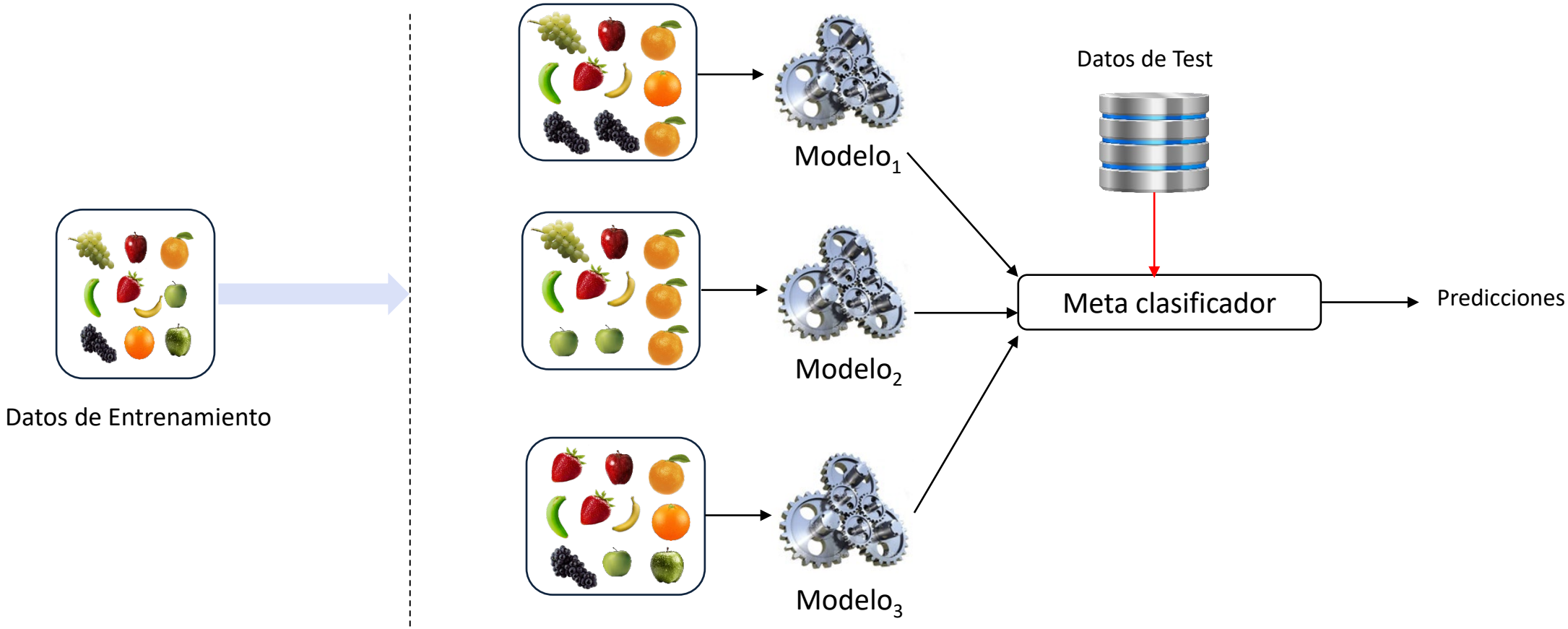
Ventajas:

- Método simple y efectivo.
- Reducción de la varianza. Útil con datos de alta dimensión donde los valores perdidos pueden conducir a una mayor varianza.

Desventajas:

- Pérdida de interpretabilidad. Es difícil obtener información precisa a través de bagging debido al método agregación involucrado.
- Elevado costo computacional. Se vuelve más lento a medida que aumenta el número de iteraciones. No es adecuado para aplicaciones en tiempo real.

Algoritmo Bagging



Random Forest

¿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, i.e., "cuánto en promedio son los valores predecidos 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 sobreajuste 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".

Random Forest

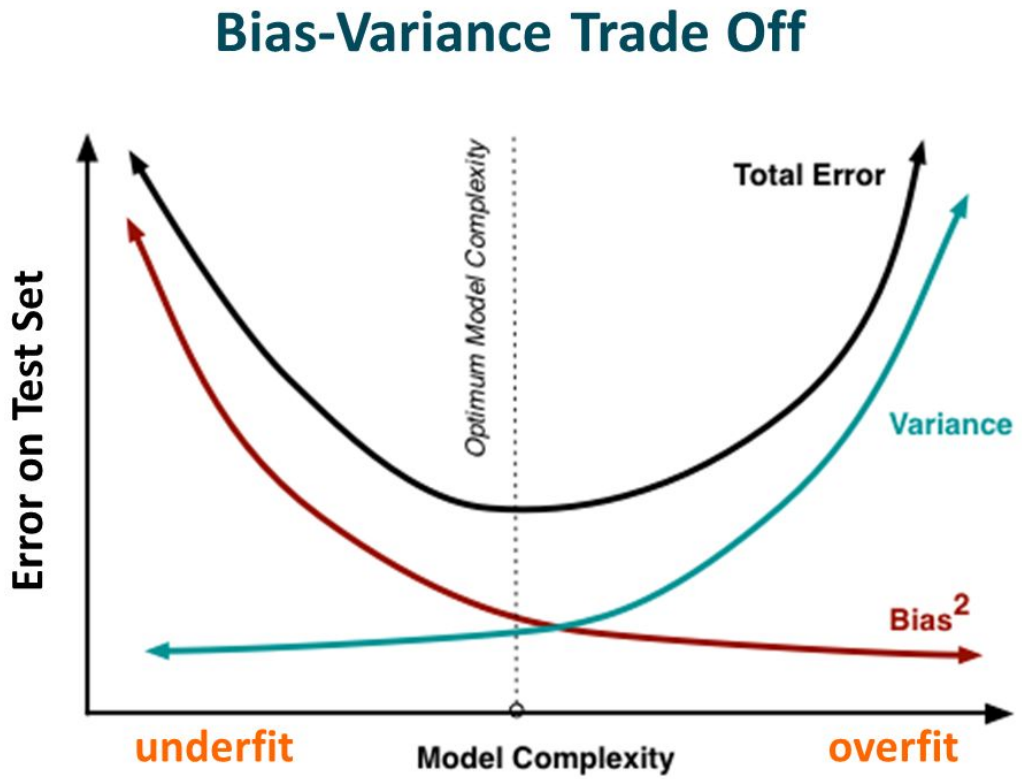


image credit: scott.fortmann-roe.com

Algoritmo Random Forest

El algoritmo Random Forest (Breiman, 2001) es una técnica de aprendizaje supervisado que genera múltiples árboles de decisión sobre un conjunto de datos de entrenamiento.

1. Toma una muestra aleatoria de m observaciones con repetición (bootstrap)

2. Hace crecer el árbol de decisión usando el subconjunto anterior en función de:

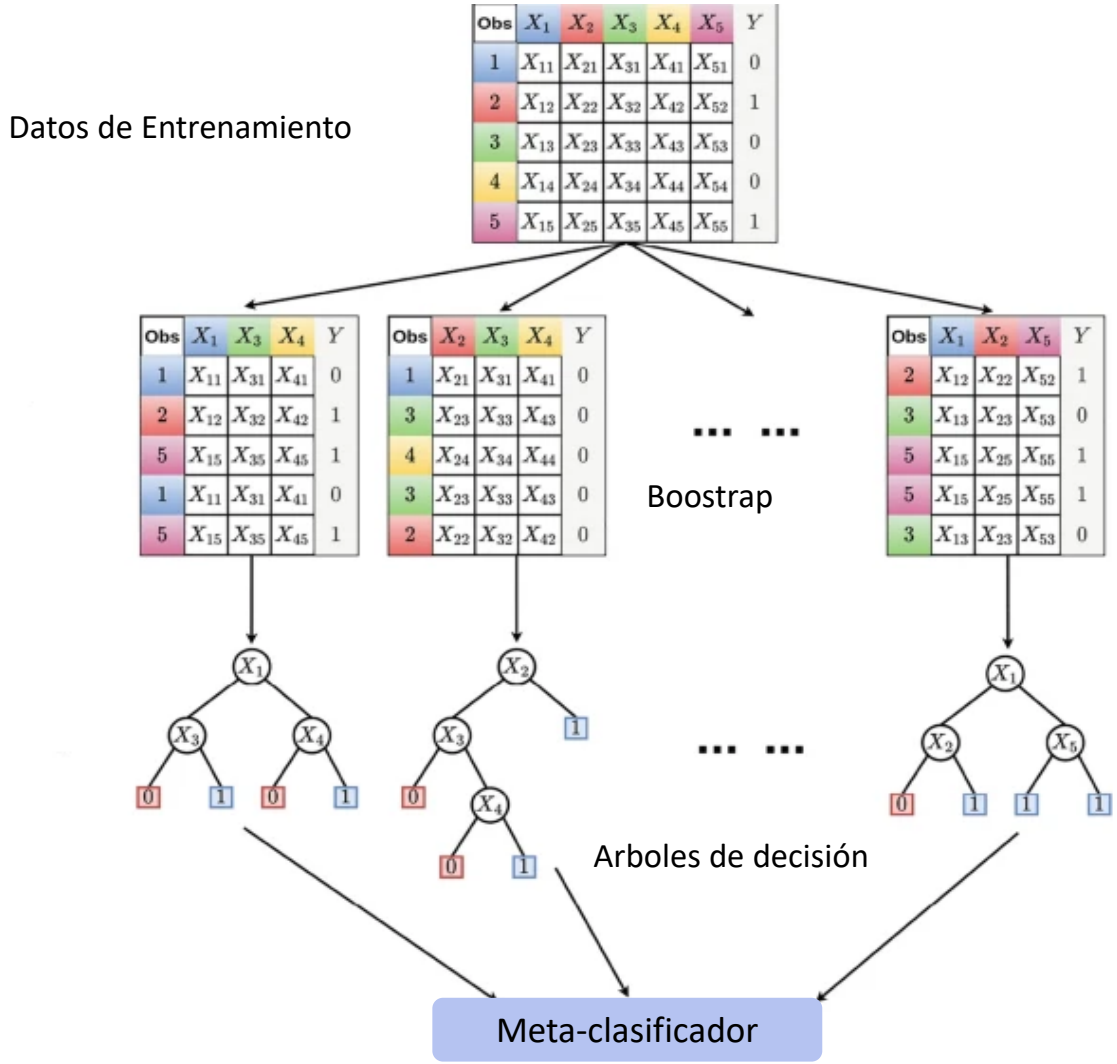
    A. La cantidad de n atributos seleccionados de forma aleatoria del subconjunto.

    B. Cada árbol crece hasta su máxima extensión basado en la función objetivo (maximizando la ganancia de información).

3. Repetir los pasos 1 y 2 para k árboles.

4. Obtiene una predicción final basada en los resultados de las predicciones de los k árboles por medio del voto mayoritario.

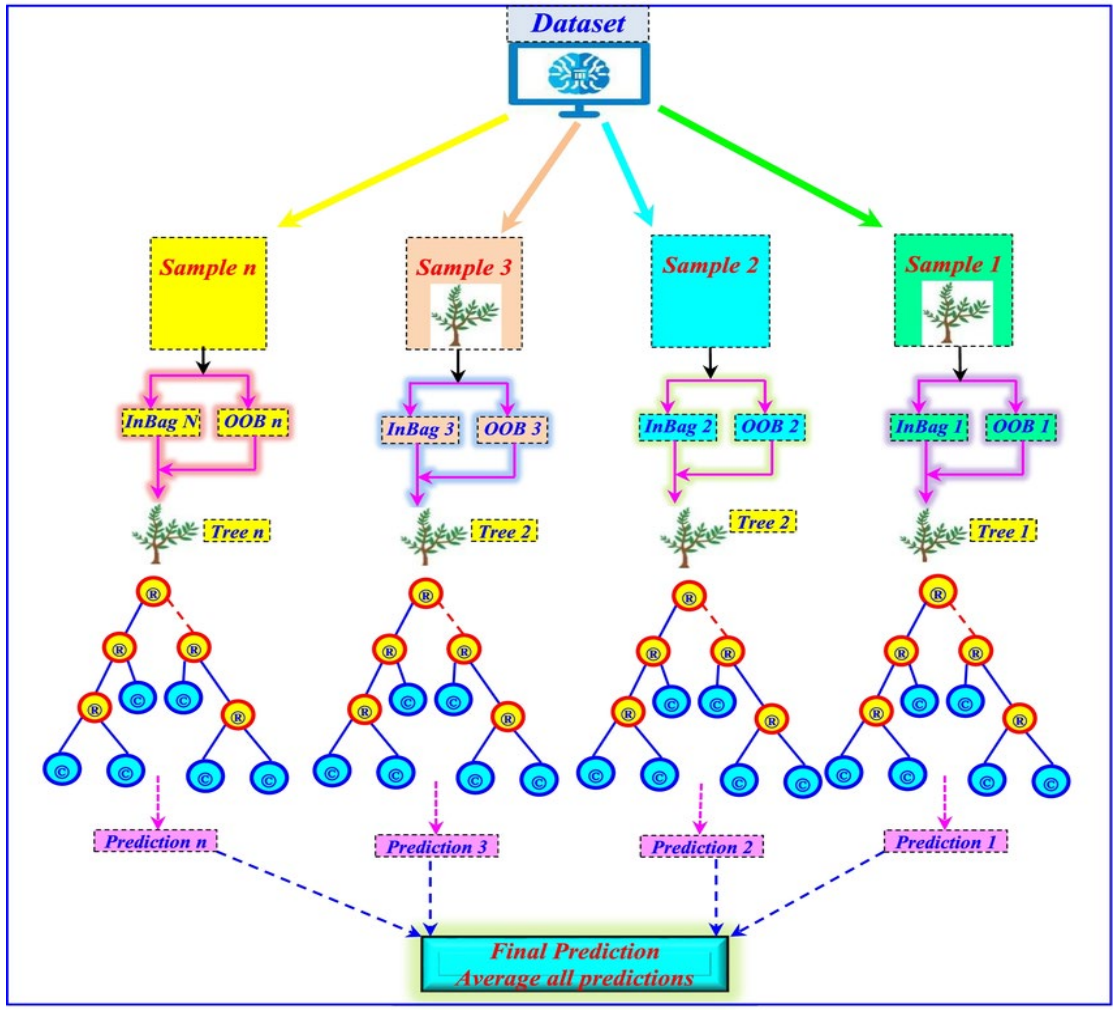
Algoritmo Random Forest



Random Forest: 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.

Random Forest: Out of bag samples y out of bag error



DOI:10.1007/s11356-022-21201-1

Random Forest: Hyper-parámetros

ntree: número de árboles en el bosque. Se quiere estabilizar el error, pero usar demasiados árboles puede ser innecesariamente ineficiente.

mtry: número de variables aleatorias como candidatas en cada ramifiación.

sampsize: el número de muestras sobre las cuales entrenar. El valor por defecto es 63.25%. Valores más bajos podrían introducir sesgo y reducir el tiempo. Valores más altos podrían incrementar el rendimiento del modelo pero a riesgo de causar overfitting. Generalmente se mantiene en el rango 60-80%.

nodesize: mínimo número de muestras dentro de los nodos terminales. Equilibrio entre bias-varianza

maxnodes: máximo número de nodos terminales.

Random Forest: Hyper-parámetros

Ventajas:

1. Tiene un desempeño muy eficiente y es una de las técnicas más certeras en bases de datos grandes.

2. Puede manejar muchos atributos sin excluir ninguno y logra estimar cuáles son los atributos más importantes, es por ello que esta técnica también se utiliza para reducción de dimensionalidad.

3. Mantiene su precisión con proporciones grandes de datos perdidos.

Desventajas:

1. La visualización gráfica de los resultados puede ser difícil de interpretar.

2. Puede sobre ajustar ciertos grupos de datos en presencia de ruido.

3. Se tiene poco control sobre lo que hace el modelo.

Ejemplo en python

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score,
ConfusionMatrixDisplay
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from scipy.stats import randint

# Tree Visualisation
from sklearn.tree import export_graphviz
from IPython.display import display
import graphviz

dataWheat = pd.read_csv("wheat.csv")

xWheat = dataWheat.drop('category', axis=1)
yWheat = dataWheat['category']
xWheatTrain, xWheatTest, yWheatTrain, yWheatReal = train_test_split(xWheat, yWheat, test_size=0.2)
```

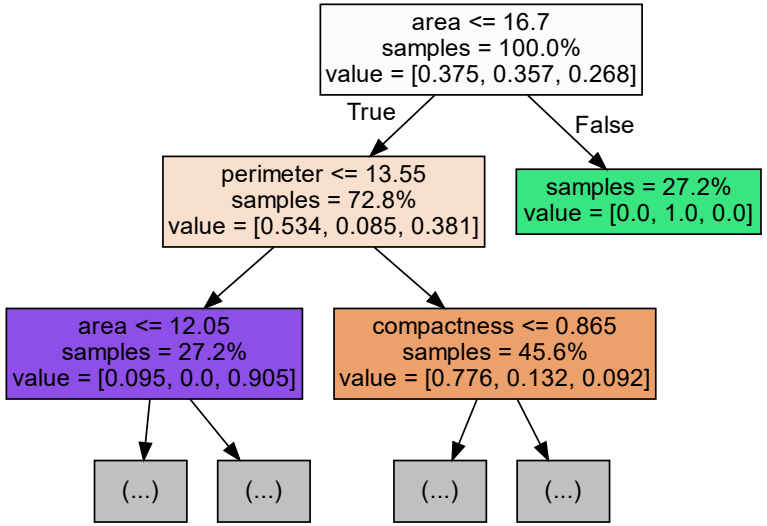
Ejemplo en python

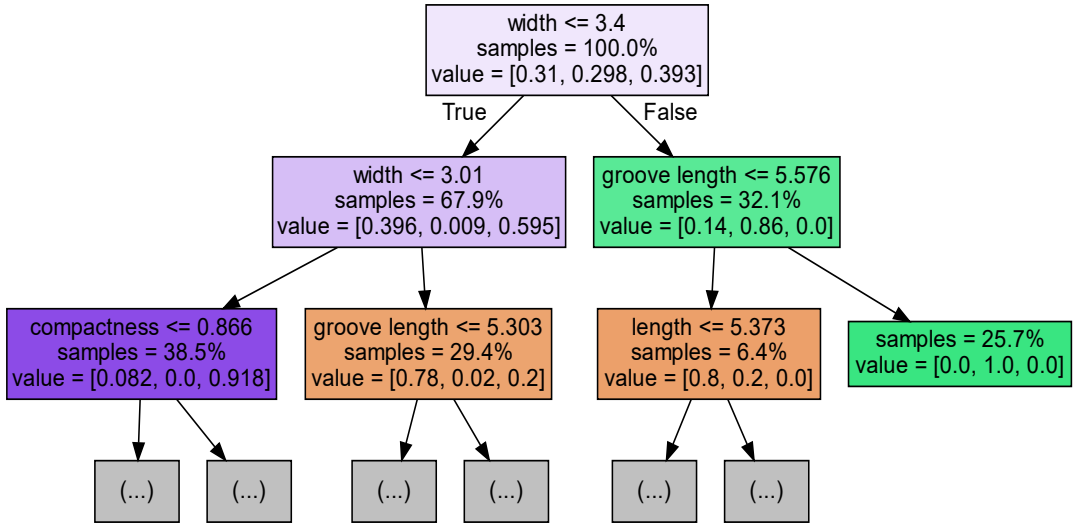
```
rf = RandomForestClassifier()
rf.fit(xWheatTrain, yWheatTrain)

y_pred = rf.predict(xWheatTest)

accuracy = accuracy_score(yWheatReal, y_pred)
print("Accuracy:", accuracy)
```

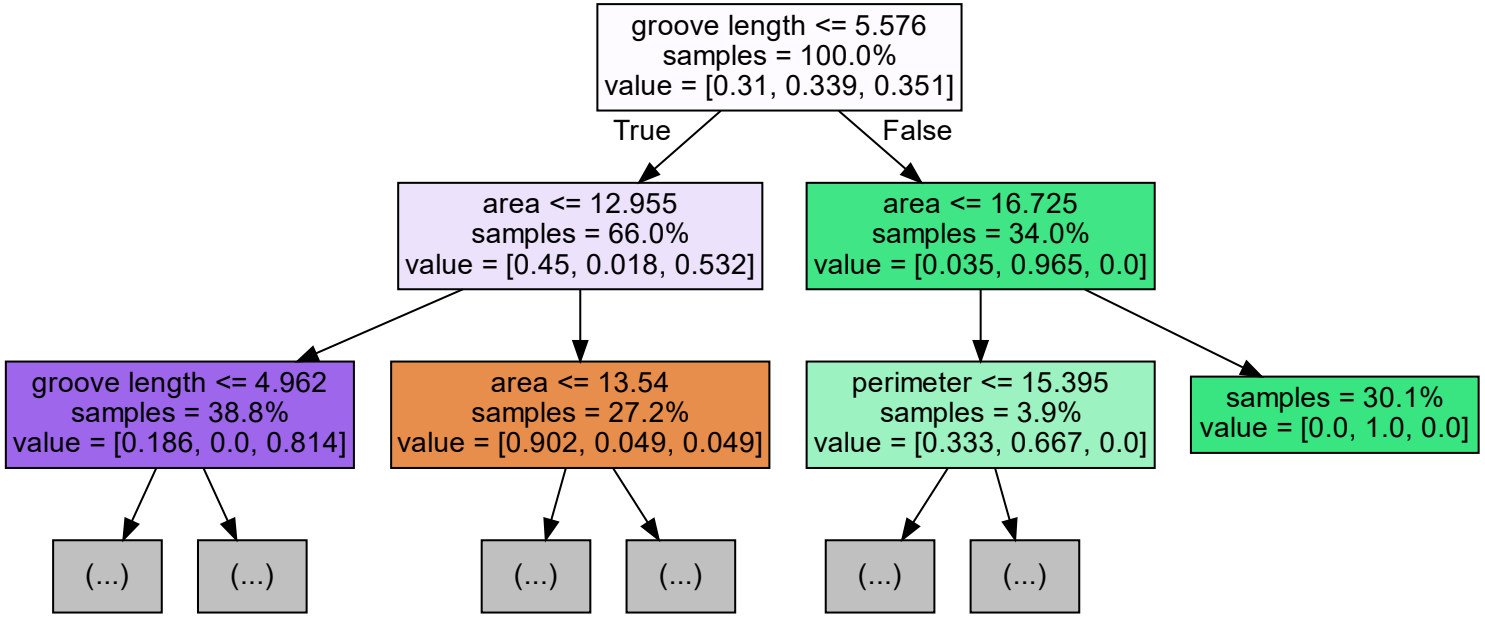
Accuracy: 0.976

```
# Export the first three decision trees from the forest
for i in range(3):
    tree = rf.estimators_[i]
    dot_data = export_graphviz(tree,
                               feature_names=xWheatTrain.columns,
                               filled=True, max_depth=2, impurity=False,
                               proportion=True)
    graph = graphviz.Source(dot_data)
    display(graph)
```

Ejemplo en python





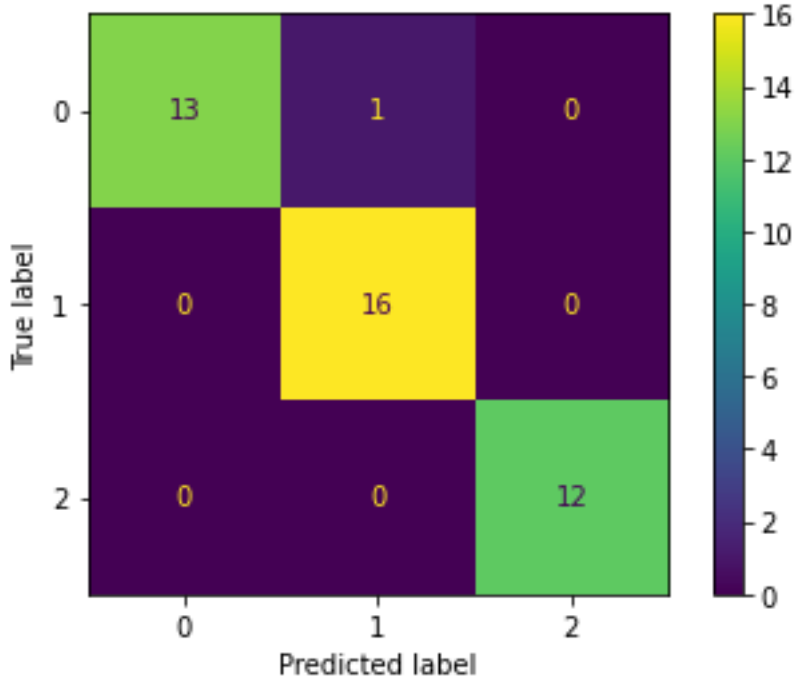


Ejemplo en python

```
param_dist = {'n_estimators': randint(50,500), 'max_depth': randint(1,20)}

rf = RandomForestClassifier()

# Use random search to find the best hyperparameters
rand_search = RandomizedSearchCV(rf, param_distributions = param_dist, n_iter=5, cv=5)

rand_search.fit(xWheatTrain, yWheatTrain)

print('Best hyperparameters:',  rand_search.best_params_)

Best hyperparameters: {'max_depth': 9, 'n_estimators': 451}

best_rf = rand_search.best_estimator_

yWheatPred = best_rf.predict(xWheatTest)

cm = confusion_matrix(yWheatReal, yWheatPred)
```

Ejemplo en python

```
ConfusionMatrixDisplay(confusion_matrix=cm).plot();

precision_score(yWheatReal, yWheatPred, average='macro')
0.980

recall_score(yWheatReal, yWheatPred, average='macro')
0.976

accuracy_score(yWheatReal, yWheatPred)
0.976
```



Boosting

La idea del boosting es ligeramente diferente. Se parte también de clasificadores base muy sencillos (también llamados débiles) de los cuales se supone que, al menos, cometen menos errores que aciertos, i.e, que funcionan ligeramente mejor que un clasificador aleatorio.

Se empieza construyendo un primer clasificador base usando el conjunto de entrenamiento original. Como el clasificador es débil (por ejemplo, un árbol de decisión de profundidad limitada), se cometen unos cuantos errores para el conjunto de entrenamiento.

Entonces, para construir el siguiente clasificador base, lo que se hace es ponderar los elementos del conjunto de entrenamiento, de forma que aquellos que han sido clasificados erróneamente por el clasificador anterior tengan más peso y, de una manera u otra, tengan mayor peso también en el proceso de creación del siguiente clasificador base.

El clasificador combinado es la combinación de la predicción del primer clasificador base más la del segundo, ponderadas de acuerdo a algún esquema de pesos que tenga en cuenta la precisión de cada clasificador.

Este clasificador combinado también cometerá unos errores que se usarán para dar más peso a aquellos elementos erróneamente clasificados, construyendo un tercer clasificador combinado, etc.

Boosting

En cada etapa se construye un clasificador nuevo que combina una serie de decisiones sucesivas que tienden a enfocarse en aquellos elementos más difíciles de clasificar, intentando que cada etapa corrija los errores de la anterior, pero sin estropear las decisiones correctas ya tomadas.

Para ello, la secuencia de pesos de los sucesivos clasificadores parciales suele ser descendiente, de forma que el procedimiento se para cuando el siguiente clasificador ya no aporta nada respecto al anterior.

Es importante destacar que el proceso de creación del clasificador combinado es secuencial, dado que para realizar una nueva iteración y obtener un nuevo clasificador combinado es necesario haber evaluado los clasificadores base anteriores.

No obstante, una vez se ha alcanzado el clasificador combinado final, la evaluación se realiza también en paralelo, ya que en una primera etapa se genera la decisión parcial para cada uno de los clasificadores base y en la segunda se obtiene la clasificación final ponderando todas la decisiones parciales.

Boosting


Original Data
Weighted Data
Weighted Data
Classifer
Classifer
Classifer
Ensemble
Classifer

Tipos de Boosting

- **Boosting adaptativo o AdaBoost:** la creación del algoritmo AdaBoost se le atribuye a Yoav Freund y Robert Schapire. Este método funciona de forma iterativa; identifica puntos de datos mal clasificados y ajusta sus pesos para minimizar el error de entrenamiento. El modelo continúa optimizándose de manera secuencial hasta que obtiene el predictor más fuerte.

- **Boosting de gradiente:** sobre la base del trabajo de Leo Breiman, Jerome H. Friedman desarrolló el boosting de gradiente, que funciona agregando secuencialmente predictores a un conjunto en el que cada uno de ellos corrige los errores de su predecesor. Sin embargo, en lugar de cambiar el peso de los puntos de datos como AdaBoost, el boosting de gradiente entrena los errores residuales del predictor anterior. Se le llama boosting de gradiente porque combina el algoritmo de descenso de gradiente y el método de boosting.

- **Boosting de gradiente extremo o XGBoost:** XGBoost es una implementación de boosting de gradiente diseñada para mejorar la velocidad y la escalabilidad computacionales. XGBoost emplea múltiples núcleos en la CPU, que permiten que el aprendizaje ocurra en paralelo durante el entrenamiento.

Boosting

Ventajas:

- Precisión mejorada: el impulso puede mejorar la precisión del modelo al combinar las precisiones de varios modelos débiles y promediarlas para la regresión o votarlas para la clasificación para aumentar la precisión del modelo final.
- Robustez frente al sobreajuste: el impulso puede reducir el riesgo de sobreajuste al volver a ponderar las entradas que están clasificadas incorrectamente.
- Mejor manejo de datos desequilibrados: Boosting puede manejar los datos desequilibrados centrándose más en los puntos de datos que están mal clasificados.
- Mejor interpretabilidad: el impulso puede aumentar la interpretabilidad del modelo al dividir el proceso de decisión del modelo en múltiples procesos.

Desventajas:

- Son vulnerables a los valores atípicos.
- Es difícil utilizar para aplicaciones en tiempo real.
- Es computacionalmente costoso para grandes conjuntos de datos.

Stacking

Stacking se refiere a los métodos que aprovechan la complementariedad mutua entre los modelos base para mejorar el rendimiento y mejorar la capacidad de generalización.

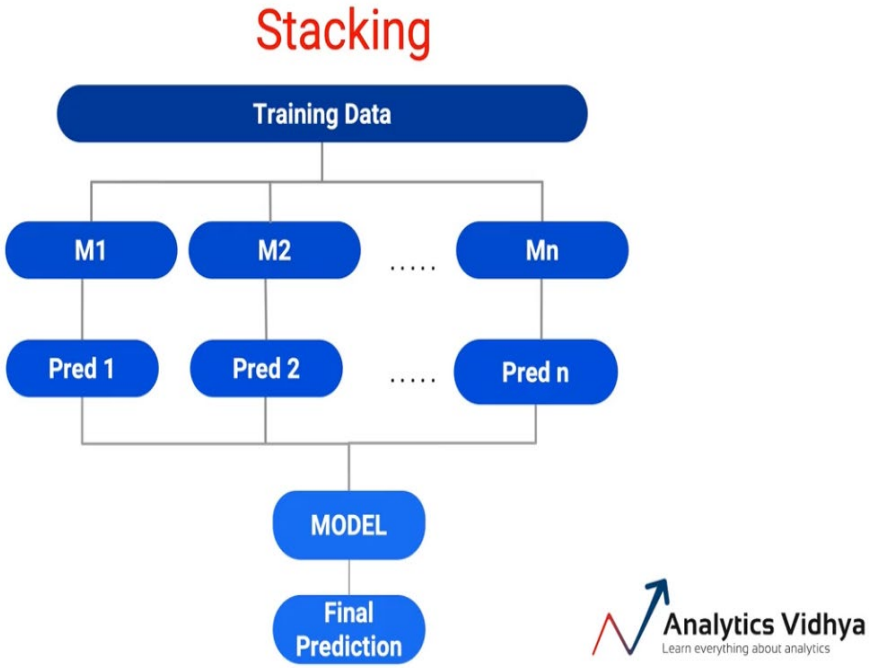
En general, stacking consta de dos fases: fase de entrenamiento de modelos base y fase de entrenamiento de metamodelos.

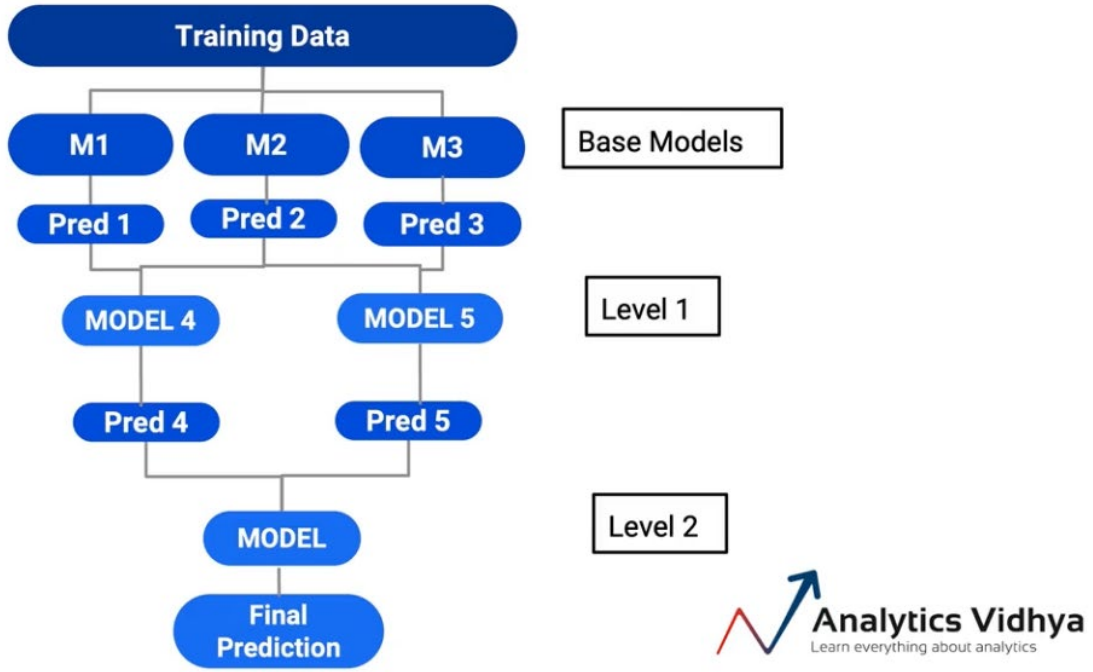
En la primera fase, los datos originales se dividen en conjunto de entrenamiento y conjunto de prueba, y el conjunto de entrenamiento se entrena utilizando la validación cruzada k-fold. La validación cruzada de k veces divide el conjunto de entrenamiento en k partes y cada parte usa las (k − 1) partes restantes de los datos para entrenar el modelo y simular las predicciones para esa parte de los datos.

En la segunda fase, el conjunto de datos de predicciones obtenido después de la validación cruzada de k veces del modelo base se vuelve a ensamblar, en el orden del conjunto de datos de entrenamiento original, para obtener un nuevo conjunto de entrenamiento.

El conjunto de entrenamiento del metamodelo se obtiene fusionando el nuevo conjunto de entrenamiento de los múltiples modelos base. De manera similar, las predicciones del conjunto de pruebas de los modelos base se combinan para obtener el conjunto de pruebas del metamodelo. El metamodelo se entrena en función del nuevo conjunto de datos.

Stacking


Stacking
Training Data
M1
M2
.....
Mn
Pred 1
Pred 2
.....
Pred n
MODEL
Final Prediction
Analytics Vidhya
Learn everything about analytics


Training Data
M1
M2
M3
Base Models
Pred 1
Pred 2
Pred 3
MODEL 4
MODEL 5
Level 1
Pred 4
Pred 5
MODEL
Level 2
Final Prediction
Analytics Vidhya
Learn everything about analytics