```
#Sean McGlincy
#Homework 1
#KNN

import numpy as np

# Main driving function
def knn_main(training_data, test_data, k):
    # Accumulators
    correct = 0
    wrong = 0
    row = 0


    # Loop through each row of test Data
    for data in test_data:
        # Call Helper function to process each row
        results = knn(training_data, data, k)
        row += 1

        # Accumulate results, if results equal answer
        if (results == int(data[0])):
            correct += 1
        else:
            wrong += 1

    accuracy = correct / (correct + wrong)
    return accuracy


# Helper function to process each row
def knn(training_data, data, k):
    #Accumulator of each row
    distance = []

    # Cycle through data and calculate distance  28 x 28 features
    for test_case in training_data:

        # Accumulator for distance formula
        dist = 0.0
        for i in range(1, len(test_case)):


            # Distance formula
            dist += (test_case[i]  - data[i])**2   # Distance without Normalization
        dist = np.math.sqrt(dist)


        # Add distance and test data value
        item = (dist, int(test_case[0]) )
        distance.append(item)

    #Sort Data by distance
    sorted_dist = sorted(distance, key=lambda tup: tup[0])
    #Rank K elements, accumulator
    ranking_array = np.zeros(10, dtype=int)
    for i in range(0, k):

        # Get result from training data and add to accumulator
        num = sorted_dist[i][1]   #tuple(dist, answer)
        ranking_array[ num ]  += 1
```

```
    return ranking_array.argmax()



#  Main Start Program HERE
training_data = np.genfromtxt('MNIST_training.csv', delimiter=',', skip_header=1)
test_data = np.genfromtxt('MNIST_test.csv', delimiter=',', skip_header=1)


# Call Main Driving Function
results = knn_main(training_data, test_data, 7)
print("Accuracy is ", results)


#
#
# #
# # Find optimal K
# find_k_array =  np.array(0)
# #  Find optimal K by cycling through odd values 1...25
# for i in range(1, 25, 2):
#     results = knn_main(training_data, test_data, i)
#     find_k_array = np.append(find_k_array, results)
#
#
# print(find_k_array)
# max = find_k_array.argmax()
# print("optimal K: ", (max * 2) -1 )
```