Farida Aliyeva

Homework 4

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

##importing neural network

from sklearn.neural\_network import MLPClassifier

##importing Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier

##importing Support Vector Machine

from sklearn.svm import SVC

## metrics to print out the matrix, F1, recall, precision and accuracy after training the data

from sklearn.metrics import classification\_report

from sklearn import metrics

##imports for extra task (shuffler and model selector fo)

from sklearn.utils import shuffle as shuffler

from sklearn import model\_selection

These are the imports of all necessary libraries used throughout the assignment

##loading the data by getting rid of ; and # using the sep function of pandas library

train\_data = pd.read\_csv(r'C:\Users\ASUS\Desktop\data\Train.txt', header=None, sep=';|#', engine='python')

cross\_valid\_data = pd.read\_csv(r'C:\Users\ASUS\Desktop\data\Cross\_Validation.txt',header=None, sep=';|#', engine='python')

test\_data = pd.read\_csv(r'C:\Users\ASUS\Desktop\data\Test.txt',header=None, sep=';|#', engine='python')

Here is the way of loading the data into the program by getting rid of ; and # signs and also giving the data a view of table

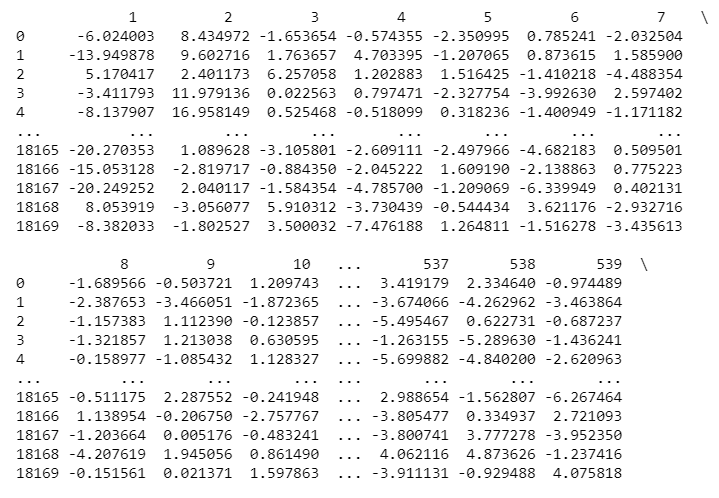
## Train data visualization

train\_X = train\_data.iloc[:,1:]

train\_Y = train\_data.iloc[:,0]

print(train\_X,train\_Y)

Output:

Taking into consideration a big scale of the output the screenshot was cropped.

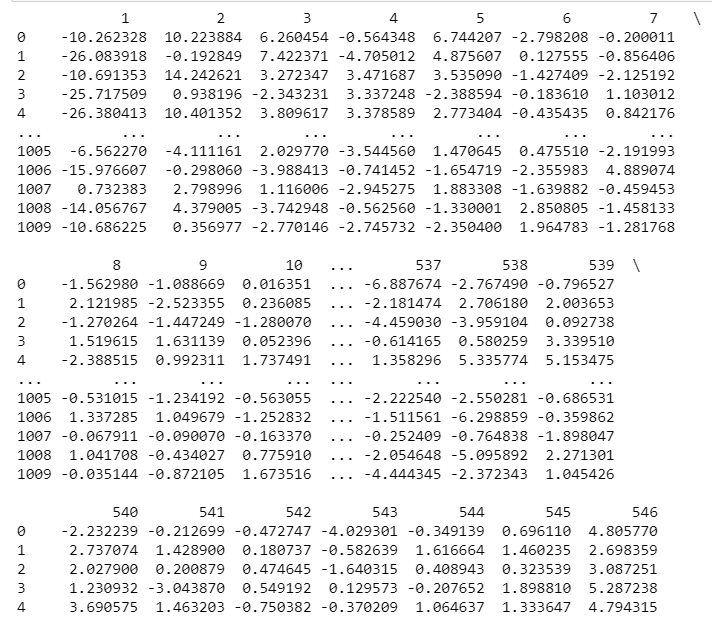
## Cross Validation data visualization

crosv\_X=cross\_valid\_data.iloc[:,1:]

crosv\_Y=cross\_valid\_data.iloc[:,0]

print(crosv\_X,crosv\_Y)

Output:

Also cropped

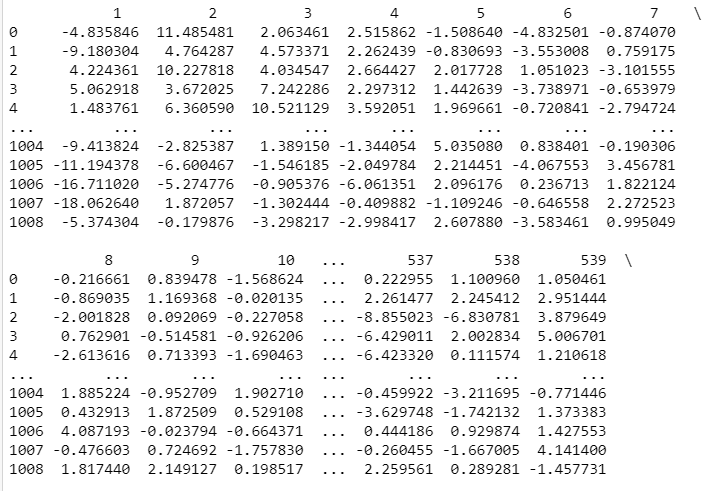
##Test data visualization

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

test\_Y=test\_data.iloc[:,0]

print(test\_X,test\_Y)

Output:

Again cropped

Now starts the NEURAL NETWORK

## Training the model by Neural Network

mc = MLPClassifier(hidden\_layer\_sizes=(80,80,80),activation='relu', solver='adam',max\_iter=500)

mc.fit(train\_X, train\_Y)

NN\_prediction1 = mc.predict(crosv\_X)

print("Accuracy:",metrics.accuracy\_score(crosv\_Y,NN\_prediction1))

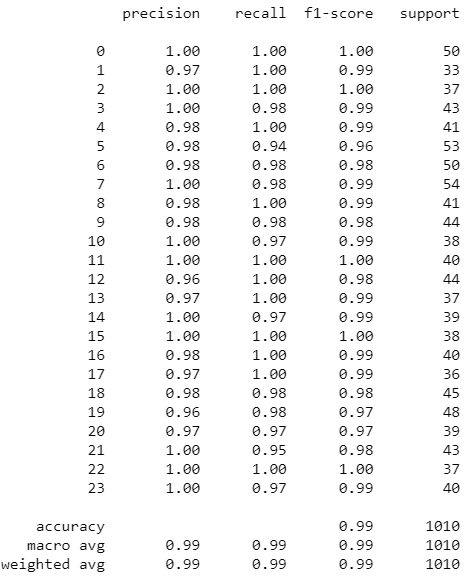
I used a separate code to print accuracy while classification report rounds up the values and I wanted to show the whole value of accuracy.

Output:



print(classification\_report(crosv\_Y,NN\_prediction1))

Output:



NN\_prediction2=mc.predict(test\_X)

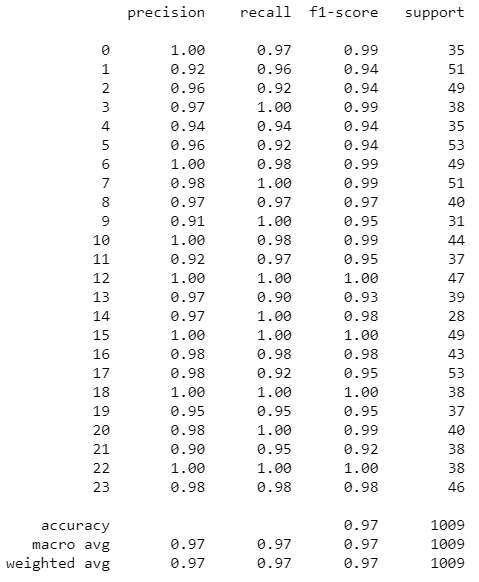
print("Accuracy:",metrics.accuracy\_score(test\_Y,NN\_prediction2))

Output:



print(classification\_report(test\_Y,NN\_prediction2))

Output:



Now starts the Support Vector Classification.

## Training the model by SVM

clsf = SVC(kernel='rbf',gamma= 'scale' , C=1)

## tried several parameters kernel and gamma (gamma scale produced the highest accuracy value)

clsf.fit(train\_X, train\_Y)

SVM\_predicted =clsf.predict(test\_X)

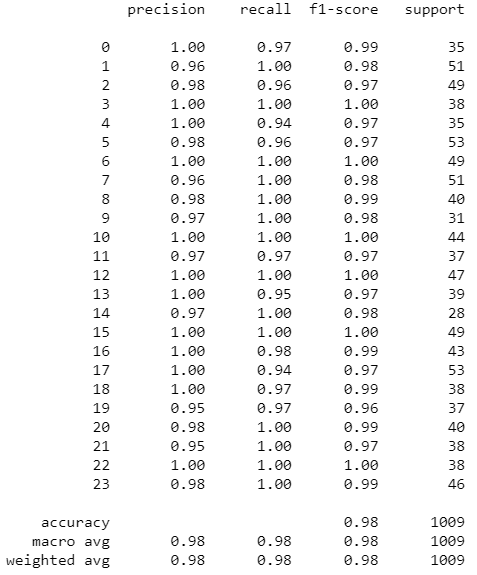
print("Accuracy:",metrics.accuracy\_score(test\_Y, SVM\_predicted))

Output:



print(classification\_report(test\_Y,SVM\_predicted))

Output:



SVM\_predicted2=clsf.predict(crosv\_X)

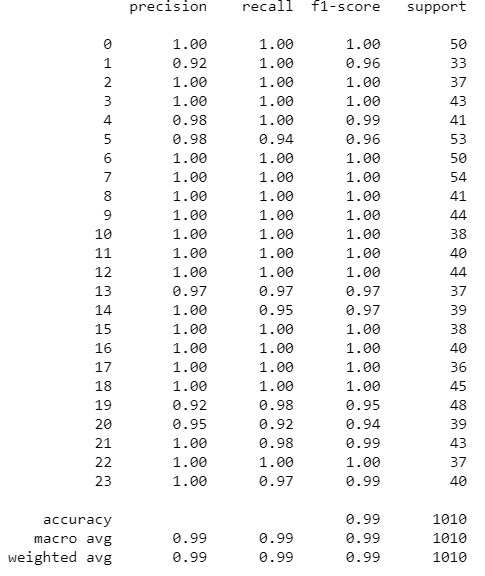
print("Accuracy:",metrics.accuracy\_score(crosv\_Y,SVM\_predicted2))

Output:



print(classification\_report(crosv\_Y,SVM\_predicted2))

Output:



Now starts the Decision Tree Classifier part.

## Training the model by Decision Tree Classifier

dtree = DecisionTreeClassifier(criterion="entropy", splitter = "best")

dtree.fit(train\_X, train\_Y)

DTC\_prediction = dtree.predict(crosv\_X)

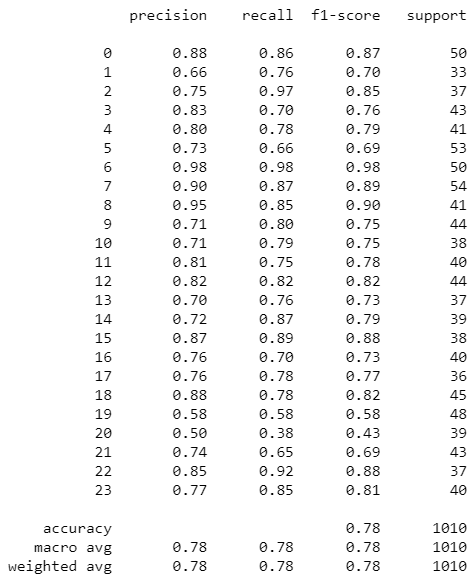
print("Accuracy:",metrics.accuracy\_score(crosv\_Y,DTC\_prediction))

Output:



print(classification\_report(crosv\_Y,DTC\_prediction))

Output:



DTC\_prediction2=dtree.predict(test\_X)

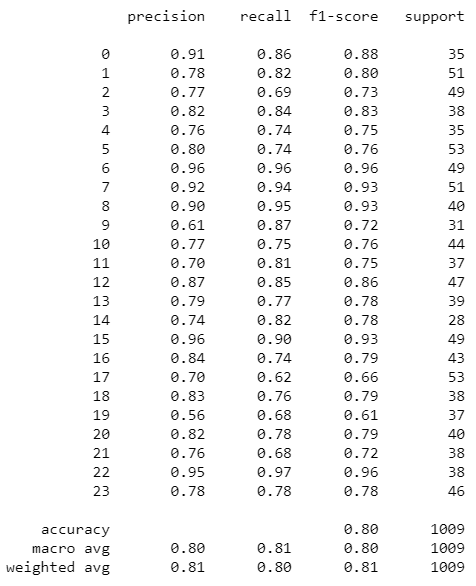
print("Accuracy:",metrics.accuracy\_score(test\_Y,DTC\_prediction2))

Output:



print(classification\_report(test\_Y,DTC\_prediction2))

Output:



From the following outputs of 3 different algorithms we can see that Decision Tree Classification is not suitable for this kind of data providing us only 0.80 and 0.78 Accuracy. On the other hand, we notice that SVM (Support Vector Machine) is the most suitable giving us the Accuracy of 0.98 and 0.99. Lastly, the Neural Network also produces high Accuracy results: 0.99 and 0.97. To sum up, SVM (Support Vector Machine) seems to be more appropriate to train model of this type of data that we use here.

**Extra task:**

For extra task I used kFold validation model which splits the data. The number of splits I gave is 10 folds as it was asked. Then it will be tested with Neural Network, SVM and DTC and the average accuracy scores will be printed.

## EXTRA TASK

result = pd.concat([train\_data,cross\_valid\_data,test\_data])

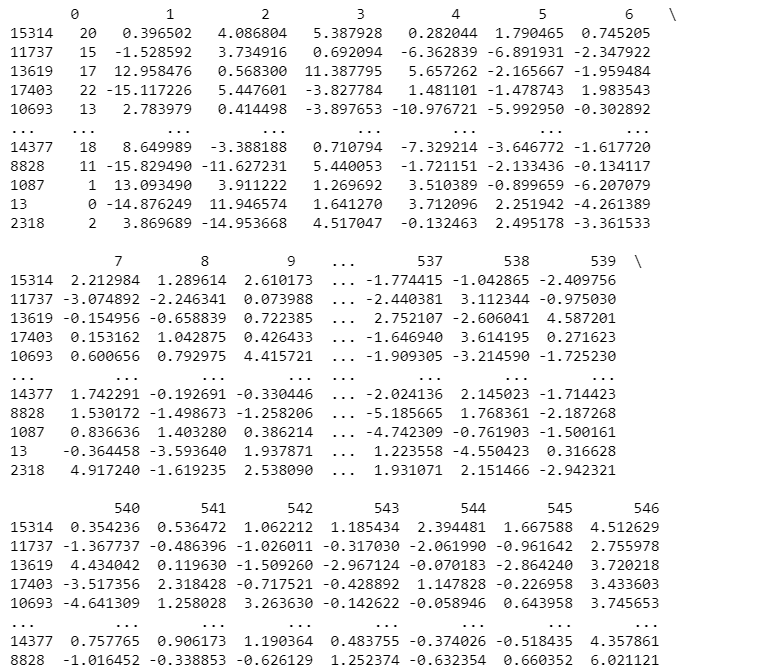
result = shuffler(result)

print(result)

x = result.iloc[:,1:]

y = result.iloc[:,0]

Output:

Shuffled data

kf = model\_selection.KFold(n\_splits=10, random\_state=None)

cv\_res = model\_selection.cross\_val\_score(MLPClassifier(), x, y, cv=kf)

print("Accuracy: " , (cv\_res.mean()))

Output:

Result for Neural Network



cv\_res1 = model\_selection.cross\_val\_score(DecisionTreeClassifier(), x, y, cv=kf)

print("Accuracy: " , (cv\_res1.mean()))

Output:

Result for Decision Tree Classifier



cv\_res2 = model\_selection.cross\_val\_score(SVC(), x, y, cv=kf)

print("Accuracy: " , (cv\_res2.mean()))

Output:

Result for Support Vector Machine

