1) Perceptron Classifier

```
import sklearn
import csv
import numpy as np
import math
import matplotlib.pyplot as plt
import copy
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score

#Open the csv file and store it in a list
with open("D:\python3_files\python3\wine_train.csv","r") as Feat_Train:
    Feat_Train_Reader = csv.reader(Feat_Train, delimiter=',')
    FeatureList = []
    for row in Feat_Train_Reader:
        if len (row) != 0:
            FeatureList = FeatureList +[row]

Feat_Train.close()

#Make a copy of the training data list with just the training data and no labels
FeatureListDec = copy.deepcopy(FeatureList)
for row in FeatureListDec:
    del row[-1]

#Make an array of the training data
Train_data = np.array(FeatureListDec).astype("float")
```

```
#Make a list containing the class labels
Class = []
for row in FeatureList:
    Class.append(row[-1])

Class_Train = np.array(Class).astype("float")

#Find mean and standard deviation
print('Before standardization:')
print(Train_data)
print('Mean : ', np.mean(Train_data, axis=0))
print('Std : ', np.std(Train_data, axis =0))

#Normalised Training data
scaler = StandardScaler()
scaler.fit(Train_data)
Train_data_norm = scaler.transform(Train_data)
print('After standardization:')
print(Train_data_norm)

#Convert to Two features
a = np.zeros(len(Train_data_norm))
b = np.zeros(len(Train_data_norm))
for i in range(len(Train_data_norm)):
    a[i] = Train_data_norm[i][0]
    b[i] = Train_data_norm[i][1]

Train_data_2feat_norm = np.column_stack([a,b])
```

```
#Apply Perceptron for two features
model = Perceptron(n_iter=1000, tol=0.0001, random_state = None)
model.fit(Train_data_2feat_norm, Class_Train)
print('Final Weights:')
print(model.coef_)
label_train_pred = model.predict(Train_data_2feat_norm)
print(label_train_pred)

acc_train = accuracy_score(Class_Train, label_train_pred)
print(acc_train)

#Load test data
with open("D:\python3_files\python3\wine_test.csv","r") as Feat_Test:
    Feat_Test_Reader = csv.reader(Feat_Test, delimiter=',')
    FeatureTestList = []
    for row in Feat_Test_Reader:
        if len (row) != 0:
            FeatureTestList = FeatureTestList +[row]

Feat_Test.close()

#Make a copy of the training data list with just the training data and no labels
FeatureTestListDec = copy.deepcopy(FeatureTestList)
for row in FeatureTestListDec:
    del row[-1]

#Make an array of the test data
Test_data = np.array(FeatureTestListDec).astype("float")
```

```
#Make a list containing the class labels
Class1 = []
for row in FeatureTestList:
    Class1.append(row[-1])

Class_Test = np.array(Class1).astype("float")

Test_data_norm = scaler.transform(Test_data)

#Convert to two features
c = np.zeros(len(Test_data_norm))
d = np.zeros(len(Test_data_norm))
for i in range(len(Test_data_norm)):
    c[i] = Test_data_norm[i][0]
    d[i] = Test_data_norm[i][1]

Test_data_2feat_norm = np.column_stack([c,d])

#Apply Perceptron to test
label_test_pred = model.predict(Test_data_2feat_norm)
print(label_test_pred)

acc_test = accuracy_score(Class_Test, label_test_pred)
print(acc_test)

#Perceptron when all features for train
model1 = Perceptron(n_iter=1000, tol=0.0001, random_state = None)
model1.fit(Train_data_norm, Class_Train)
print('Final Weights:')
print(model1.coef_)
label1_train_pred = model1.predict(Train_data_norm)
print(label1_train_pred)

acc_train1 = accuracy_score(Class_Train, label1_train_pred)
print(acc_train1)

#Perceptron when all features for test
label1_test_pred = model1.predict(Test_data_norm)
print(label1_test_pred)

acc1_test = accuracy_score(Class_Test, label1_test_pred)
print(acc1_test)

#part d for problem 2 for two features - train and test data set
Weight1 = np.zeros((3,2))
TrainAcc = 0
Sample = np.zeros((3,2))
Init_Weight = np.zeros((3,2))
for i in range(100):
    model2 = Perceptron(n_iter=1000, tol=0.0001, random_state = None)
    Sample = np.random.randn(3,2)
    model2.fit(Train_data_2feat_norm, Class_Train, coef_init = Sample)
    label1_train_pred = model2.predict(Train_data_2feat_norm)
    acc_train1 = accuracy_score(Class_Train, label1_train_pred)
    if(acc_train1 > TrainAcc):
        TrainAcc = acc_train1
        Weight1 = model2.coef_
        Init_Weight = Sample
```

```
print(Weight1)

model2.fit(Train_data_2feat_norm, Class_Train, coef_init = Init_Weight)
print(TrainAcc)

label_test_pred = model2.predict(Test_data_2feat_norm)
acc_test = accuracy_score(Class_Test, label_test_pred)
print(acc_test)

#part d for problem 2 for all features - train and test data set
Weight = np.zeros((3,13))
TrainAcc = 0
Sample = np.zeros((3,13))
Init_Weight = np.zeros((3,13))
for i in range(100):
  model = Perceptron(n_iter=1000, tol=0.0001, random_state = None)
  Sample = np.random.randn(3,13)
  model.fit(Train_data_norm, Class_Train, coef_init = Sample)
  label1_train_pred = model.predict(Train_data_norm)
  acc_train1 = accuracy_score(Class_Train, label1_train_pred)
  if(acc_train1 > TrainAcc):
    TrainAcc = acc_train1
    Weight = model.coef_
    Init_Weight = Sample

print(Weight)

model.fit(Train_data_norm, Class_Train, coef_init = Init_Weight)
```

```
print(TrainAcc)


label_test_pred = model.predict(Test_data_norm)

acc_test = accuracy_score(Class_Test, label_test_pred)

print(acc_test)
```

2) MSE Classifier

```
import sklearn

import csv

import numpy as np

import math

import matplotlib.pyplot as plt

import copy

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score

from sklearn.linear_model import LinearRegression

from sklearn.multiclass import OneVsRestClassifier


#Open the csv file and store it in a list

with open("D:\python3_files\python3\wine_train.csv","r") as Feat_Train:

  Feat_Train_Reader = csv.reader(Feat_Train, delimiter=',')

  FeatureList = []

  for row in Feat_Train_Reader:

    if len (row) != 0:

      FeatureList = FeatureList +[row]


Feat_Train.close()


#Make a copy of the training data list with just the training data and no labels
```

```
FeatureListDec = copy.deepcopy(FeatureList)
for row in FeatureListDec:
    del row[-1]

#Make an array of the training data
Train_data = np.array(FeatureListDec).astype("float")

#Convert to Two features
a = np.zeros(len(Train_data))
b = np.zeros(len(Train_data))
for i in range(len(Train_data)):
    a[i] = Train_data[i][0]
    b[i] = Train_data[i][1]

Train_data_2feat = np.column_stack([a,b])

#Make a list containing the class labels
Class = []
for row in FeatureList:
    Class.append(row[-1])

Class_Train = np.array(Class).astype("float")

#Find mean and standard deviation
print('Before standardization:')
print(Train_data)
print('Mean : ', np.mean(Train_data, axis=0))
print('Std : ', np.std(Train_data, axis =0))
```

```
#Normalised Training data
scaler = StandardScaler()
scaler.fit(Train_data)
Train_data_norm = scaler.transform(Train_data)
print('After standardization:')
print(Train_data_norm)

#Convert to Two features
a = np.zeros(len(Train_data_norm))
b = np.zeros(len(Train_data_norm))
for i in range(len(Train_data_norm)):
    a[i] = Train_data_norm[i][0]
    b[i] = Train_data_norm[i][1]

Train_data_2feat_norm = np.column_stack([a,b])

#Define Class MSE Binary
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(len(X))
        for i in range(len(X)):
            if(y[i]<thr):
                y_int[i] = 0
            else:
                y_int[i] = 1
        return y_int

#Load test data
with open("D:\python3_files\python3\wine_test.csv","r") as Feat_Test:
    Feat_Test_Reader = csv.reader(Feat_Test, delimiter=',')
    FeatureTestList = []
    for row in Feat_Test_Reader:
        if len (row) != 0:
            FeatureTestList = FeatureTestList +[row]

Feat_Test.close()

#Make a copy of the training data list with just the training data and no labels
FeatureTestListDec = copy.deepcopy(FeatureTestList)
for row in FeatureTestListDec:
    del row[-1]

#Make an array of the test data
Test_data = np.array(FeatureTestListDec).astype("float")

#Convert to Two features
a = np.zeros(len(Test_data))
b = np.zeros(len(Test_data))
for i in range(len(Test_data)):
    a[i] = Test_data[i][0]
    b[i] = Test_data[i][1]

Test_data_2feat = np.column_stack([a,b])
```

```
#Make a list containing the class labels
Class1 = []
for row in FeatureTestList:
    Class1.append(row[-1])

Class_Test = np.array(Class1).astype("float")

Test_data_norm = scaler.transform(Test_data)

#Convert to two features
c = np.zeros(len(Test_data_norm))
d = np.zeros(len(Test_data_norm))
for i in range(len(Test_data_norm)):
    c[i] = Test_data_norm[i][0]
    d[i] = Test_data_norm[i][1]

Test_data_2feat_norm = np.column_stack([c,d])

#Apply to training data without normalization
binary_model = MSE_binary()
model = OneVsRestClassifier(binary_model)
model.fit(Train_data, Class_Train)
print('Final Weights:')
print(model.coef_)
label_train_pred = model.predict(Train_data)
print(label_train_pred)

acc_train = accuracy_score(Class_Train, label_train_pred)
```

```
print(acc_train)

#Apply to test data without normalization
label_test_pred = model.predict(Test_data)
print(label_test_pred)

acc_test = accuracy_score(Class_Test, label_test_pred)
print(acc_test)

#Apply to training data with two features without normalization
binary_model = MSE_binary()
model = OneVsRestClassifier(binary_model)
model.fit(Train_data_2feat, Class_Train)
print('Final Weights:')
print(model.coef_)
label_train_pred = model.predict(Train_data_2feat)
print(label_train_pred)

acc_train = accuracy_score(Class_Train, label_train_pred)
print(acc_train)

#Apply to test data with two features without normalization
label_test_pred = model.predict(Test_data_2feat)
print(label_test_pred)

acc_test = accuracy_score(Class_Test, label_test_pred)
print(acc_test)

#Apply to training data with normalization
```

```
binary_model = MSE_binary()
model = OneVsRestClassifier(binary_model)
model.fit(Train_data_norm, Class_Train)
print('Final Weights:')
print(model.coef_)
label_train_pred = model.predict(Train_data_norm)
print(label_train_pred)

acc_train = accuracy_score(Class_Train, label_train_pred)
print(acc_train)

#Apply to test data with normalization
label_test_pred = model.predict(Test_data_norm)
print(label_test_pred)

acc_test = accuracy_score(Class_Test, label_test_pred)
print(acc_test)

#Apply to training data with two features with normalization
binary_model = MSE_binary()
model = OneVsRestClassifier(binary_model)
model.fit(Train_data_2feat_norm, Class_Train)
print('Final Weights:')
print(model.coef_)
label_train_pred = model.predict(Train_data_2feat_norm)
print(label_train_pred)

acc_train = accuracy_score(Class_Train, label_train_pred)
print(acc_train)
```

```
#Apply to test data with two features with normalization
label_test_pred = model.predict(Test_data_2feat_norm)
print(label_test_pred)

acc_test = accuracy_score(Class_Test, label_test_pred)
print(acc_test)
```