**EXPERIMENT 2 [PART B & C]**

**AIM:**

Implementation of Classification algorithm using:

1. Decision Tree ID3
2. Naïve Bayes algorithm

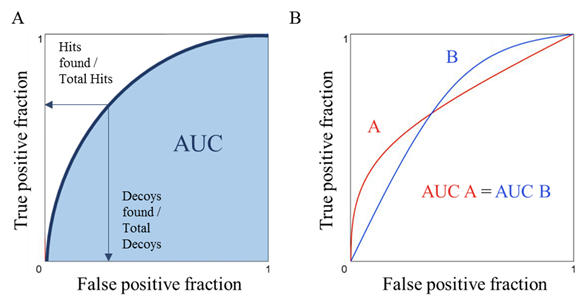
**THEORY:**

**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.



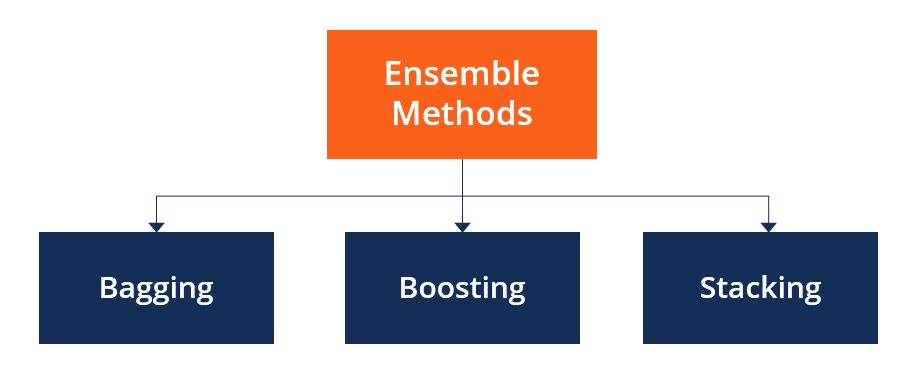
**Cross Validation using K-Fold**

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.



**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' )

1. **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()

**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(dt\_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(dt\_scores)] , ignore\_index=True ,axis=1) )

plot\_KFold(nb\_scores , dt\_scores , 'OLIVETTI FACES')

1. **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='upper 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)

**OUTPUT:**

**PART B:**

1. **ROC AUC CURVE**

Chart, line chart

Description automatically generated

1. **Comparison of Results**

Chart, line chart

Description automatically generated

**PART C:**

**1) K-Fold Cross Validation**

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

**2) Ensemble models**

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

**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.