Lab 9 Assignment: Bagging and Boosting

Aim: The objective of this assignment is to compare the performance of bagging and boosting algorithms on a classification problem. You will use the scikit-learn library and the wine quality dataset to implement and evaluate the bagging and boosting methods.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Exercise 1

- Download the wine quality dataset and save it as winequality-white.csv in your working directory.

In [2]:
```
# Loading/Storing csv into dataframe named as Wine
df=pd.read_csv(r"C:\Users\raval\Downloads\winequality-white.csv",sep=";")
```

In [3]:
```
# printing the data frame
df
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.00100	3.00	0.45	8.8	6
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.99400	3.30	0.49	9.5	6
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.99510	3.26	0.44	10.1	6
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	6
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99560	3.19	0.40	9.9	6
...	...	...	...	...	...	...	...	...	...	...	...	...
4893	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	6
4894	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	5
4895	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	6
4896	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	7
4897	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	6

4898 rows × 12 columns

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

Exercise 2

- Load the dataset using pandas and split it into features (X) and target (y). The target variable is quality, which is a rating from 0 to 10. Convert the target variable into binary labels, where 1 means good quality (quality > 6) and 0 means bad quality (quality <= 6).**

In [5]:
```
X = df.drop(columns=["quality"])
y = df["quality"].apply(lambda x: 1 if x > 6 else 0)
```

In [6]:
```
X.shape
```

Out[6]:
```
(4898, 11)
```

In [7]:
```
print(y.shape)
y.head()
```

```
(4898,)
```

Out[7]:
```
0    0
1    0
2    0
3    0
4    0
Name: quality, dtype: int64
```

Exercise 3

- Split the data into training and test sets, using 80% of the data for training and 20% for testing. Use a random state of 42 for reproducibility.**

In [8]:
```
# Importing Train Test Split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Exercise 4

- Create a bagging classifier with 100 base estimators, using the default parameters of scikit-learn. Fit the classifier to the training data and predict the labels of the test data. Compute and print the accuracy score of the bagging classifier on the test data.**

In [9]:
```
# Importing Libraries
from sklearn.ensemble import BaggingClassifier
from sklearn.metrics import accuracy_score
```

In [10]:
```
# 100 Base Estimator
bagging_classifier = BaggingClassifier(n_estimators=100)
```

In [11]:
```
#Fitting training /testing data
bagging_classifier.fit(X_train,y_train)
```

Out[11]:
```
BaggingClassifier
BaggingClassifier(n_estimators=100)
```

In [12]:
```
# Accuracy
y_pred = bagging_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:",accuracy)
```

```
Accuracy: 0.8908163265306123
```

Exercise 5

*Create an AdaBoost classifier with 100 base estimators, using the default parameters of scikit-learn. Fit the classifier to the training data and predict the labels of the test data. Compute and print the accuracy score of the AdaBoost classifier on the test data.**

In [13]:
```
#importing library
from sklearn.ensemble import AdaBoostClassifier
```

In [14]:
```
# 100 estimators
AdaBoost_classifier= AdaBoostClassifier(n_estimators=100)
```

In [15]:
```
# fitting + prediction
AdaBoost_classifier.fit(X_train,y_train)
Ada_y_pred = AdaBoost_classifier.predict(X_test)
```

In [16]:
```
accuracy = accuracy_score(y_test, Ada_y_pred)
print("Accuracy:",accuracy)
```

```
Accuracy: 0.813265306122449
```

Exercise 6

- Compare the accuracy scores of the bagging and boosting classifiers. Which one performs better on this dataset? Why do you think so? Write your answer in a comment below your code.**

Accuracy of bagging algorithm is almost 89% while for boosting it is 81% , bagging performs better due to the state of dataset as it may contain data that has high variance, low bias which suits bagging method to prevent overfitting.