Bagging - Bootstrap Aggregation

Import Libraries

In [ ]:
```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
```

Load Dataset and Split

In [ ]:
```
data = datasets.load_wine(as_frame= True)
X = data.data
print(X)
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 22)
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0      14.23        1.71  2.43               15.6      127.0           2.80
1      13.20        1.78  2.14               11.2      100.0           2.65
2      13.16        2.36  2.67               18.6      101.0           2.80
3      14.37        1.95  2.50               16.8      113.0           3.85
4      13.24        2.59  2.87               21.0      118.0           2.80
..       ...         ...   ...                ...        ...            ...
173    13.71        5.65  2.45               20.5       95.0           1.68
174    13.40        3.91  2.48               23.0      102.0           1.80
175    13.27        4.28  2.26               20.0      120.0           1.59
176    13.17        2.59  2.37               20.0      120.0           1.65
177    14.13        4.10  2.74               24.5       96.0           2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0          3.06                  0.28             2.29             5.64  1.04
1          2.76                  0.26             1.28             4.38  1.05
2          3.24                  0.30             2.81             5.68  1.03
3          3.49                  0.24             2.18             7.80  0.86
4          2.69                  0.39             1.82             4.32  1.04
..          ...                   ...              ...              ...   ...
173        0.61                  0.52             1.06             7.70  0.64
174        0.75                  0.43             1.41             7.30  0.70
175        0.69                  0.43             1.35            10.20  0.59
176        0.68                  0.53             1.46             9.30  0.60
177        0.76                  0.56             1.35             9.20  0.61

     od280/od315_of_diluted_wines  proline
0                            3.92   1065.0
1                            3.40   1050.0
2                            3.17   1185.0
3                            3.45   1480.0
4                            2.93    735.0
..                            ...      ...
173                          1.74    740.0
174                          1.56    750.0
175                          1.56    835.0
176                          1.62    840.0
177                          1.60    560.0

[178 rows x 13 columns]
```

Build a Dicision Tree Model and Test on it

In [ ]:
```
dtree = DecisionTreeClassifier(random_state = 22)
dtree.fit(X_train,y_train)

y_pred = dtree.predict(X_test)

print("Train data accuracy:",accuracy_score(y_true = y_train, y_pred = dtree.predict(X_train)))
print("Test data accuracy:",accuracy_score(y_true = y_test, y_pred = y_pred))
```

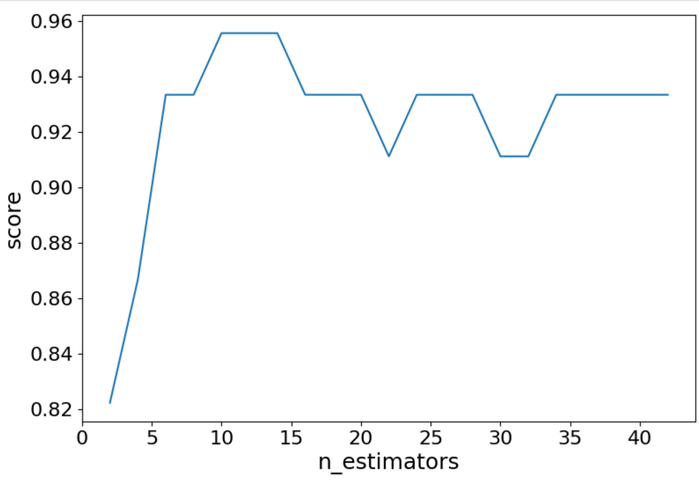
```
Train data accuracy: 1.0
Test data accuracy: 0.8222222222222222
```

we can see that Decision Tree Classifier overfits on train Data. Bias is low but variance is high.

Now let's check if Bagging Helps

In [ ]:
```
models = []
scores = []

# this is for validation of appropriate no. of estimators to be used
estimator_range = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42]

for n_estimators in estimator_range:

    # Create bagging classifier
    clf = BaggingClassifier(n_estimators = n_estimators, random_state = 22)

    # Fit the model
    clf.fit(X_train, y_train)

    # Append the model and score to their respective list
    models.append(clf)
    scores.append(accuracy_score(y_true = y_test, y_pred = clf.predict(X_test)))
```

In [ ]:
```
import matplotlib.pyplot as plt

# Generate the plot of scores against number of estimators
plt.figure(figsize=(9,6))
plt.plot(estimator_range, scores)

# Adjust labels and font (to make visable)
plt.xlabel("n_estimators", fontsize = 18)
plt.ylabel("score", fontsize = 18)
plt.tick_params(labelsize = 16)

# Visualize plot
plt.show()
```



Work for you to do on the same Dataset

```
- Divide the Dataset into 3 parts : Train, Validation and Test
- plot the validation graph as above for parameter : max_samples and max_features
- return the best accuracy on test set
```

Learn More here

- Bagging https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html
- Read on Random forest here http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

In [2]:
```
%%capture
!pip install nbconvert
!sudo apt-get install texlive-xetex texlive-fonts-recommended texlive-plain-generic
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```