```
# wine.py
import numpy as np
from python3.plotDecBoundaries import plotDecBoundaries
from tools.nearest_centroid_classifier import *



train_data = np.genfromtxt('python3/wine_train.csv',  delimiter = ',')
test_data =  np.genfromtxt('python3/wine_test.csv' ,  delimiter = ',')

train_data_labels = []
test_data_labels = []

train_data_labels= getLabels(train_data)
test_data_labels= getLabels(test_data)
print(train_data_labels)
#print(train_data_labels)
train_error_set_for_features = []
test_error_set_for_features = []
feature1 = 0
feature2 = 1
mean_set_label1 = train_classifer_label(train_data,1,feature1,feature2)
mean_set_label2 = train_classifer_label(train_data,2,feature1,feature2)
mean_set_label3 = train_classifer_label(train_data,3,feature1,feature2)
sample_mean_set_unlabelled =
np.array([mean_set_label1,mean_set_label2,mean_set_label3])
mean_set_label1 = np.append(mean_set_label1,1)
mean_set_label2 = np.append(mean_set_label2,2)
mean_set_label3 = np.append(mean_set_label3,3)
sample_mean_set_labelled =
np.array( [mean_set_label1,mean_set_label2,mean_set_label3])
estimate_label_set=
nearest_classifier(train_data,sample_mean_set_labelled,feature1,feature2)
TrainDataErrorRate = computeErrorRate(estimate_label_set,train_data_labels)
plotDecBoundaries(train_data[:,[feature1,feature2]],train_data_labels,sampl
e_mean_set_unlabelled)
minErrorRate = 100000000000000000000
minFeature1 = 0
minFeature2 = 0
ErrorRateSet = []
minUnlabelledSampleMean = []
for i in range(13):
    for j in range(i+1,13):
        ErrorRate,sample_mean_set_unlabelled , sample_mean_set_labelled =
```

```
searchFeature(train_data,test_data, i , j)
        ErrorRateSet.append(ErrorRate)
        if minErrorRate > ErrorRate:
            minErrorRate = ErrorRate
            minFeature1 = i
            minFeature2 = j
            minUnlabelledSampleMean = sample_mean_set_unlabelled

errormean = 0
sumerror = 0
for error in ErrorRateSet:
    sumerror += error
variance = 0
errormean = sumerror/len(ErrorRateSet)
for error in ErrorRateSet:
    variance += (error-errormean)*(error-errormean)

print("The most suitable feature")
print(minFeature1, minFeature2)
print(min(ErrorRateSet))
print(errormean)
print(variance)
plotDecBoundaries(train_data[:,[minFeature1,minFeature2]],train_data_labels
,minUnlabelledSampleMean)
'''
for i in range(len(train_data)-1):
    for j in range(i+1,len(train_data)-1):
        mean_set =  []
        train_mean_set_nearest_classifier(train_data,mean_set,i,j)
        train_error_set_for_features.append(computeErrorRate())
'''
```

wine2.py

```
import numpy as np
from python3.plotDecBoundaries import plotDecBoundaries
from tools.nearest_centroid_classifier import *

train_data = np.genfromtxt('python3/wine_train.csv',  delimiter = ',')
test_data =  np.genfromtxt('python3/wine_test.csv' ,  delimiter = ',')

feature1 = 0
feature2 = 1
train_data_labels= getLabels(train_data)
```

```
test_data_labels= getLabels(test_data)


TrainDataErrorRate,sample_mean_set_unlabelled , sample_mean_set_labelled =
searchFeature(train_data,test_data, feature1 , feature2)

estimatedTestDataLabel =
nearest_classifier(test_data,sample_mean_set_labelled,0,1)
computeErrorRate(test_data_labels,estimatedTestDataLabel)
```

nearest_centroid_classifier.py

```
import numpy as np
from math import *


def __init__(self):
    print("This nearest_centroid_classifier")


def computeVectorEuclideanDistance(v1 ,v2):
    sum = 0
    difference =np.array(v1) -np.array(v2)
    for feature in difference:
        sum += feature*feature

    return sqrt(sum)

def nearest_centroid_classifier(test_data,
sample_mean_set,estimate_label_set):
    for data in test_data:
        minClass =  0
        minDistance = 99
        for mean in sample_mean_set:
            mean_feature_set = mean[:, [0,1]]
            if
computeVectorEuclideanDistance(mean_feature_set,data)<minDistance :
                minDistance =
computeVectorEuclideanDistance(mean_feature_set,data)
                minClass = mean[2]
        estimate_label_set.append(minClass)

def nearest_classifier(dataset , sample_mean_set, f1,f2):
```

```
    estimated_label_set = []
    for data in dataset :
        specificData = [data[f1],data[f2]]
        minDistance = 10000000000000000000000000000000000000000000
        minClass  = 0
        for mean in sample_mean_set:
            specificMean = [mean[0],mean[1]]
            if
minDistance>computeVectorEuclideanDistance(specificData,specificMean):
                minClass = mean[2]
                minDistance =
computeVectorEuclideanDistance(specificMean,specificData)
        estimated_label_set.append(minClass)
    return estimated_label_set
def computeErrorRateForDataSet(dataset,sample_mean_set, f1,f2):
    estimated_label_set =
nearest_centroid_classifier(dataset,sample_mean_set,f1,f2)


def computeErrorRate(labelset1, labelset2):
    LabelCount = 0
    ErrorCount = 0
    for i in range(len(labelset1)):
        if(labelset1[i]!=labelset2[i]):
            ErrorCount+=1
        LabelCount+=1
    result = ErrorCount/LabelCount
    print("The error rate: ",result,"  The total test data:  ",LabelCount)
    return result


def getLabels(dataset):
    labelset = []
    for data in dataset:
         labelset=np.append(labelset,data[13])
    return labelset
# input data_set you can select 2 features
# return the mean vector
def train_classifier(data_set,f1 ,f2):
    sum = 0
    count = 0
    mean_set = []
    data_set_train = data_set[:,[f1,f2]]
    for data in data_set_train:
```

```
        sum+= data
        count+= 1
    mean_set= sum/count
    return mean_set
# input original data set and label only a number

def train_classifer_label(data_set, label , f1 ,f2 ):
    sum = 0
    count = 0

    for i in range(len(data_set)):
        data = data_set[i]
        #print(data[f1:f2+1])
        if data[13] == label :
            sum+= np.array([data[f1],data[f2]])
            count += 1
    mean_set = sum/count
    return mean_set

def get_data_set_by_label(dataset,label):
    result = []
    for data in dataset:
        if data[13] == label:
            dataset.append(data)
    return result



# input:
def classify_with_two_feature(dataset=[[0,0]]):
    result =  []
    for data in dataset :
        result.append(data)


def model_validation(test_dataset, mean_set,f1,f2,errorrate):
    test_dataset = []


def searchFeature(train_data_set, test_Data_set, feature1, feature2):
    mean_set_label1 = train_classifer_label(train_data_set, 1, feature1,
feature2)
    mean_set_label2 = train_classifer_label(train_data_set, 2, feature1,
feature2)
```

```
    mean_set_label3 = train_classifer_label(train_data_set, 3, feature1,
feature2)
    sample_mean_set_unlabelled = np.array([mean_set_label1,
mean_set_label2, mean_set_label3])

    mean_set_label1 = np.append(mean_set_label1, 1)
    mean_set_label2 = np.append(mean_set_label2, 2)
    mean_set_label3 = np.append(mean_set_label3, 3)

    sample_mean_set = [mean_set_label1, mean_set_label2, mean_set_label3]
    estimate_label_set = nearest_classifier(train_data_set,
sample_mean_set, feature1, feature2)
    train_data_labels = getLabels(train_data_set)
    TrainDataErrorRate = computeErrorRate(estimate_label_set,
train_data_labels)

    return TrainDataErrorRate,sample_mean_set_unlabelled ,sample_mean_set

def findBestFeatureByTrainData(train_data, test_data):
    ErrorRateSet = []
    train_data_labels = getLabels(train_data)
    for i in range(13):
        for j in range(i + 1, 13):
            ErrorRate, sample_mean_set_unlabelled,sample_mean_set_labelled =
searchFeature(train_data, test_data, i, j)
            ErrorRateSet.append(ErrorRate)
            if minErrorRate > ErrorRate:
                minErrorRate = ErrorRate
                minFeature1 = i
                minFeature2 = j
                minUnlabelledSampleMean = sample_mean_set_unlabelled
    print("The most suitable feature")
    print(minFeature1, minFeature2)

plotDecBoundaries(train_data[:,[minFeature1,minFeature2]],train_data_labels
,minUnlabelledSampleMean)
```

synthetic.py

```
import numpy as np
from python3.plotDecBoundaries import plotDecBoundaries
from tools.nearest_centroid_classifier import *


```

```
train_data = np.genfromtxt('python3/synthetic2_train.csv',delimiter=',')
test_data =  np.genfromtxt('python3/synthetic2_test.csv',delimiter = ',')
#train_data_2 = np.genfromtxt('python3/synthetic2_train.csv', delimiter =
',')
#print(train_data_1)


sumClass1 = np.array([0,0])
sumClass2 = np.array([0,0])
countClass1 = 0
countClass2 = 0
train_labels = []
test_labels = []
for data in train_data:
    train_labels = np.append(train_labels,data[2])
for data in test_data:
    test_labels = np.append(test_labels,data[2])

for data in train_data:
   if data[2] == 1 :
      sumClass1[0] += data[0]
      sumClass1[1] += data[1]
      countClass1+=1
   else:
      sumClass2[0] += data[0]
      sumClass2[1] += data[1]
      countClass2+=1
estimate_labels = []
test_data_unlabelled = test_data[:,[0,1]]
mean_class1 = sumClass1/countClass1
mean_class2 = sumClass2/countClass2
mean_class1_label = np.append(mean_class1,1)
mean_class2_label = np.append(mean_class2,2)
mean_sample = np.array([mean_class1,mean_class2])
for data in test_data_unlabelled:
   if computeVectorEuclideanDistance(data,mean_class1)>
computeVectorEuclideanDistance(data,mean_class2):
      estimate_labels.append(2)
   else:
      estimate_labels.append(1)

errorrate = computeErrorRate(estimate_labels,test_labels)
```

```
train_data_plot = test_data[:,[0,1]]
plotDecBoundaries(test_data[:,[0,1]],test_labels,mean_sample)
```