EXPERIMENT NO. 5 B

- **Aim**: Use different voting mechanism and Apply AdaBoost (Adaptive Boosting), Gradient Tree Boosting (GBM), XGBoost classification on Iris dataset and compare the performance of three models using different evaluation measures. Dataset Link https://www.kaggle.com/datasets/uciml/iris

Hardware Requirement:

- 6 GB free disk space.
- 2 GB RAM.
- 2 GB of RAM, plus additional RAM for virtual machines.
- 6 GB disk space for the host, plus the required disk space for the virtual machine(s).
- Virtualization is available with the KVM hypervisor
- Intel 64 and AMD64 architectures

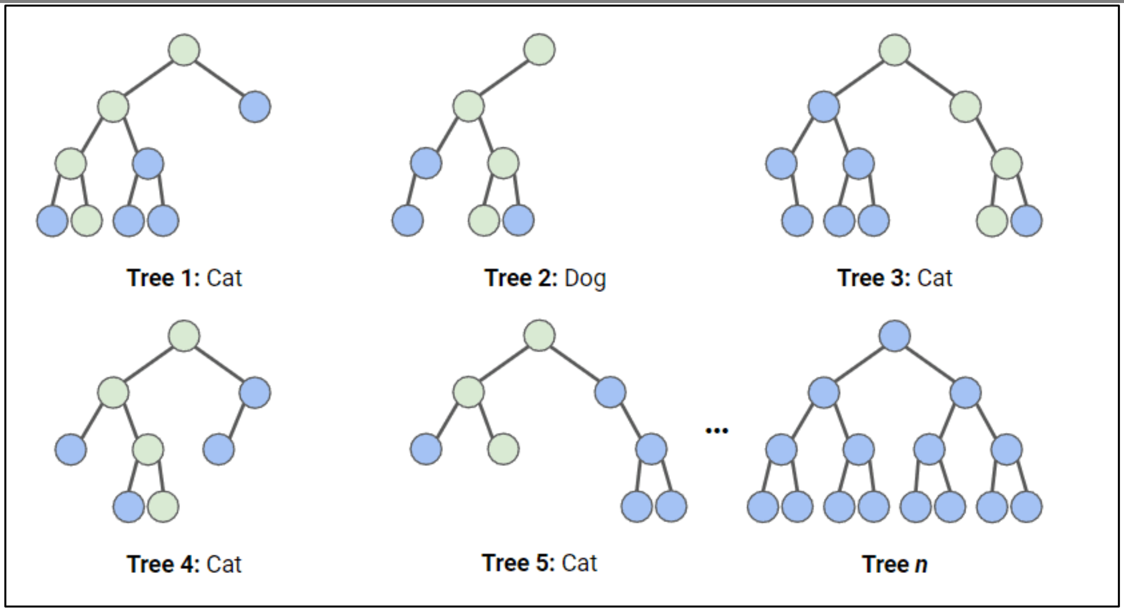
- **Software Requirement:**
  Jypiter Nootbook/Ubuntu

- **Theory:**

Imagine you have a complex problem to solve, and you gather a group of experts from different fields to provide their input. Each expert provides their opinion based on their expertise and experience. Then, the experts would vote to arrive at a final decision.

In a random forest classification, multiple decision trees are created using different random subsets of the data and features. Each decision tree is like an expert, providing its opinion on how to classify the data. Predictions are made by calculating the prediction for each decision tree, then taking the most popular result. (For regression, predictions use an averaging technique instead.)

In the diagram below, we have a random forest with n decision trees, and we've shown the first 5, along with their predictions (either "Dog" or "Cat"). Each tree is exposed to a different number of features and a different sample of the original dataset, and as such, every tree can be different. Each tree makes a prediction. Looking at the first 5 trees, we can see that 4/5 predicted the sample was a Cat. The green circles indicate a hypothetical path the tree took to reach its decision. The random forest would count the number of predictions from decision trees for Cat and for Dog, and choose the most popular prediction.



Implementation:

```
import pandas as pd
 from sklearn.datasets import load_digits
 digits = load_digits()

dir(digits)

['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names']

%matplotlib inline
 import matplotlib.pyplot as plt

plt.gray()
 for i in range(4):
     plt.matshow(digits.images[i])

<Figure size 640x480 with 0 Axes>

df = pd.DataFrame(digits.data)
 df.head()

     0    1    2     3     4     5    6    7    8    9  ...   54   55   56  \
0  0.0  0.0  5.0  13.0   9.0   1.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
1  0.0  0.0  0.0  12.0  13.0   5.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
2  0.0  0.0  0.0   4.0  15.0  12.0  0.0  0.0  0.0  0.0  ...  5.0  0.0  0.0
3  0.0  0.0  7.0  15.0  13.0   1.0  0.0  0.0  0.0  8.0  ...  9.0  0.0  0.0
4  0.0  0.0  0.0   1.0  11.0   0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
```

```
      57   58    59     60       61   62   63
 0  0.0  6.0  13.0  10.0   0.0  0.0  0.0
 1  0.0  0.0  11.0  16.0  10.0  0.0  0.0
 2  0.0  0.0   3.0  11.0  16.0  9.0  0.0
 3  0.0  7.0  13.0  13.0   9.0  0.0  0.0
 4  0.0  0.0   2.0  16.0   4.0  0.0  0.0

 [5 rows x 64 columns]
```

df['target'] = digits.target

df[0:12]

```
       0       1    2    3      4       5     6   7       8    9 ...  55   56   57  \
 0   0.0  0.0   5.0  13.0   9.0   1.0   0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
 1   0.0  0.0   0.0  12.0  13.0   5.0   0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
 2   0.0  0.0   0.0   4.0  15.0  12.0   0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
 3   0.0  0.0   7.0  15.0  13.0   1.0   0.0  0.0  0.0  8.0  ...  0.0  0.0  0.0
 4   0.0  0.0   0.0   1.0  11.0   0.0   0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
 5   0.0  0.0  12.0  10.0   0.0   0.0   0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
 6   0.0  0.0   0.0  12.0  13.0   0.0   0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
 7   0.0  0.0   7.0   8.0  13.0  16.0  15.0  1.0  0.0  0.0  ...  0.0  0.0  0.0
 8   0.0  0.0   9.0  14.0   8.0   1.0   0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
 9   0.0  0.0  11.0  12.0   0.0   0.0   0.0  0.0  0.0  2.0  ...  0.0  0.0  0.0
 10  0.0  0.0   1.0   9.0  15.0  11.0   0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
 11  0.0  0.0   0.0   0.0  14.0  13.0   1.0  0.0  0.0  0.0  ...  0.0  0.0  0.0

      58     59    60       61   62   63  target
 0     6.0  13.0  10.0   0.0  0.0  0.0         0
 1     0.0  11.0  16.0  10.0  0.0  0.0         1
 2     0.0   3.0  11.0  16.0  9.0  0.0         2
 3     7.0  13.0  13.0   9.0  0.0  0.0         3
 4     0.0   2.0  16.0   4.0  0.0  0.0    4
 5     9.0  16.0  16.0  10.0  0.0  0.0         5
 6     1.0   9.0  15.0  11.0  3.0  0.0         6
 7   13.0   5.0   0.0   0.0  0.0  0.0     7
 8   11.0  16.0  15.0  11.0  1.0  0.0     8
 9     9.0  12.0  13.0   3.0  0.0  0.0         9
 10   1.0  10.0  13.0   3.0  0.0  0.0     0
 11   0.0   1.0  13.0  16.0  1.0  0.0     1

 [12 rows x 65 columns]
```

Train and the model and prediction

```
X = df.drop('target',axis='columns')
 y = df.target
```

```
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)
```

```
from sklearn.ensemble import RandomForestClassifier
 model = RandomForestClassifier(n_estimators=20)
 model.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=20)
```

```
model.score(X_test, y_test)
```

```
0.9805555555555555
```

```
y_predicted = model.predict(X_test)
```

Confusion Matrix

```
from sklearn.metrics import confusion_matrix
 cm = confusion_matrix(y_test, y_predicted)
 cm
```

```
array([[32,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 30,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 32,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 37,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 35,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 41,  1,  0,  0,  1],
       [ 0,  0,  0,  0,  1,  0, 35,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 52,  0,  2],
       [ 1,  0,  0,  0,  0,  0,  0,  0, 32,  0],
       [ 0,  0,  0,  0,  1,  0,  0,  0,  0, 27]])
```

```
%matplotlib inline
 import matplotlib.pyplot as plt
 import seaborn as sn
 plt.figure(figsize=(10,7))
 sn.heatmap(cm, annot=True)
 plt.xlabel('Predicted')
 plt.ylabel('Truth')
```

```
Text(95.72222222222221, 0.5, 'Truth')
```