```python
import csv
import numpy as np
import matplotlib.pyplot as plt

tmpfile = open("MNIST_training.csv", "rt", encoding='utf-8')
training = np.array(list(csv.reader(tmpfile)))
trainingY = np.array(training[1:, 0], dtype=int)
trainingX = np.array(training[1:, 1:], dtype=int)

tmpfile = open("MNIST_test.csv", "rt", encoding='utf-8')
test = np.array(list(csv.reader(tmpfile)))
testY = np.array(test[1:, 0], dtype=int)
testX = np.array(test[1:, 1:], dtype=int)

## Distance between test sample and training sample
Distance = np.zeros((len(testX), len(trainingX)))
for i in range (len(testX)):
    for j in range (len(trainingX)):
        Distance[i, j] = (np.sum((testX[i]-trainingX[j])**2))**(1/2)

## Make some matrix to help prediction
KNNlabel = np.zeros((len(testY), 10), dtype=int) # count the number of each
label of KNN of test samples.
Prediction = np.zeros((len(testY), 1), dtype=int) # predict which label testY
is.
Accuracy = np.zeros((95, 1), dtype=np.float16)  # accuracy of KNN algorithm
based on size of K.
Distance2 = np.copy(Distance)

## Calculate Accuracy and Prediction
for k in range(1, 96): # k is the number of nearest neighbor
    for i in range(0, 50): # for i-th sample in test data
        for j in range(0, k):
            index = np.argmin(np.array(Distance2[i]))
            label = trainingY[index]

            # count which label i-th test sample is predicted as
            KNNlabel[i, label] = (KNNlabel[i, label]) + 1

            # exclude this sample in training data for next loop
            Distance2[i, index] = np.max(Distance2[i])

        Prediction[i] = np.argmax(np.array(KNNlabel[i]))

    # the number of correctly classificated
    correct = 0
    for x in range(0, 50):
        if int(Prediction[x]) == int(testY[x]):
            correct = correct + 1

    # calculate Accuracy
    Accuracy[k-1, 0] = format((correct/50), '.2f')
    # put the original distance in Distance2 array
    Distance2 = np.copy(Distance)

    print('k = ', k)
    print('correct = ', correct)

## Plot the Result
print(Accuracy)
print('optimum K = ', np.argmax(Accuracy))
print('Accuracy = ', np.max(Accuracy))

plt.plot(Accuracy, 'r--')
plt.axis([0, 95, 0, 1])
plt.show()
```