Assignment1-G5

**Consider the attached wine dataset from the UCI machine learning repository:**

**https://archive.ics.uci.edu/ml/datasets/Wine. The data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.**

**All attributes are continuous. NOTE: 1st attribute is class identifier (1-3).**

**Use 70% of the data in each class for training a Naive Bayes classifier and the remaining 30% to test the classifier performance based on given 13 attributes as:**

**1) Alcohol 2) Malic acid 3) Ash 4) Alcalinity of ash 5) Magnesium 6) Total phenols 7) Flavanoids 8) Nonflavanoid phenols 9) Proanthocyanins 10)Color intensity 11)Hue 12)OD280/OD315 of diluted wines 13)Proline**

**Provide the confusion matrix, sensitivity, specificity, total accuracy, F1-score, Roc curve, and area under curve.**

***Solution:***

import os

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import ConfusionMatrixDisplay

from sklearn.preprocessing import label\_binarize

from sklearn.metrics import roc\_curve, auc

from itertools import cycle

from sklearn.metrics import classification\_report

# Load dataset file

dataset = pd.read\_csv('/Users/furqaankhan/Desktop/wine.data', sep = ',')

# Split the dataset into train and test sets

training\_set, test\_set = train\_test\_split(dataset, test\_size = 0.3, train\_size = 0.7, random\_state = 42)

# Separate the Label from the Features

training\_data = training\_set.iloc[:,1:]

training\_target = training\_set.iloc[:,0]

test\_data = test\_set.iloc[:,1:]

test\_target = test\_set.iloc[:,0]

# Run Naive Bayes classifier algorithm

clf\_gnb = GaussianNB()

# Fit

clf\_gnb.fit(training\_data, training\_target)

# Predict

preds\_gnb = clf\_gnb.predict(test\_data)

# Confusion matrix

print('Confusion Matrix : \n', confusion\_matrix(test\_target, preds\_gnb))

ConfusionMatrixDisplay.from\_predictions(test\_target, preds\_gnb)

plt.show()

# Calculate classification report

target\_names = ['Class 1', 'Class 2', 'Class 3']

print(classification\_report(test\_target, preds\_gnb, target\_names=target\_names))

binary\_labels = label\_binarize(test\_target, classes=[1, 2, 3])

n\_classes = binary\_labels.shape[1]

score = clf\_gnb.fit(training\_data, training\_target).predict\_proba(test\_data)

fpr = dict()

tpr = dict()

roc\_auc = dict()

for i in range(n\_classes):

fpr[i], tpr[i], \_ = roc\_curve(binary\_labels[:, i], score[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

colors = cycle(['blue', 'red', 'green'])

for i, color in zip(range(n\_classes), colors):

plt.plot(fpr[i], tpr[i], color=color, linewidth=2,

label='ROC curve of class {0} (area = {1:0.2f})'

''.format(i, roc\_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', linewidth=2)

plt.xlim([-0.05, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.legend(loc="lower right")

plt.show()