CODE:

1) Code7.py

```
########################################
#    Harikrishna Prabhu            ##
#    3333077042                  ##
########################################
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import random

from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score


op=input("Enter the option for # feature:\n 1.first 2 or\n 2.13(all)\n")

#############################################################################
###.    data initialization
##Training data
X_TRAIN=np.genfromtxt("wine_train.csv", delimiter=',')
class_Label_train=X_TRAIN[:,13]
if(op==1):
        given_Data_TRAIN=X_TRAIN[:,0:2]
else:
        given_Data_TRAIN=X_TRAIN[:,0:13]

##Test Data
X_TEST=np.genfromtxt("wine_test.csv", delimiter=',')
class_Label_test=X_TEST[:,13]
if(op==1):
        given_Data_TEST=X_TEST[:,0:2]
else:
        given_Data_TEST=X_TEST[:,0:13]

'''
print("Before Standardization\nTRAIN data:")
print(given_Data_TRAIN)
print('Mean: ', np.mean(given_Data_TRAIN, axis=0))
print('Std: ', np.std(given_Data_TRAIN, axis=0))
'''
```

```
################################################################################
######. data standardization

#standardize data
scaler = StandardScaler() #initializes a StandardScaler object
scaler.fit(given_Data_TRAIN) # (x-mu)/sigma

#Standardizing Train and Test Data
given_Data_TRAIN_std = scaler.transform(given_Data_TRAIN)
given_Data_TEST_std = scaler.transform(given_Data_TEST)


'''
print('After standardization:')
print(given_Data_TRAIN_std)
print('Mean: ', np.mean(given_Data_TRAIN_std, axis=0))
print('Std: ', np.std(given_Data_TRAIN_std, axis=0))


print("Before Standardization\nTEST data:")
print(given_Data_TEST)
print('Mean: ', np.mean(given_Data_TEST, axis=0))
print('Std: ', np.std(given_Data_TEST, axis=0))
# to check the labels present in test
print("Uniquely Identified Labels in the TEST data set:")
print("Unique labels: {0}".format(np.unique(class_Label_test))),;print("\n\n\n")
print('Test Data Standardized:')
print(given_Data_TEST_std)
print('Mean: ', np.mean(given_Data_TEST_std, axis=0))
print('Std: ', np.std(given_Data_TEST_std, axis=0))
'''
################################################################################
######. Perceptron Implementation

#a=np.array([[1.0,1.0],[1.0,1.0],[1.0,1.0]]) # weight vectors hardcoded default
#b=np.array([1.0, 1.0, 1.0])
max_test1=0
max_test2=0
max_train=0
count_test1=0
count_test2=0
count_train=0
if(op==1):
        max_w=np.zeros((3,13))
```

```
        max_w2=np.zeros((3,13))
else:
        max_w=np.zeros((3,2))
        max_w2=np.zeros((3,2))
#
#
po=input("Select Perceptron parameters: \n1. Default or \n2. Include random initial weight
vector.\n")
if(po==1):  ##for default parameters
        model=Perceptron()
        model.fit(given_Data_TRAIN_std,class_Label_train)
        test_pred=model.predict(given_Data_TEST_std)
        train_pred=model.predict(given_Data_TRAIN_std)

        accuracy_test=accuracy_score(class_Label_test,test_pred)*100
        accuracy_train=accuracy_score(class_Label_train,train_pred)*100
else:
        for i in range(0,100):
                # weight vectors
                #a=np.array([[0.1,0.1],[0.1,0.1],[0.1,0.1]])
                if(op==1):
                        c=np.random.rand(3,2) # weight vector for 2 Feature case
                else:
                        c=np.random.rand(3,13) # weight vectors for 13 Features case
                '''c=np.array([  [0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1],
                                        [0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1],
                                        [0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1]
])'''
                #b=np.array([1, 1, 1]) #augmented weight vector
                #b=np.random.rand(1,3)
                model=Perceptron(max_iter=10000)

        model.fit(given_Data_TRAIN_std,class_Label_train,coef_init=c)#,intercept_init=b)


                test_pred=model.predict(given_Data_TEST_std)
                train_pred=model.predict(given_Data_TRAIN_std)

                accuracy_test=accuracy_score(class_Label_test,test_pred)*100
                accuracy_train=accuracy_score(class_Label_train,train_pred)*100

                if(max_train<=accuracy_train):
                        max_train=accuracy_train
                        count_train=i
```

```
                    max_w=np.copy(c)
                    max_test1=accuracy_test
                    count_test1=i

                #if(count_train==100):

                if(max_test2<accuracy_test):
                    max_test2=accuracy_test
                    count_test2=i
                    max_w2=np.copy(c)

if(po==1):
    print("Using Default Perceptron Parameters:")
    if(op!=1):
        print("13 Features data set has same accuracy rate for any random weight
vector. So we have taken the weight that gets max accuracy in test data set ::")
        print("TEST
accuracy:"),;print("#"),;print(accuracy_test),;print("%"),;print("\nTrain
accuracy:"),;print(accuracy_train),;print("%")
        print("weight_coef:")
        print(model.coef_)
    else:
        print("2 Features data set has same accuracy rate for any random weight vector.
So we have taken the weight that gets max accuracy in test data set ::")
        print("TEST
accuracy:"),;print("#"),;print(accuracy_test),;print("%"),;print("\nTrain
accuracy:"),;print(accuracy_train),;print("%")
        print("weight_coef:")
        print(model.coef_)
else:
    print("Using inintial weight parameter in Perceptron Parameters:")
    if(op!=1):
        print("13 Features data set has same accuracy rate for any random weight
vector. So we have taken the weight that gets max accuracy in test data set ::")
        print("TEST accuracy:"),;print("#"),;print(max_test1),;print("%"),;print("\nTrain
accuracy:"),;print(max_train),;print("%")
        print("weight_coef:")
        print(max_w)
    else:
        print("2 Feature data set is taken. The weight corresponding to max accuracy of
the data set is:")
        print("at iteration I:"),;print(count_test2)
        print("TEST:"),;print("#"),;print(max_test2),;print("train"),;print(max_train)
        print("weight_coef:")
```

```
                print(max_w)




#print(model.coef_)
```

(2) Code8.py

```
##########################################
#    Harikrishna Prabhu            ##
#    3333077042                  ##
##########################################
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import random
import math

from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
from sklearn.multiclass import OneVsRestClassifier

class MSE_binary(LinearRegression):
        def __init__(self):
                #print("Calling newly created MSE_binary function...")
                super(MSE_binary, self).__init__()
        def predict(self, X):
                thr = 0.5
                y = self._decision_function(X)
                y_int = (np.zeros(y.shape)).astype(int)
                y_int[y>thr] = 1
                return y_int
#class labels
```

```
op=input("Enter the option for # feature:\n 1.first 2 or\n 2.13(all)\n")

#############################################################################
###.    data initialization
##Training data
X_TRAIN=np.genfromtxt("wine_train.csv", delimiter=',')
class_Label_train=X_TRAIN[:,13]
if(op==1):
        given_Data_TRAIN=X_TRAIN[:,0:2]
else:
        given_Data_TRAIN=X_TRAIN[:,0:13]

##Test Data
X_TEST=np.genfromtxt("wine_test.csv", delimiter=',')
class_Label_test=X_TEST[:,13]
if(op==1):
        given_Data_TEST=X_TEST[:,0:2]
else:
        given_Data_TEST=X_TEST[:,0:13]

#standardize data
scaler = StandardScaler() #initializes a StandardScaler object
scaler.fit(given_Data_TRAIN) # (x-mu)/sigma

#Standardizing Train and Test Data
given_Data_TRAIN_std = scaler.transform(given_Data_TRAIN)
given_Data_TEST_std = scaler.transform(given_Data_TEST)

 ######
po=input("Want to check (1)non-normalized data or (2)Normalized data \n ")
if(po==1):
        binary_model = MSE_binary()
        #model = LinearRegression()
        model = OneVsRestClassifier(binary_model)
        model.fit(given_Data_TRAIN,class_Label_train)
        test_pred=model.predict(given_Data_TEST)
        train_pred=model.predict(given_Data_TRAIN)
else:
        binary_model = MSE_binary()
        model = OneVsRestClassifier(binary_model)
        model.fit(given_Data_TRAIN_std,class_Label_train)
        test_pred=model.predict(given_Data_TEST_std)
        train_pred=model.predict(given_Data_TRAIN_std)
```

```
#print(test_pred)
'''
length=len(test_pred)
print(length)
test_pred_bin=np.zeros(length)
for i in range(0,length):
        test_pred_bin[i]=round(test_pred[i])
print(test_pred_bin)
'''
accuracy_test=accuracy_score(class_Label_test,test_pred)*100
print("TEST accuracy:"),;print(accuracy_test),;print("%")

accuracy_train=accuracy_score(class_Label_train,train_pred)*100
print("TRAIN accuracy:"),;print(accuracy_train),;print("%")
```