**CSC6515 – Machine Learning for Big Data**  
**Assignment 1**

**Task a - Decision Tree Classifier Code**

*The code is self-explanatory, please read the comments on code*

*# -\*- coding: utf-8 -\*-*

*"""*

*Created on Sun Oct 16 17:47:58 2016*

*@author: Yamuna*

*"""*

import pandas as pd

import os

from sklearn.cross\_validation import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

import sklearn.metrics

*# Opening path where dataset is located*

os.chdir("C:\\Users\\Yamuna\\Desktop\\Big Data\\")

*# Read dataset*

in\_data = pd.read\_csv("satellite.csv")

*# Cleaning dataset*

data\_clean = in\_data.dropna()

data\_clean.dtypes

data\_clean.describe()

*# Defining predictors*

predictors = data\_clean[['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z',

'AA','AB','AC','AD','AE','AF','AG','AH','AI','AJ']]

*# Defining target data*

targets = data\_clean.AK

*# Dividing train and test data*

train\_data, test\_data, target\_train, target\_test = train\_test\_split(predictors,targets,test\_size=.3)

*# Shaping data*

train\_data.shape

test\_data.shape

target\_train.shape

target\_test.shape

*# Initialising Decision tree classifier*

classifier=DecisionTreeClassifier()

classifier=classifier.fit(train\_data,target\_train)

*# Predicting test and train data*

test\_predictions=classifier.predict(test\_data)

train\_predictions=classifier.predict(train\_data)

*#Train data confusion matrix and accuracy score*

train\_confu\_mat = sklearn.metrics.confusion\_matrix(target\_train, train\_predictions)

train\_accu\_score = sklearn.metrics.accuracy\_score(target\_train, train\_predictions)

*#Test data confusion matrix and accuracy score*

test\_confu\_mat = sklearn.metrics.confusion\_matrix(target\_test, test\_predictions)

test\_accu\_score = sklearn.metrics.accuracy\_score(target\_test, test\_predictions)

*#Visualising Decision Tree*

from sklearn import tree

from io import BytesIO as StringIO

from IPython.display import Image

import pydotplus

out = StringIO()

tree.export\_graphviz(classifier, out\_file = out)

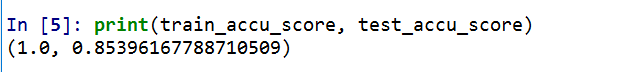
graph=pydotplus.graph\_from\_dot\_data(out.getvalue())

Image(graph.create\_png())

graph.write\_pdf("C:\\Users\\Yamuna\\Desktop\\Big Data\\tree.pdf")

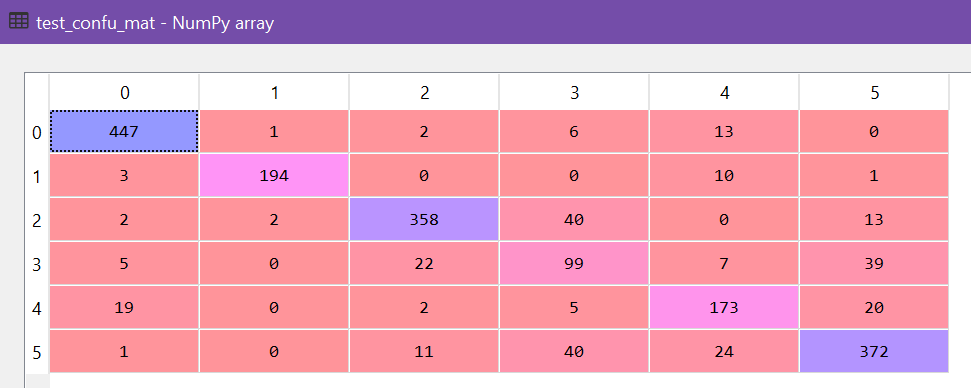
**Output and Explanation**

**Accuracy scores**



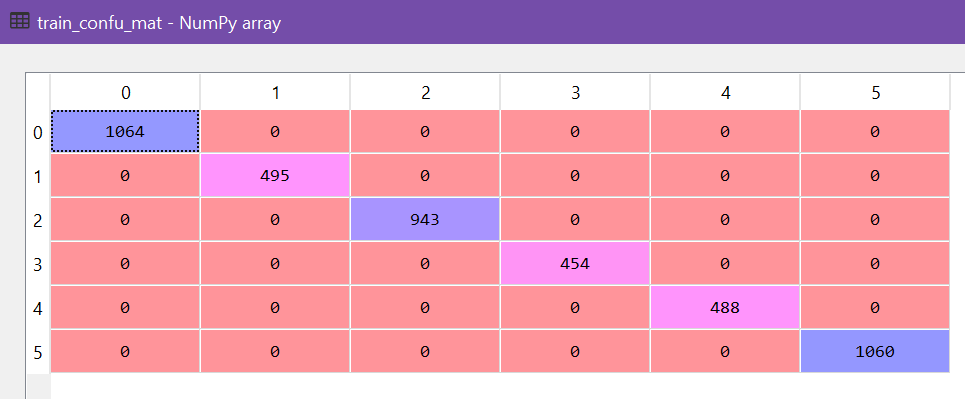
The train accuracy score is 100% and is larger than the test accuracy score (85% approx.). This is because, classifier was trained using the train data and hence the classification done on train data is accurate. The test data is slightly different and so the accuracy is not 100%

**Test data Confusion Matrix**



This matrix above shows the Actual class along the Y axis and the Predicted class along the X axis. In the above confusion matrix of test data, 447 predictions are correctly predicted under class 0. Three predictions are done wrongly as class 0 which actually belongs to class 1 and so on. The predictions that are diagonal are the correctly predicted values for that class.

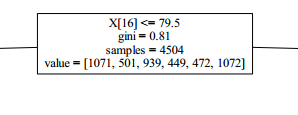
**Train data Confusion Matrix**



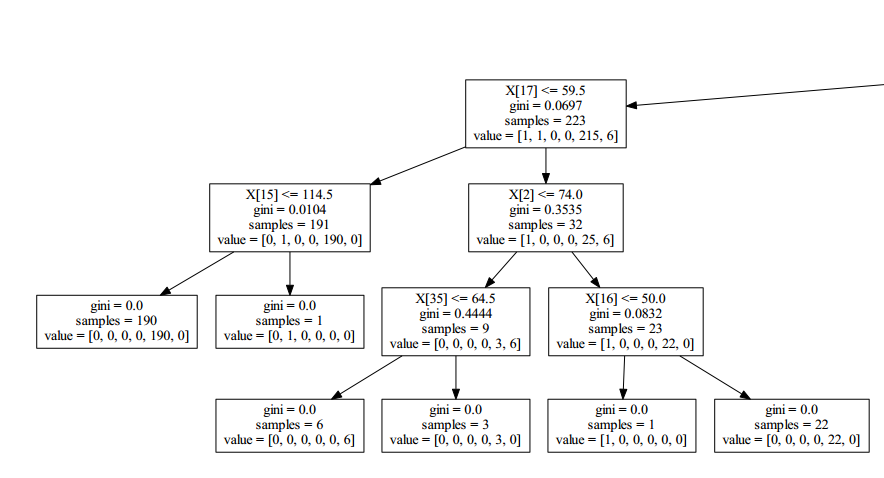
The confusion matrix for train data is very accurate and none of the classes are wrongly predicted. Since, the decision tree classifier was trained with the train data. And so is the accuracy.

**Tree Visualization – Tree.pdf is attached with this submission which has the entire decision tree**

**Root**



**Sample Branch**



**Task b – Naïve Bayes and Random Forest classifier code**

# -\*- coding: utf-8 -\*-

"""

Created on Sun Oct 23 01:29:00 2016

@author: Yamuna

"""

import pandas as pd

import numpy as np

import os

import sklearn.metrics

from sklearn.ensemble import RandomForestClassifier

from sklearn import cross\_validation

from sklearn.naive\_bayes import GaussianNB

os.chdir("C:\\Users\\Yamuna\\Desktop\\Big Data\\")

*#importing given dataset*

in\_data = pd.read\_csv("satellite.csv")

*# Initialising 10 fold cross validation*

kf\_total = cross\_validation.KFold(len(in\_data), n\_folds=10, shuffle=False, random\_state=None)

*# Initialising Random Forest Classifier*

rf\_classifier=RandomForestClassifier(n\_estimators=10,

max\_features='auto',

max\_depth=None,

min\_samples\_split=2,

random\_state=0)

*# Initialising Naive Bayes Classifier*

nb\_classifier = GaussianNB()

*# Calculating accuracy scoreof naive bayes and Random forest for 10 folds*

nb\_accu\_score=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

rf\_accu\_score=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

loop=0

*# Looping for each fold*

for train\_index, test\_index in kf\_total:

*# Training the Naive Bayes Classifier for each fold with train data and train result*

nb\_classifier.fit(np.array(in\_data.ix[train\_index[0]:train\_index[len(train\_index)-1],:36]), np.array(in\_data.ix[train\_index[0]:train\_index[len(train\_index)-1],36:37]))

*# Predicting the test data using the trained Naive Bayes Classifier*

nb\_predicted = nb\_classifier.predict(np.array(in\_data.ix[test\_index[0]:test\_index[len(test\_index)-1],:36]))

*# Calculating accuracy of Naive Bayes Classifier for each fold and storing it in an array*

nb\_accu\_score[loop] = sklearn.metrics.accuracy\_score(nb\_predicted,np.array(in\_data.ix[test\_index[0]:test\_index[len(test\_index)-1],36:37]))

*# Training the Random Forest Classifier for each fold with train data and train result*

rf\_classifier.fit(np.array(in\_data.ix[train\_index[0]:train\_index[len(train\_index)-1],:36]), (np.array(in\_data.ix[train\_index[0]:train\_index[len(train\_index)-1],36:37])).ravel())

*# Predicting the test data using the trained Random Forest Classifier*

rf\_predicted=rf\_classifier.predict(np.array(in\_data.ix[test\_index[0]:test\_index[len(test\_index)-1],:36]))

*# Calculating accuracy of Random Forest Classifier for each fold and storing it in an array*

rf\_accu\_score[loop] = sklearn.metrics.accuracy\_score(rf\_predicted,np.array(in\_data.ix[test\_index[0]:test\_index[len(test\_index)-1],36:37]))

loop=loop+1

*# Calculating Mean accuracy for NaiveBayes Classifier*

NB\_Mean\_accu = np.mean(nb\_accu\_score)

*# Calculating Mean Accuracy for Random Forest Classifier*

RF\_Mean\_accu = np.mean(rf\_accu\_score)

*# Calculating Standard deviation of NaiveBayes classifier accuracy*

NB\_std\_dev=np.std(nb\_accu\_score, dtype=np.float64)

*# Calculating Standard deviation of Random Forest Classifier accuracy*

RF\_std\_dev=np.std(rf\_accu\_score, dtype=np.float64)

from scipy import stats

*# Given alpha value in question*

alpha=0.05

*# Calculating statistical test for Mean and standard deviation of accuracy scores in NaiveBayes and Random Forest*

t, p = stats.ttest\_ind\_from\_stats(NB\_Mean\_accu, NB\_std\_dev, 10,

RF\_Mean\_accu, RF\_std\_dev, 10,

equal\_var=False)

print("ttest\_ind\_from\_stats: t = %g p = %g" % (t, p))

*# Determinig the Statistical significance using ttest and alpha*

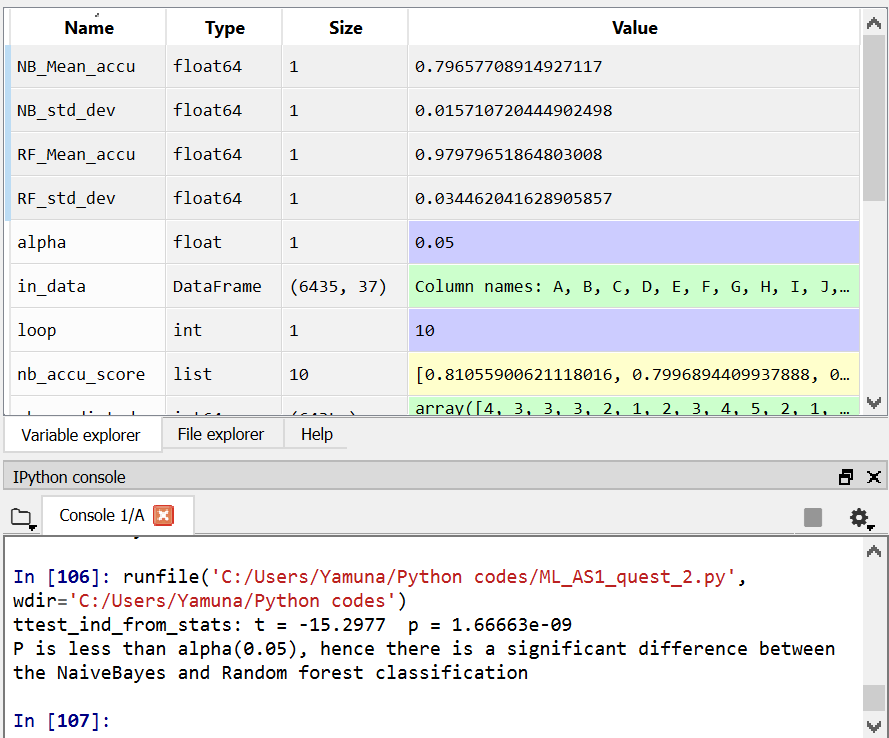
if (p<alpha):

print("P is less than alpha(0.05), hence there is a significant difference between the NaiveBayes and Random forest classification")

else:

print("There is no significant difference between the NaiveBayes and Random forest classification")

**Output and Explanation**



The value of ‘p’ from the statistical significance test is 1.66663e -09 and is less than the alpha value (0.05).

If p<0.05, then the observations are having statistically significant difference and

if p>0.05, then the observations are having statistically no significant difference.

Based on the output, Naïve Bayes and Random forest classifier are having statistically significant difference.

**Naïve Bayes vs Random Forest 10 - Fold accuracy values**

