NAME: Meith Navlakha
SAP: 60004190068
DWM Practical

EXPERIMENT 2 [PART B & C]

AIM:

Implementation of Classification algorithm using:

1. Decision Tree ID3
2. Naïve Bayes algorithm

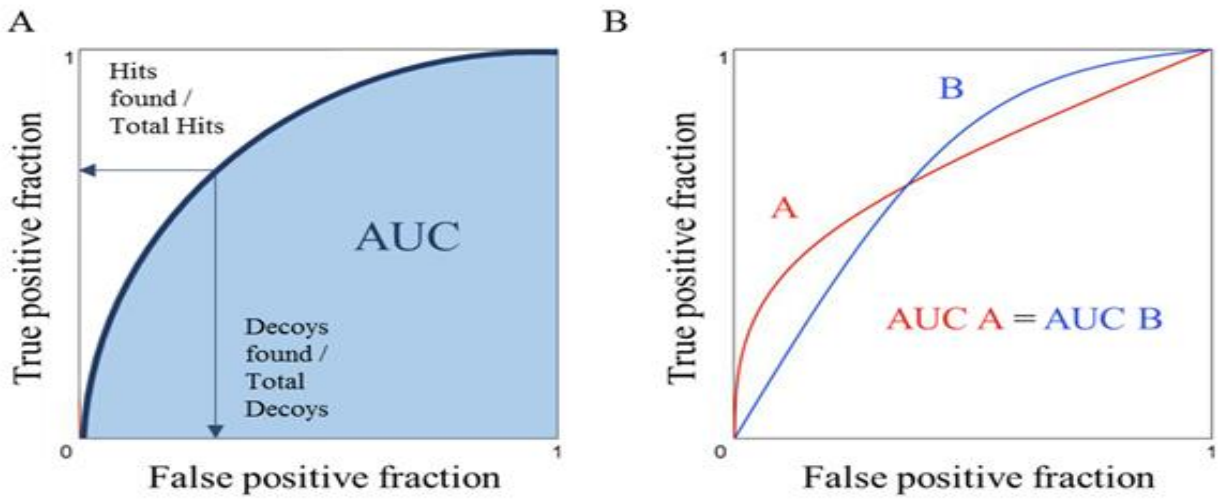
THEORY:

ROC AUC Curve

The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the 'signal' from the 'noise'. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

When AUC = 1, then the classifier is able to perfectly distinguish between all the Positive and the Negative class correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives. When 0.5<AUC<1, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values.



NAME: Meith Navlakha
SAP: 60004190068
DWM Practical

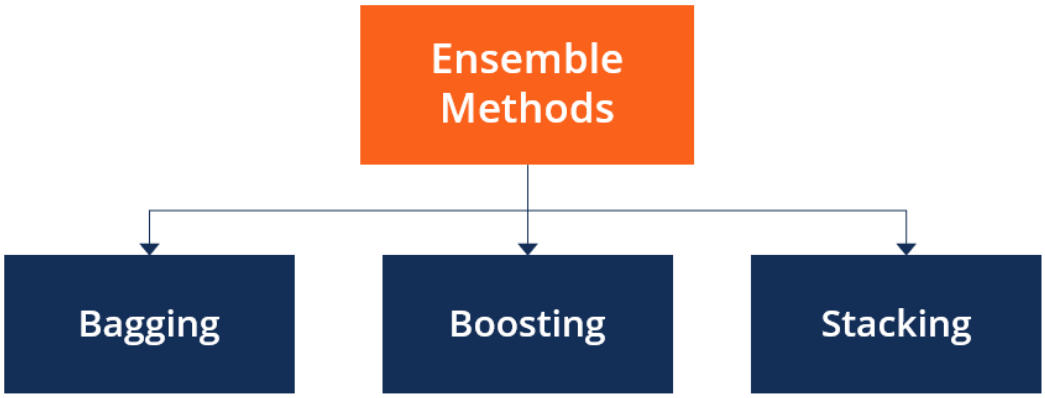
Cross Validation using K-Fold

K Fold cross validation; the data is divided into k subsets. Now the holdout method is repeated k times, such that each time, one of the k subsets is used as the validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get total effectiveness of our model. Every data point gets to be in a validation set exactly once, and gets to be in a training set k-1 times. This significantly reduces bias as we are using most of the data for fitting, and also significantly reduces variance as most of the data is also being used in validation set. Interchanging the training and test sets also adds to the effectiveness of this method. Mostly, K = 5 or 10 is preferred, but nothing's fixed and it can take any value.

Cross Validation is a very useful technique for assessing the effectiveness of your model, particularly in cases where you need to mitigate overfitting.

Ensemble Methods

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model. Ensemble learning is primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unfortunate selection. Other applications of ensemble learning include assigning a confidence to the decision made by the model, selecting optimal (or near optimal) features, data fusion, incremental learning, nonstationary learning and error-correcting.



NAME: Meith Navlakha
SAP: 60004190068

CODE:

PART B:

1) Plot ROC AUC Curve

[NOTE : ROC AUC is only plotted for Breast Cancer Dataset as all other dataset is multi-labelled dataset.]

```
def plot_auc(fpr_nb , tpr_nb, fpr_dt , tpr_dt , name ):
  fig , ax = plt.subplots()
  fig.set_size_inches(8,8)

  ax.plot(fpr_nb , tpr_nb , linewidth=5)
  ax.plot(fpr_dt , tpr_dt , linewidth=5)
  ax.plot([0,1] , [0,1] ,ls='--' , color='black' , lw=0.5)
  ax.legend(['NB', 'DT'], loc='upper right')
  ax.set(Xlabel='FPR' , ylabel = 'TPR' , xlim=[-0.01 , 1.01] ,
   ylim=[-0.01 , 1.01] , title='ROC Curve BREAST CANCER')
  ax.grid(True)
  plt.show()

// BREAST CANCER
from sklearn.datasets import load_breast_cancer
X, y = load_breast_cancer(return_X_y=True)
acc_nb_3 , acc_dt_3 , fpr_nb , tpr_nb, fpr_dt , tpr_dt = trainModel(X, y ,
'BREAST CANCER')
plot_auc(fpr_nb , tpr_nb, fpr_dt , tpr_dt ,'BREAST CANCER' )
```

2) Comparison

```
# from sklearn import sns
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(models_pd.iloc[:,0])
plt.plot(models_pd.iloc[:,1])

plt.subplots_adjust(left=0.0, right=1.0, bottom=0.0, top=1.0)
plt.title('NB vs DT Accuracy')
plt.ylabel('accuracy')
plt.xlabel('Datasets')
plt.legend(['NB', 'DT'], loc='upper right')
plt.show()
```

NAME: Meith Navlakha
SAP: 60004190068

PART C:

1) K Fold Cross Validation

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split

def plot_KFold(nb_scores , dt_scores ,name):
  # fig , axList = plt.subplots(ncols=2)
  fig , ax = plt.subplots()
  fig.set_size_inches(8,8)

  ax.plot( nb_scores , linewidth=5)
  ax.plot(dt_scores , linewidth=5)
  ax.legend(['NB', 'DT'], loc='upper right')
  title = 'K-FOLDS ' + name
  ax.set(Xlabel='K FOLDS' , ylabel = 'ACCURACY' , title=title)
  ax.grid(True)
  plt.show()


# LOAD DIGIT
from sklearn.datasets import load_digits
X, y = load_digits(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3 , random_state=5
)

nb = GaussianNB()
dt = tree.DecisionTreeClassifier()


# K-FOLD
k_fold = KFold( n_splits=10, shuffle=True, random_state=0)

cv= 10
nb_scores = cross_val_score(nb, X_train , y_train, cv=k_fold, scoring='accuracy')
nb_scores = nb_scores

dt_scores = cross_val_score(dt, X_train , y_train, cv=k_fold , scoring='accuracy')
cross_results = pd.DataFrame( pd.concat([pd.DataFrame(nb_scores) , pd.DataFrame(d
t_scores)] , ignore_index=True ,axis=1) )


plot_KFold(nb_scores , dt_scores , ' LOAD DIGITS ')
```

```
# OLIVETTI FACES
from sklearn.datasets import fetch_olivetti_faces
X, y = fetch_olivetti_faces(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3 , random_state=5
)

nb = GaussianNB()
dt = tree.DecisionTreeClassifier()

# K-FOLD
# kf = KFold(n_splits=3)
k_fold = KFold( n_splits=10, shuffle=True, random_state=0)

cv= 10
nb_scores = cross_val_score(nb, X_train , y_train, cv=k_fold, scoring='accuracy')
nb_scores = nb_scores

dt_scores = cross_val_score(dt, X_train , y_train, cv=k_fold , scoring='accuracy')
cross_results = pd.DataFrame( pd.concat([pd.DataFrame(nb_scores) , pd.DataFrame(d
t_scores)] , ignore_index=True ,axis=1) )

plot_KFold(nb_scores , dt_scores , 'OLIVETTI FACES')
```

2) Ensemble models

```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier


def ensemble(X, y, name , i):
  cv=10
  # ADABOOST
  ada = AdaBoostClassifier(n_estimators=100)
  acc_ada = cross_val_score(ada, X, y, cv=cv)
  # GRADIENT BOOST
  gradboost = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,
     max_depth=1, random_state=0).fit(X, y)
  acc_gb = cross_val_score(gradboost, X, y, cv=cv)
  # RANDOM FOREST
  rf = RandomForestClassifier(n_estimators=10, max_depth=None,
     min_samples_split=2, random_state=0)
  acc_rf = cross_val_score(rf, X, y, cv=cv)
```

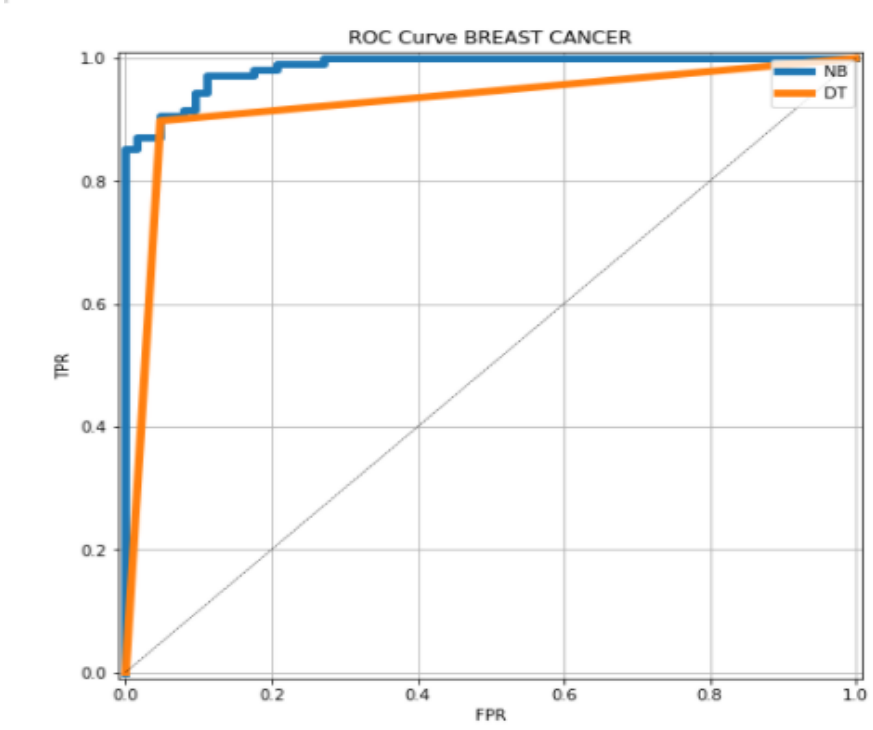
```
# PLOT
  fig , ax = plt.subplots()
  fig.set_size_inches(8,8)

  ax.plot(acc_ada , linewidth=3)
  ax.plot(acc_gb , linewidth=3)
  ax.plot(acc_rf , linewidth=3)

  ax.legend(['ADABOOST', 'GRADIENT BOOST' , 'RANDOM FOREST' ], loc='upp
er right')

  title = 'ENSEMBLE ACCURACY ' + name
  ax.set(Xlabel='SAMPLES' , ylabel = 'ACCURACY' , title= title )

  ax.grid(True)
  plt.show()


# DATASET

from sklearn.datasets import load_digits


# LOAD DIGITS
X, y = load_digits(return_X_y=True)
ensemble(X,y, 'LOAD DIGITS' ,0)


# IRIS
from sklearn.datasets import load_iris
X, y = load_iris(return_X_y=True)
cv= 10
ensemble(X,y, 'IRIS' ,1)
```

NAME: Meith Navlakha
SAP: 60004190068
DWM Practical

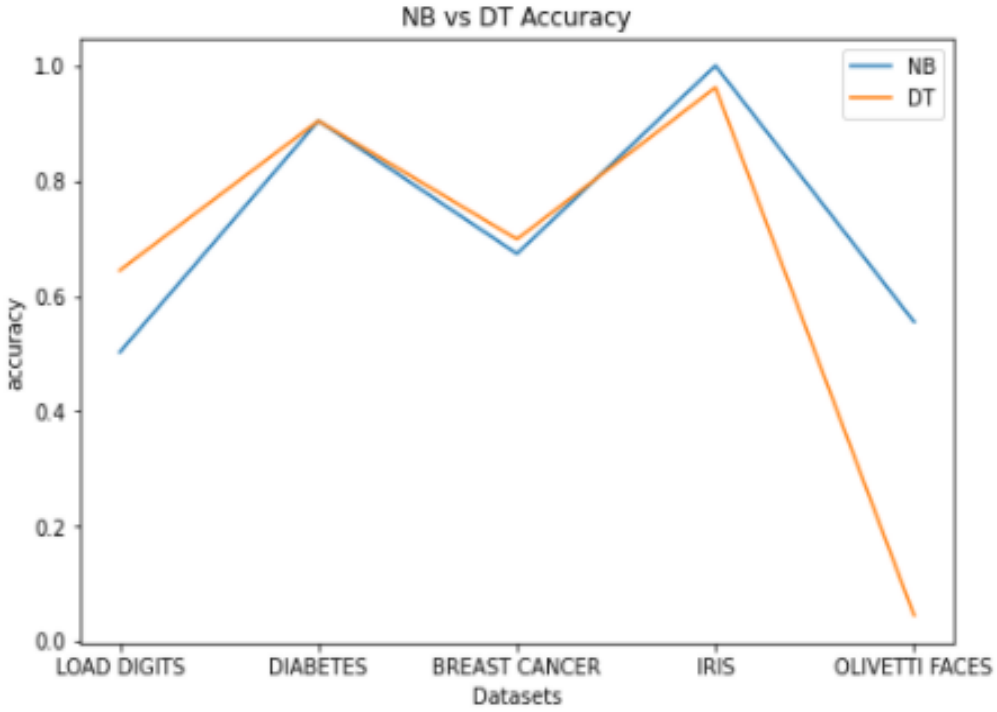
OUTPUT:

PART B:

1) **ROC AUC CURVE**



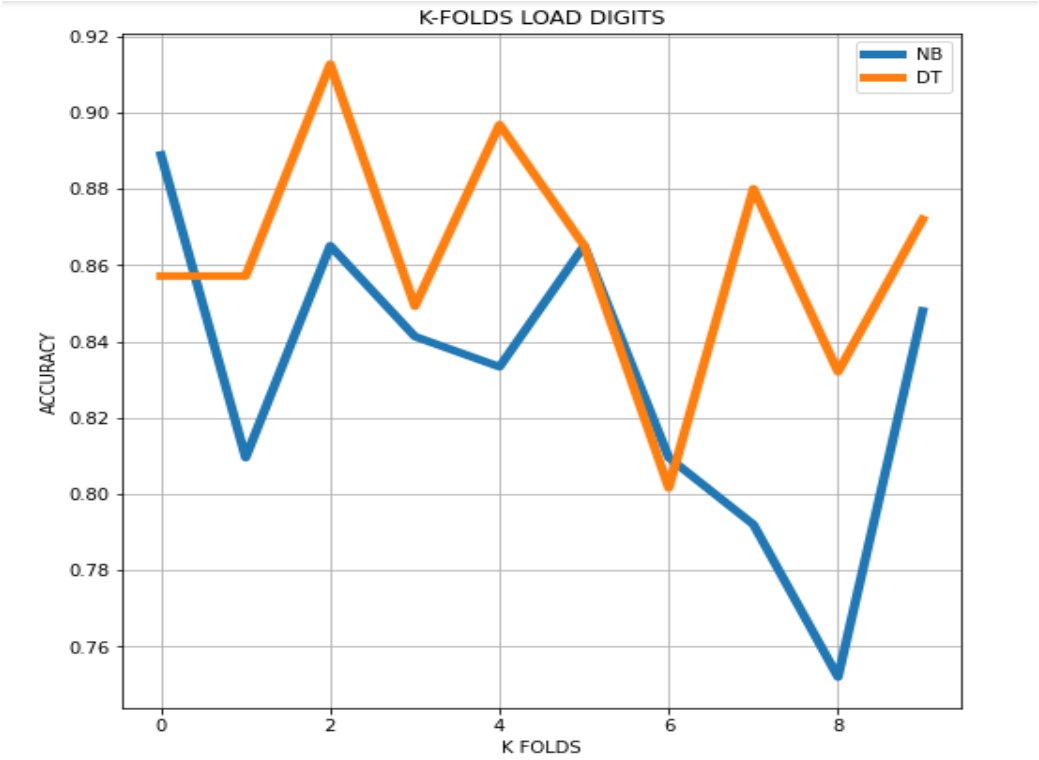
2) **Comparison of Results**

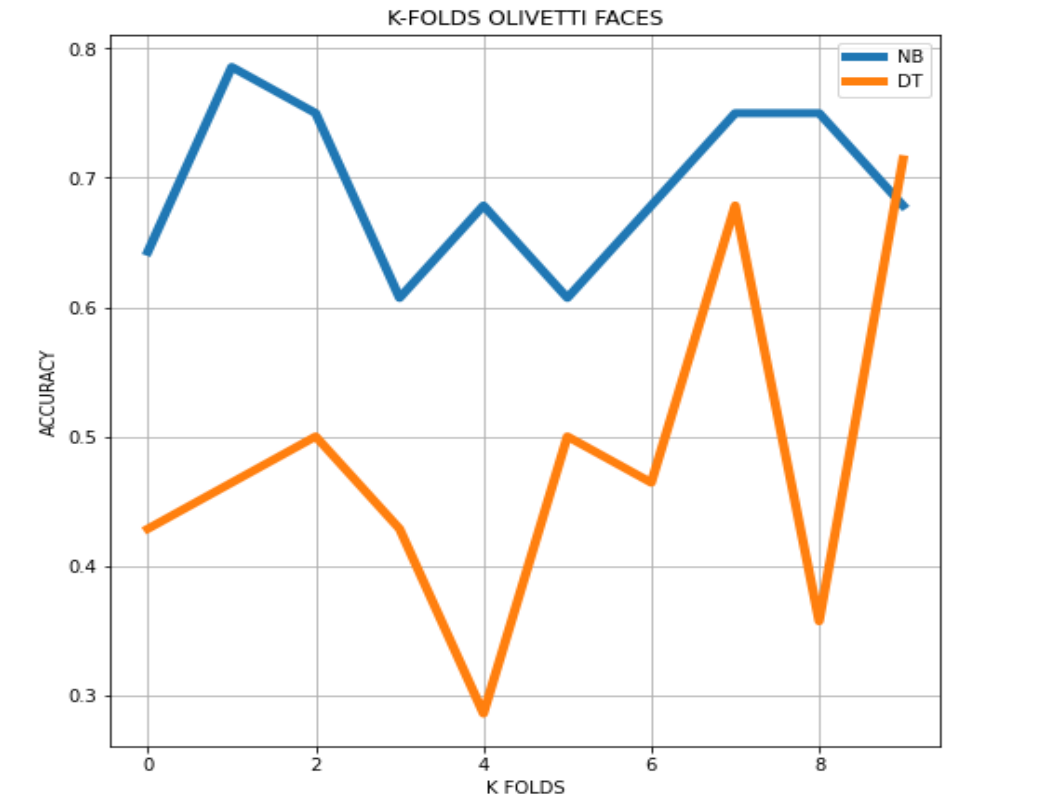


NAME: Meith Navlakha
SAP: 60004190068
DWM Practical

PART C:

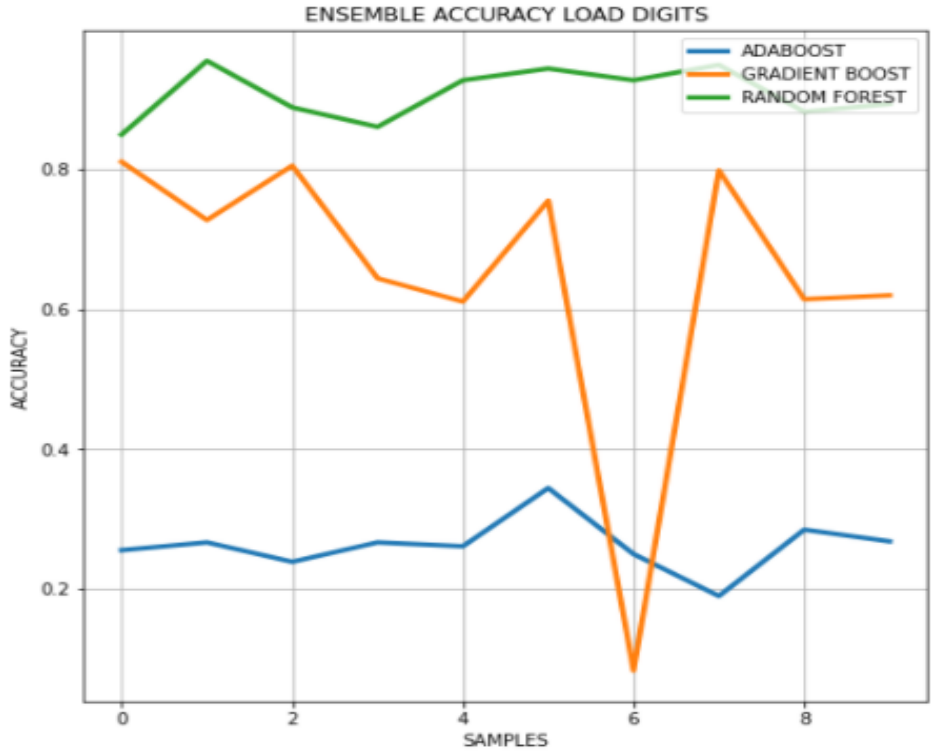
1) K-Fold Cross Validation

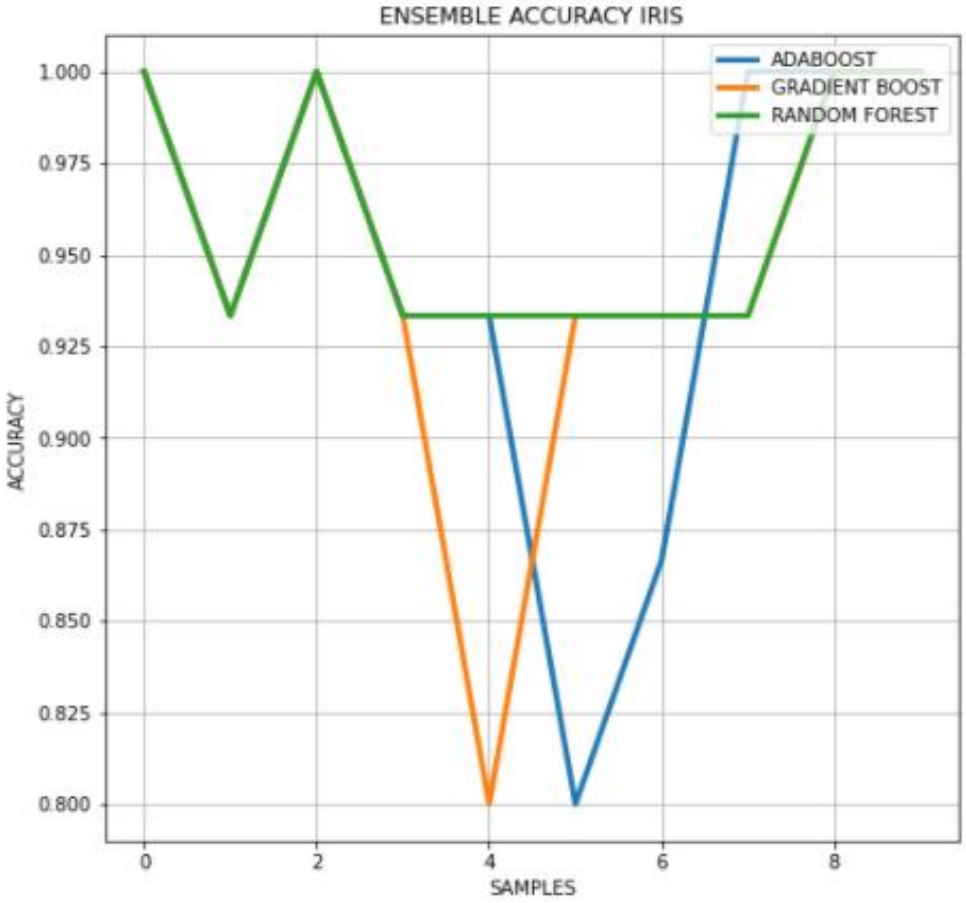




NAME: Meith Navlakha
SAP: 60004190068
DWM Practical

2) Ensemble models





NAME: Meith Navlakha
SAP: 60004190068
DWM Practical

CONCLUSION: In this experiment, I plotted a line chart for the comparison of accuracy of 5 datasets for the Naïve Bayes and Decision Tree models. Iris dataset has high accuracy of 1.00 and 0.96 for Naïve Bayes and Decision Tree respectively. Diabetes, Breast Cancer and Load digits have commendable accuracy between 0.6 to 0.9, Olivetti Faces dataset has extremely low accuracy (around 0.132) for Decision Tree. Doing K-Fold(K=10) Cross Validation on Load Digits produced the best results (accuracy=0.912 and 0.865) for K=2 for Decision Tree and Naïve Bayes respectively. Thus, using K-Fold Cross Validation increased the accuracy of Load Digits from 0.50 to 0.865 and 0.62 to 0.912 for Naïve Bayes and Decision Tree respectively. The accuracy of Olivetti Faces Dataset increased immensely using K-Fold Cross Validation 0.786(K=1) and 0.714(K=9) for Naïve Bayes and Decision Tree respectively. Using, Ensemble on Load Digits gives best results for Random Forest. Gradient Boost on Load Digits falls sharply at number of samples is 6 and AdaBoost produces low results throughout.