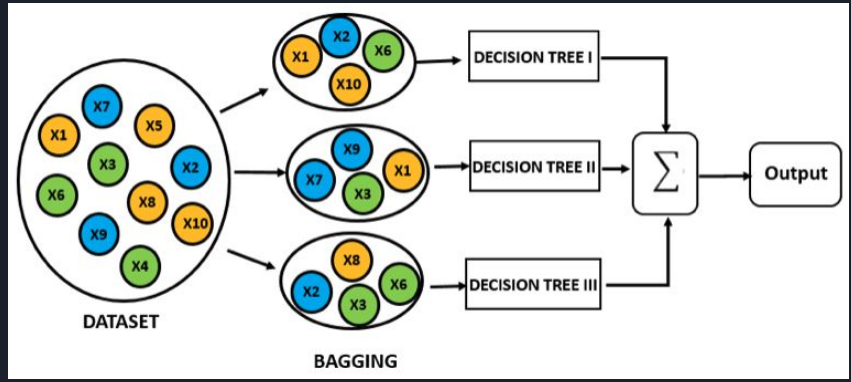
Random Forest

Erik Manuel Velásquez Rodríguez

Definición: Random Forest

Random forests o random decision forests son un método de aprendizaje por conjuntos para clasificación, regresión y otras tareas que operan mediante la construcción de una multitud de árboles de decisión en el momento del entrenamiento.

Es una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos. Es una modificación sustancial de bagging que construye una larga colección de árboles no correlacionados y luego los promedia

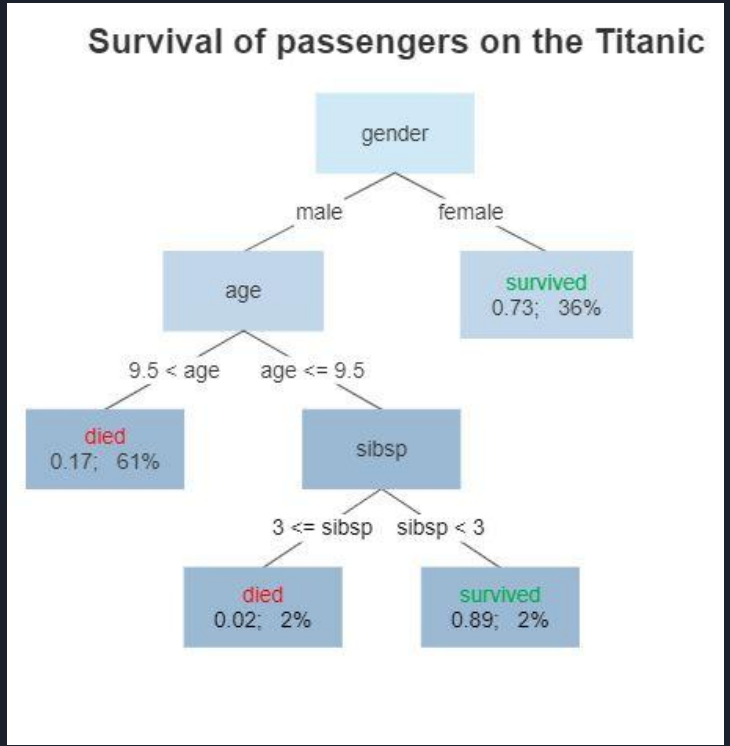


Árbol de decisión

Utiliza un árbol de decisión como un modelo predictivo que mapea observaciones.

Es uno de los enfoques de modelado predictivo utilizadas en estadísticas, minería de datos y aprendizaje automático. Los modelos de árbol, donde la variable de destino puede tomar un conjunto finito de valores se denominan árboles de clasificación.

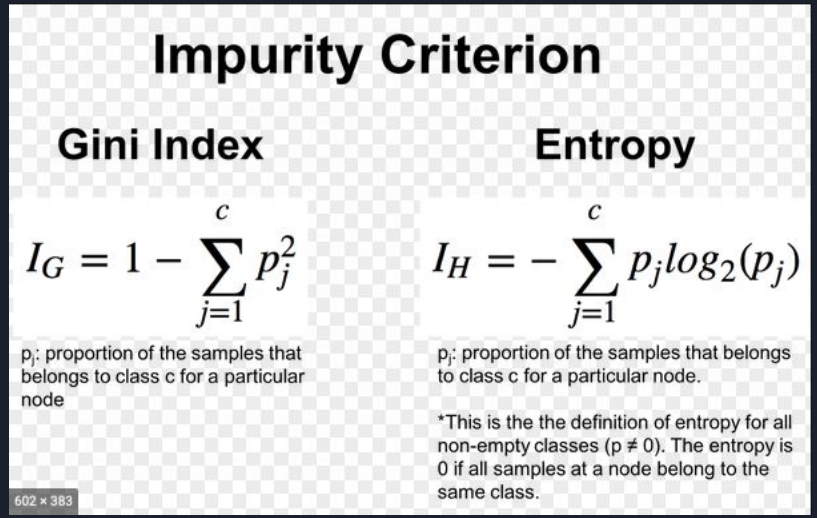
En estas estructuras de árbol, las hojas representan etiquetas y las ramas representan las características que conducen a esas etiquetas.



Árbol de decisión

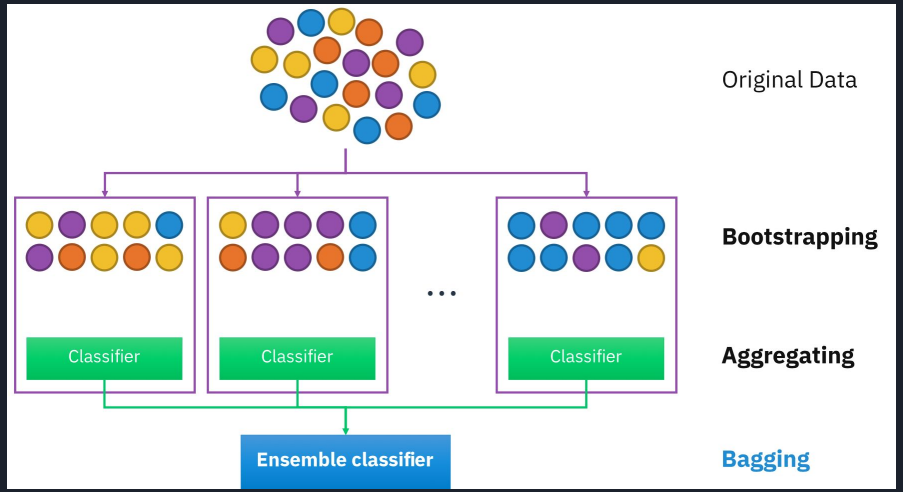
Los algoritmos para la construcción de árboles de decisión suelen trabajar de manera top-down, escogiendo en cada paso la variable que mejor divide el conjunto de elementos:

- **Impureza de Gini**: la impureza de Gini es una medida de cuán a menudo un elemento elegido aleatoriamente del conjunto sería etiquetado incorrectamente si fue etiquetado de manera aleatoria de acuerdo a la distribución de las etiquetas en el subconjunto
- Ganancia de la Información (**Entropía**): Ganancia de información se basa en el concepto de entropía de la teoría de la información.



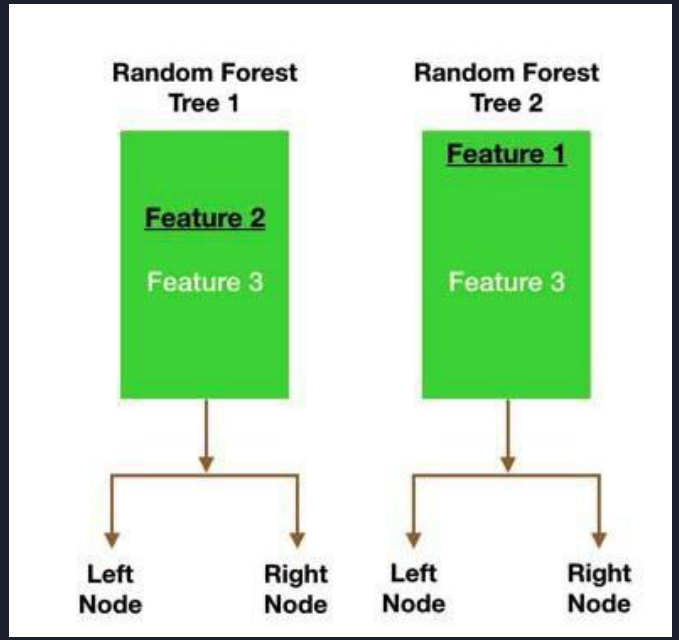
Random Forest: Bagging

Bagging (Bootstrap aggregating), es un meta-algoritmo de conjunto de aprendizaje automático diseñado para mejorar la estabilidad y precisión de los algoritmos de aprendizaje automático utilizados en la clasificación y regresión estadísticas. También reduce la variación y ayuda a evitar el sobreajuste. Aunque normalmente se aplica a métodos de árbol de decisión, se puede utilizar con cualquier tipo de método.



Random Forest: Selección de variables

El otro concepto principal en el Random forest es que solo se considera un subconjunto de todas las variables para dividir cada nodo en cada árbol de decisión. Generalmente, esto se establece para la clasificación, lo que significa que si hay 16 variables, en cada nodo de cada árbol, sólo se considerarán 4 variables aleatorias para dividir el nodo. (El random forest también se puede entrenar considerando todas las variables en cada nodo como es común en la regresión).

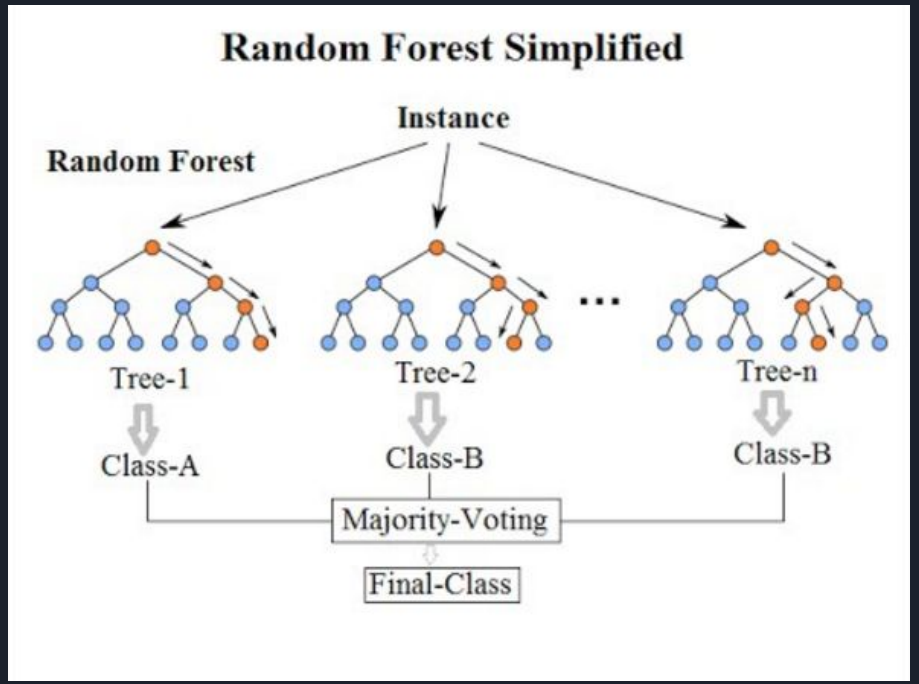


Random Forest

El random forest combina cientos o miles de árboles de decisión.

Entrena a cada uno en un conjunto ligeramente diferente de observaciones, dividiendo los nodos en cada árbol considerando un número limitado de variables.

Las predicciones finales del random forest se realizan promediando las predicciones de cada árbol individual.



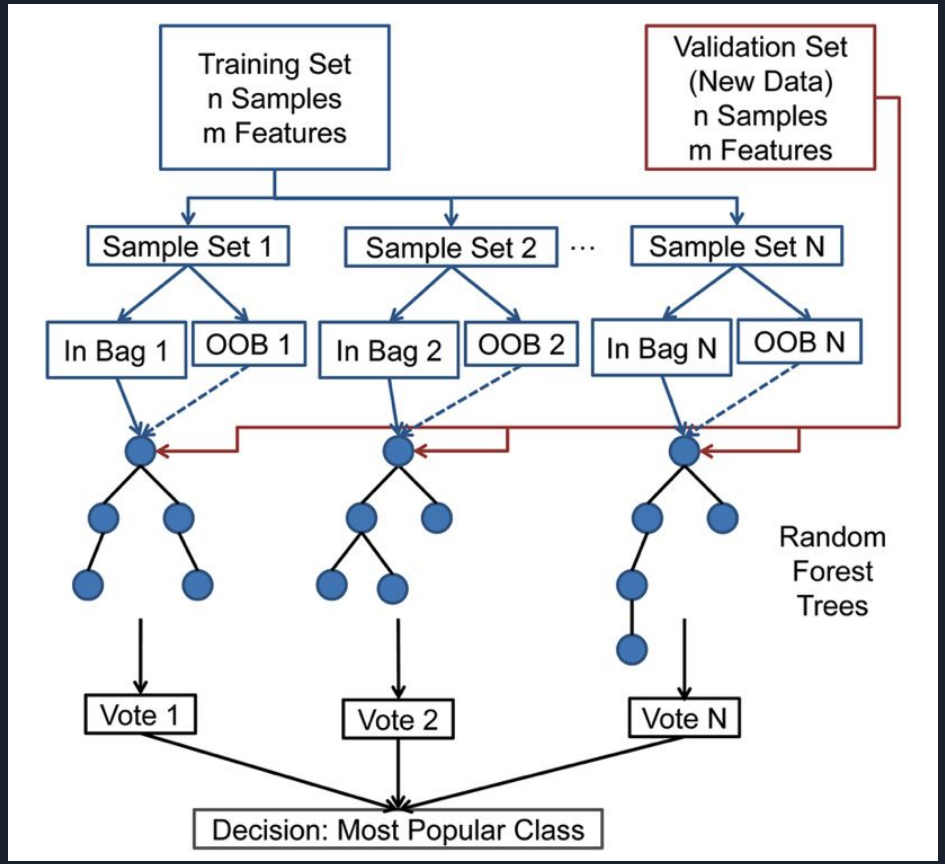
Random Forest: Algoritmo

Cada árbol es construido usando el siguiente algoritmo:

1. Sea N el número de casos de prueba, M es el número de variables en el clasificador.
2. Sea m el número de variables de entrada a ser usado para determinar la decisión en un nodo dado; m debe ser mucho menor que M
3. Elegir un conjunto de entrenamiento para este árbol y usar el resto de los casos de prueba para estimar el error.
4. Para cada nodo del árbol, elegir aleatoriamente m variables en las cuales basar la decisión. Calcular la mejor partición del conjunto de entrenamiento a partir de las m variables.

Para la predicción un nuevo caso es empujado hacia abajo por el árbol. Luego se le asigna la etiqueta del nodo terminal donde termina. Este proceso es iterado por todos los árboles en el ensamblado, y la etiqueta que obtenga la mayor cantidad de incidencias es reportada como la predicción.

Random Forest: Algoritmo



Random Forest: Desventajas

Se ha observado que Random forests efectúa un sobreajuste en ciertos grupos de datos con tareas de clasificación/regresión ruidosas.

A diferencia de los árboles de decisión, la clasificación hecha por random forests es difícil de interpretar.

Para los datos que incluyen variables categóricas con diferente número de niveles, el random forests se parcializa a favor de esos atributos con más niveles. Por consiguiente, la posición que marca la variable no es fiable para este tipo de datos.

Si los datos contienen grupos de atributos correlacionados con similar relevancia para el rendimiento, entonces los grupos más pequeños están favorecidos sobre los grupos más grandes.

Random Forest: Ventajas

Ser uno de los algoritmos de aprendizaje más certeros que hay disponible. Para un set de datos lo suficientemente grande produce un clasificador muy certero.

Correr eficientemente en bases de datos grandes.

Manejar cientos de variables de entrada sin excluir ninguna.

Dar estimaciones de qué variables son importantes en la clasificación.

Tener un método eficaz para estimar datos perdidos y mantener la exactitud cuando una gran proporción de los datos está perdida.

Computar los prototipos que dan información sobre la relación entre las variables y la clasificación.

Computar las proximidades entre los pares de casos que pueden usarse en los grupos, localizando valores atípicos, o (ascendiendo) dando vistas interesantes de los datos.

Ofrecer un método experimental para detectar las interacciones de las variables.

Random Forest: Librerías

Python:

- Scikit-learn (https://scikit-learn.org)

R:

- packages("randomForest")

Nota: Existe un gran variedad, estas son las más conocidas y usadas.

Ejemplos en Python

URL: https://github.com/velasquezerik/master-sgdi-final

Ejemplos de Regresión

Ejemplos de Clasificación

Librería:

- Sklearn
- Pandas

Ejemplo de Clasificación

Importing Libraries

In [1]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Importing the dataset

In [2]:
```
dataset = pd.read_csv('Social_Network_Ads.csv')
print(dataset)
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

```
      User ID  Gender  Age  EstimatedSalary  Purchased
0    15624510    Male   19            19000          0
1    15810944    Male   35            20000          0
2    15668575  Female   26            43000          0
3    15603246  Female   27            57000          0
4    15804002    Male   19            76000          0
..        ...     ...  ...              ...        ...
395  15691863  Female   46            41000          1
396  15706071    Male   51            23000          1
397  15654296  Female   50            20000          1
398  15755018    Male   36            33000          0
399  15594041  Female   49            36000          1

[400 rows x 5 columns]
```

https://github.com/velasquezerik/master-sgdi-final/blob/main/2-Random-Forest-Classification/random_forest_classification.ipynb

Ejemplo de Clasificación

Splitting the dataset into the Training set and Test set

In [3]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

Feature Scaling

In [4]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Fitting Random Forest Classification to the Training set

In [5]:
```
classifier = RandomForestClassifier(n_estimators=10, criterion='gini', random_state=0)
classifier = classifier.fit(X_train, y_train)
```

Predicting the Test set results

In [6]:
```
y_pred = classifier.predict(X_test)
```

Making the Confusion Matrix

In [7]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[64  4]
 [ 4 28]]
```

https://github.com/velasquezerik/master-sgdi-final/blob/main/2-Random-Forest-Classification/random_forest_classification.ipynb

Ejemplo de Regresión

Random Forest Regression

A random forest regressor.

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

Importing Libraries

In [1]:
```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
```

Importing the dataset

Simple dataset containing job position and level, along with salary. The idea is to estimate the salary based on the level, using a regression model.

Position,Level,Salary

In [2]:
```
dataset = pd.read_csv('Position_Salaries.csv')
print(dataset)
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values
```

```
            Position  Level   Salary
0   Business Analyst      1    45000
1  Junior Consultant      2    50000
2  Senior Consultant      3    60000
3            Manager      4    80000
4    Country Manager      5   110000
5     Region Manager      6   150000
6            Partner      7   200000
7     Senior Partner      8   300000
8            C-level      9   500000
9                CEO     10  1000000
```

https://github.com/velasquezerik/master-sgdi-final/blob/main/1-Random-Forest-Regression/random_forest_regression.ipynb

Ejemplo de Regresión

Fitting Random Forest Regression to the dataset

In [3]:
```
regressor = RandomForestRegressor(n_estimators=10, random_state=0)
regressor = regressor.fit(X, y)
```

Predicting a new result

In [4]:
```
y_pred = regressor.predict([[6.5]])
print(y_pred)
```

```
[167000.]
```

In [ ]:

https://github.com/velasquezerik/master-sgdi-final/blob/main/1-Random-Forest-Regression/random_forest_regression.ipynb

Muchas Gracias...