```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon May 4 05:22:40 2020

@author: melodychen
"""


import matplotlib.pyplot as plt
import csv

ALPHA = 4 # specified by my id number
test_row_begin = 10 * (ALPHA - 1)
test_row_end = 10 * ALPHA

# used to plot data with different colors
x_test_1 = []
x_test_2 = []
x_1_train_1 = []
x_1_train_2 = []
x_0_train_1 = []
x_0_train_2 = []


# loads data from csv file based on alpha
def load_data():
    global y, x, y_test, x_test
    y = []
    x = []
    y_test = []
    x_test = []
    with open("Q2data.csv") as csvfile:
        readCSV = csv.reader(csvfile, delimiter=',')
        for index, row in enumerate(readCSV):
            if test_row_begin <= index < test_row_end:
                x_test_1.append(float(row[0]))
                x_test_2.append(float(row[1]))
                x_test.append([float(row[0]), float(row[1])])
                y_test.append(float(row[2]))
            else:
                if float(row[2]) == 1:
                    x_1_train_1.append(float(row[0]))
                    x_1_train_2.append(float(row[1]))
                else:
                    x_0_train_1.append(float(row[0]))
                    x_0_train_2.append(float(row[1]))
                x.append([float(row[0]), float(row[1])])
                y.append(float(row[2]))


# plots our data in format specified on test
def plot_data():
```

```
    plt.scatter(x_1_train_1, x_1_train_2, label='Label: 1', color='red')
    plt.scatter(x_0_train_1, x_0_train_2, label='Label: 0', color='blue')
    plt.scatter(x_test_1, x_test_2, label='Label: Training', color='cyan')
    plt.xlabel('x1')
    plt.ylabel('x2')
    plt.title("Plot for KNN Data")
    plt.legend()
    plt.axis([min(x_1_train_1 + x_0_train_1 + x_test_1) - 1, max(x_1_train_1 + x_0_train_1 + x_test_1) + 1,
              min(x_1_train_2 + x_0_train_2 + x_test_2) - 1, max(x_1_train_2 + x_0_train_2 + x_test_2) + 1])
    plt.show()


# finds nearest neighbor based on input_points provided
def find_nearest_neighbor(x_input, actual_output, input_points, k, y_tie):
    # calculate L_1 distance between current point and all training points
    distances = []
    for index, row in enumerate(x_input):
        l_distance = 0
        for point, input_point in zip(row, input_points):
            l_distance = l_distance + abs(point - input_point)
        distances.append((index, l_distance))
    # store the distance into an array, find the smallest
    sorted_dist = sorted(distances, key=lambda sl: (sl[1], sl[0]))
    output_0 = 0
    output_1 = 0
    for ind in range(k):
        if actual_output[sorted_dist[ind][0]] == 1:
            output_1 = output_1 + 1
        else:
            output_0 = output_0 + 1
    # if there is a tie, return specified y_tie
    if output_0 == output_1:
        return y_tie
    elif output_0 > output_1:
        return 0
    else:
        return 1


# computes the testing accuracy for each k by calling find_nearest_neighbor
def get_testing_accuracy(k, y_tie):
    correct = 0
    for row, output in zip(x_test, y_test):
        if find_nearest_neighbor(x, y, row, k, y_tie) == output:
            correct += 1
    return correct / len(y_test)


def plot_testing_accuracy(y_tie):
    testing_accuracy = []
    x_axis = []
```

```
105     for index in range(1, 10):
106         accuracy = get_testing_accuracy(index, y_tie)
107         print("K = "+str(index)+" Accuracy = "+str(accuracy))  # prints out
    accuracy
108         testing_accuracy.append(accuracy)
109         x_axis.append(index)
110     plt.title('Testing Accuracy for KNN Classifier')
111     plt.scatter(x_axis, testing_accuracy)
112     plt.axis([0, 10, 0, 1])
113     plt.xlabel('K')
114     plt.ylabel('Testing Accuracy')
115     plt.show()
116
117
118 if __name__ == "__main__":
119     load_data()
120     # plot_data() # uncomment this to plot data, and comment plot(0) out below to
    prevent graph overlap
121     plot_testing_accuracy(0)  # 0 as we want to classify class to 0 when there is
    a tie
122
123
```