Random Forest

Random Forest is a supervised learning algorithm. Like you can already see from it's name, it creates a forest and makes it somehow random. The „forest“ it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

In [1]:
```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score


# load the data
df = sns.load_dataset('tips')
df.head()
```

Out[1]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [2]:
```
# encode features which are categorical or object using for loop
le = LabelEncoder()
for i in df.columns:
    if df[i].dtype == 'object' or df[i].dtype == 'category':
        df[i] = le.fit_transform(df[i])
df.head()
```

Out[2]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	0	0	2	0	2
1	10.34	1.66	1	0	2	0	3
2	21.01	3.50	1	0	2	0	3
3	23.68	3.31	1	0	2	0	2
4	24.59	3.61	0	0	2	0	4

In [3]:
```
# split the data into X and y for classification
X = df.drop('sex', axis = 1)
y = df['sex']
# train test split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)
# create, train and predict the mode
model_cl = RandomForestClassifier(n_estimators=200, random_state=42)
model_cl.fit(X_train, y_train)
y_pred = model_cl.predict(X_test)

#evaluate the model
print('accuracy score: ', accuracy_score(y_test, y_pred))
print(";;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;")
print('confusion matrix:\n', confusion_matrix(y_test, y_pred))
print(";;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;")
print('classification report:\n', classification_report(y_test, y_pred))
```

```
accuracy score:  0.6122448979591837
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
confusion matrix:
 [[ 7 12]
 [ 7 23]]
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
classification report:
               precision    recall  f1-score   support

           0       0.50      0.37      0.42        19
           1       0.66      0.77      0.71        30

    accuracy                           0.61        49
   macro avg       0.58      0.57      0.57        49
weighted avg       0.60      0.61      0.60        49
```

In [4]:
```
# USe random Forest for Regression task
X = df.drop('tip', axis = 1)
y = df['tip']

# train test split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)

#create, train and predict the model
model_reg = RandomForestRegressor()
model_reg.fit(X_train, y_train)
y_pred = model_reg.predict(X_test)

# evaluate the model
print('mean squared error: ', mean_squared_error(y_test, y_pred))
print(";;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;")
print('mean absolute error: ', mean_absolute_error(y_test, y_pred))
print(";;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;")
print('r2 score: ', r2_score(y_test, y_pred))
print(";;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;")
print('root mean squared error: ', np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
mean squared error:  0.9472707683673482
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
mean absolute error:  0.767748979591837
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
r2 score:  0.24216601288519235
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
root mean squared error:  0.9732783611934195
```