```
#nearestMeansClassifier.py

# Name: Tejas Acharya
# Class: EE-559
# Date: 14-06-2023
# Assignment: Homework 1

import numpy as np
from plotDecBoundaries import plotDecBoundaries

class NearestMeansClassifier():
    def __init__(self):
        self.C = 0
        self.means = None
        self.classes = None
        self.features_idx = None


    def fit(self, X, y, features_idx):
        self.classes = np.unique(y)
        self.C = len(self.classes)
        self.features_idx = features_idx
        D = len(features_idx)

        self.means = np.empty((self.C, D))

        total_sum = np.zeros((self.C, D))
        N = np.zeros((self.C, ))

        for i in range(len(y)):
            total_sum[y[i] - 1, :] += X[i, features_idx]
            N[y[i] - 1] += 1

        for j in range(self.C):
            self.means[j, :] = total_sum[j,:] / N[j]

        return


    def predict(self, X):
        N = len(X)

        y_hat = np.empty((N,))

        for i in range(N):
            y_hat[i] = self.get_nearest_class(X[i, self.features_idx])

        y_hat = y_hat.astype('int32')
```

```
        return y_hat


    def get_nearest_class(self, x):
        l2_distances = np.empty((self.C, ))

        for i in range(self.C):
            l2_distances[i] = self.get_l2_norm(self.means[i, :], x)

        return self.classes[np.argmin(l2_distances)]


    def get_l2_norm(self, a, b):
        e = a - b
        return np.sqrt(np.dot(e.T, e))


    def get_error_rate(self, y, y_hat):
        return (sum(y != y_hat) / len(y)) * 100


    def get_class_means(self):
        return self.means


    def plot_boundary(self, X, y):
        plotDecBoundaries(X[:, self.features_idx], y, self.means)
        return
################################################################################

#./synthetic_1/main.py

# Name: Tejas Acharya
# Class: EE-559
# Date: 14-06-2023
# Assignment: Homework 1

import sys
import os
import numpy as np

CURRENT_PATH = os.getcwd()
sys.path.append(CURRENT_PATH)

from nearestMeansClassifier import NearestMeansClassifier

#DATASET FILENAME
```

```
TRAIN_DATASET_FILENAME = os.path.join(CURRENT_PATH, 'synthetic_1',
'synthetic1_train.csv')
TEST_DATASET_FILENAME = os.path.join(CURRENT_PATH, 'synthetic_1',
'synthetic1_test.csv')

def load_data(filename):
    data = np.loadtxt(filename, delimiter=',', dtype=float)
    X = data[:, :-1]
    y = data[:, -1].astype('int32')
    return (X, y)


#LOAD DATA
X_train, y_train = load_data(TRAIN_DATASET_FILENAME)
X_test, y_test = load_data(TEST_DATASET_FILENAME)

features_idx = np.array([0, 1])

#NEAREST MEANS CLASSIFIER MODEL
model = NearestMeansClassifier()

#Fit the Model
model.fit(X_train, y_train, features_idx)

#Predict the TRAIN DATA
y_train_predict = model.predict(X_train)

#Error Rate on TRAIN DATA
train_error_rate = model.get_error_rate(y_train, y_train_predict)

#Plot the Boundary
model.plot_boundary(X_train, y_train)
model.plot_boundary(X_train, y_train_predict)

#Predict the TEST DATA
y_test_predict = model.predict(X_test)

#Error Rate on TEST DATA
test_error_rate = model.get_error_rate(y_test, y_test_predict)

print(f'Train Error Rate: {train_error_rate:.2f}%')
print(f'Test Error Rate: {test_error_rate:.2f}%')
##############################################################################

#./synthetic_2/main.py

# Name: Tejas Acharya
# Class: EE-559
```

```
# Date: 14-06-2023
# Assignment: Homework 1

import sys
import os
import numpy as np

CURRENT_PATH = os.getcwd()
sys.path.append(CURRENT_PATH)

from nearestMeansClassifier import NearestMeansClassifier

#DATASET FILENAME
TRAIN_DATASET_FILENAME = os.path.join(CURRENT_PATH, 'synthetic_2',
'synthetic2_train.csv')
TEST_DATASET_FILENAME = os.path.join(CURRENT_PATH, 'synthetic_2',
'synthetic2_test.csv')


def load_data(filename):
    data = np.loadtxt(filename, delimiter=',', dtype=float)
    X = data[:, :-1]
    y = data[:, -1].astype('int32')
    return (X, y)

#LOAD DATA
X_train, y_train = load_data(TRAIN_DATASET_FILENAME)
X_test, y_test = load_data(TEST_DATASET_FILENAME)

features_idx = np.array([0, 1])

#NEAREST MEANS CLASSIFIER MODEL
model = NearestMeansClassifier()

#Fit the Model
model.fit(X_train, y_train, features_idx)

#Predict the TRAIN DATA
y_train_predict = model.predict(X_train)

#Error Rate on TRAIN DATA
train_error_rate = model.get_error_rate(y_train, y_train_predict)

#Plot the Boundary
model.plot_boundary(X_train, y_train)
model.plot_boundary(X_train, y_train_predict)

#Predict the TEST DATA
```

```
y_test_predict = model.predict(X_test)

#Error Rate on TEST DATA
test_error_rate = model.get_error_rate(y_test, y_test_predict)

print(f'Train Error Rate: {train_error_rate:.2f}%')
print(f'Test Error Rate: {test_error_rate:.2f}%')
################################################################################

#./wine/main.py

# Name: Tejas Acharya
# Class: EE-559
# Date: 14-06-2023
# Assignment: Homework 1

import sys
import os
import numpy as np

CURRENT_PATH = os.getcwd()
sys.path.append(CURRENT_PATH)

from nearestMeansClassifier import NearestMeansClassifier

#DATASET FILENAME
TRAIN_DATASET_FILENAME = os.path.join(CURRENT_PATH, 'wine', 'wine_train.csv')
TEST_DATASET_FILENAME = os.path.join(CURRENT_PATH, 'wine', 'wine_test.csv')


def load_data(filename):
    data = np.loadtxt(filename, delimiter=',', dtype=float)
    X = data[:, :-1]
    y = data[:, -1].astype('int32')
    return (X, y)


#LOAD DATA
X_train, y_train = load_data(TRAIN_DATASET_FILENAME)
X_test, y_test = load_data(TEST_DATASET_FILENAME)

num_features = X_test.shape[1]

#(c) Features - [alcohol content, malic acid content]
features_idx = np.array([0, 1])

#NEAREST MEANS CLASSIFIER MODEL
model = NearestMeansClassifier()
```

```
#Fit the Model
model.fit(X_train, y_train, features_idx)

#Predict the TRAIN DATA
y_train_predict = model.predict(X_train)

#Error Rate on TRAIN DATA
train_error_rate = model.get_error_rate(y_train, y_train_predict)

#Plot the Boundary
model.plot_boundary(X_train, y_train)
model.plot_boundary(X_train, y_train_predict)

#Predict the TEST DATA
y_test_predict = model.predict(X_test)

#Error Rate on TEST DATA
test_error_rate = model.get_error_rate(y_test, y_test_predict)

print(f'Train Error Rate: {train_error_rate:.2f}%')
print(f'Test Error Rate: {test_error_rate:.2f}%')


#(d) Best Feature
features_list = []
for i in range(num_features):
    for j in range(num_features):
        feature_set = {i, j}
        if feature_set not in features_list:
            features_list.append(feature_set)

train_error_rate_list = []

for feature in features_list:
    model = NearestMeansClassifier()
    model.fit(X_train, y_train, list(feature))
    y_train_predict = model.predict(X_train)
    train_error_rate = model.get_error_rate(y_train, y_train_predict)
    train_error_rate_list.append(train_error_rate)

train_error_rate_list = np.array(train_error_rate_list)

best_feature = list(features_list[np.argmin(train_error_rate_list)])
print(f'Best Feature: {best_feature}')

best_model = NearestMeansClassifier()
best_model.fit(X_train, y_train, best_feature)
```

```
y_train_predict_best = best_model.predict(X_train)
train_error_rate_best = model.get_error_rate(y_train, y_train_predict_best)
best_model.plot_boundary(X_train, y_train)
best_model.plot_boundary(X_train, y_train_predict_best)

y_test_predict_best = best_model.predict(X_test)
test_error_rate_best = model.get_error_rate(y_test, y_test_predict_best)

print(f'Train Error Rate(Best Feature): {train_error_rate_best:.2f}%')
print(f'Test Error Rate(Best Feature): {test_error_rate_best:.2f}%')


#(e)
test_error_rate_list = []

for feature in features_list:
    model = NearestMeansClassifier()
    model.fit(X_train, y_train, list(feature))
    y_test_predict = model.predict(X_test)
    test_error_rate = model.get_error_rate(y_test, y_test_predict)
    test_error_rate_list.append(test_error_rate)

test_error_rate_list = np.array(test_error_rate_list)

train_error_mean = train_error_rate_list.mean()
test_error_mean = test_error_rate_list.mean()


train_error_std = train_error_rate_list.std()
test_error_std = test_error_rate_list.std()

train_error_cv = train_error_std / train_error_mean
test_error_cv = test_error_std / test_error_mean

print(f'Train Error STD: {train_error_std:.2f}')
print(f'Train Error CV: {train_error_cv:.2f}')

print(f'Test Error STD: {test_error_std:.2f}')
print(f'Test Error CV: {test_error_cv:.2f}')
```