

Homework Assignment 1

Name: Linqi Xiao

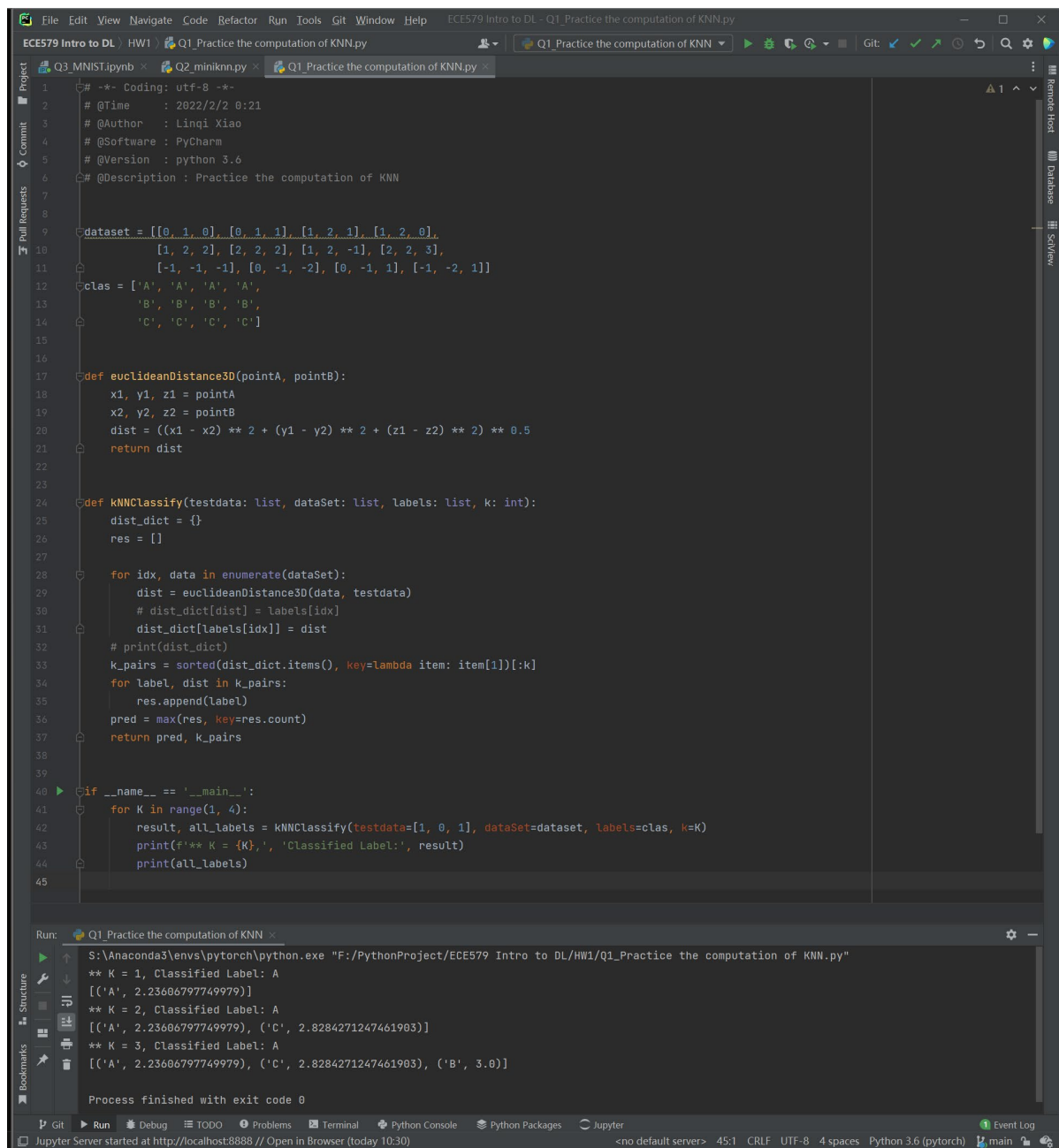
NetID: lx130

Problem 1 (Practice the computation of KNN)

When $K = 1$, classified label is 'A'

When $K = 2$, classified label is 'A'

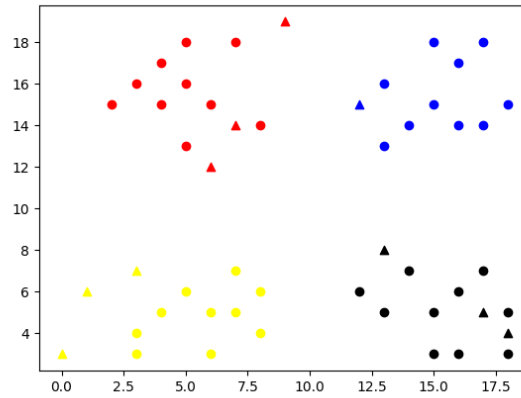
When $K = 3$, classified label is 'A'



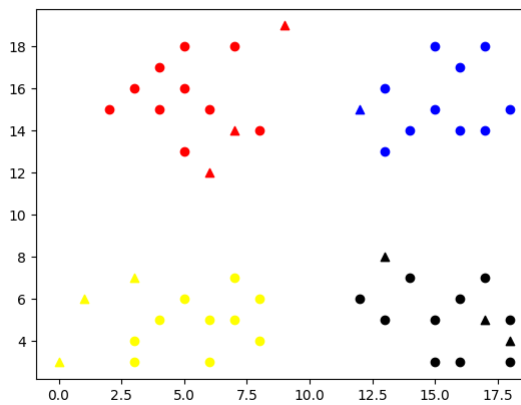
```
1  # -*- Coding: utf-8 -*-
2  # @Time      : 2022/2/2 0:21
3  # @Author    : Linqi Xiao
4  # @Software  : PyCharm
5  # @Version   : python 3.6
6  # @Description: Practice the computation of KNN
7
8
9  dataset = [[0, 1, 0], [0, 1, 1], [1, 2, 1], [1, 2, 0],
10            [1, 2, 2], [2, 2, 2], [1, 2, -1], [2, 2, 3],
11            [-1, -1, -1], [0, -1, -2], [0, -1, 1], [-1, -2, 1]]
12  clas = ['A', 'A', 'A', 'A',
13          'B', 'B', 'B', 'B',
14          'C', 'C', 'C', 'C']
15
16
17  def euclideanDistance3D(pointA, pointB):
18      x1, y1, z1 = pointA
19      x2, y2, z2 = pointB
20      dist = ((x1 - x2) ** 2 + (y1 - y2) ** 2 + (z1 - z2) ** 2) ** 0.5
21      return dist
22
23
24  def KNNClassify(testdata: list, dataSet: list, labels: list, k: int):
25      dist_dict = {}
26      res = []
27
28      for idx, data in enumerate(dataSet):
29          dist = euclideanDistance3D(data, testdata)
30          # dist_dict[dist] = labels[idx]
31          dist_dict[labels[idx]] = dist
32          # print(dist_dict)
33      k_pairs = sorted(dist_dict.items(), key=lambda item: item[1])[:k]
34      for label, dist in k_pairs:
35          res.append(label)
36      pred = max(res, key=res.count)
37      return pred, k_pairs
38
39
40  if __name__ == '__main__':
41      for K in range(1, 4):
42          result, all_labels = KNNClassify(testdata=[1, 0, 1], dataSet=dataset, labels=clas, k=K)
43          print(f'** K = {K}, 'Classified Label:', result)
44          print(all_labels)
45
46
47  Run: S:\Anaconda3\envs\pytorch\python.exe "F:/PythonProject/ECE579 Intro to DL/HW1/Q1_Practice the computation of KNN.py"
48  ** K = 1, Classified Label: A
49  [['A', 2.23606797749979]]
50  ** K = 2, Classified Label: A
51  [['A', 2.23606797749979], ('C', 2.8284271247461903)]
52  ** K = 3, Classified Label: A
53  [['A', 2.23606797749979], ('C', 2.8284271247461903), ('B', 3.0)]
54
55  Process finished with exit code 0
```

Problem 2 (KNN for simple data)

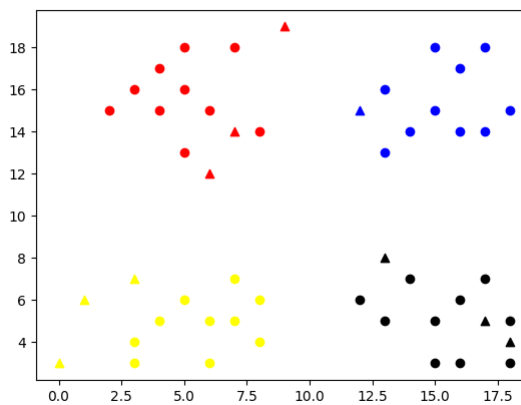
$K = 1$



$K = 5$



$K = 10$

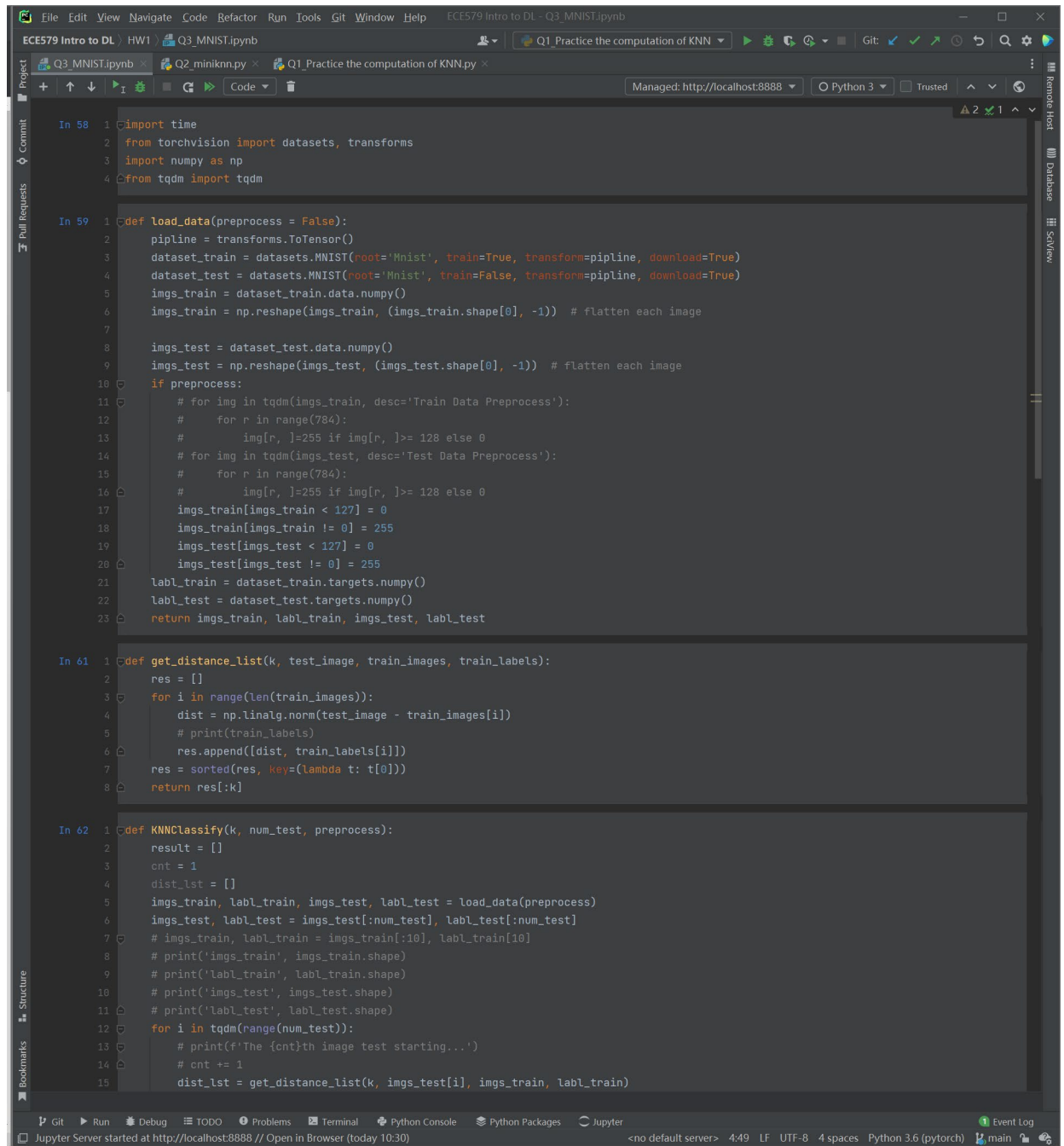


```
File Edit View Navigate Code Refactor Run Tools VCS Window Help HW1 - minikmn.py
minikmn.py
1 import matplotlib as mpl
2 import numpy as np
3 import matplotlib.pyplot as plt
4 # mpl.use('Agg')
5
6
7 # load mini training data and labels
8 mini_train = np.load('knn_mini_train.npy')
9 mini_train_label = np.load('knn_mini_train_label.npy')
10
11 # randomly generate test data
12 np.random.seed(0)
13 mini_test = np.random.randint(20, size=20)
14 mini_test = mini_test.reshape(10, 2)
15
16
17 # Define knn classifier
18 def KNNClassify(newInput, dataSet, labels, k):
19     result = []
20     for new_id, new_xy in enumerate(newInput):
21         dist_dict = {}
22         knn_labels = []
23         for old_id, old_xy in enumerate(dataSet):
24             dist = euclideanDistance2D(new_xy, old_xy)
25             dist_dict[old_id] = dist
26             k_pairs = sorted(dist_dict.items(), key=lambda item: item[1])[:k]
27             for old_knn_id, knn_dist in k_pairs:
28                 knn_labels.append(labels[old_knn_id])
29             pred = max(knn_labels, key=knn_labels.count)
30             result.append(pred)
31     return result
32
33
34 def euclideanDistance2D(pointA, pointB):
35     x1, y1 = pointA
36     x2, y2 = pointB
37     dist = ((x1 - x2) ** 2 + (y1 - y2) ** 2) ** 0.5
38     return dist
39
40
Run minikmn
S:\Anaconda3\envs\pytorch\python.exe "F:/PythonProject/ECE579 Intro to DL/HW1/minikmn.py"
random test points are: [[12 15]
[ 0  3]
[ 3  7]
[ 9 19]
[18  4]
[ 6 12]
[ 1  6]
[ 7 14]
[17  5]
[13  8]]
knn classified labels for test: [1, 2, 2, 0, 3, 0, 2, 0, 3, 3]
Process finished with exit code 0
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help HW1 - minikmn.py
minikmn.py
37
38 def euclideanDistance2D(pointA, pointB):
39     x1, y1 = pointA
40     x2, y2 = pointB
41     dist = ((x1 - x2) ** 2 + (y1 - y2) ** 2) ** 0.5
42     return dist
43
44
45 outputLabels = KNNClassify(mini_test, mini_train, mini_train_label, 5)
46
47 print('random test points are:', mini_test)
48 print('knn classified labels for test:', outputLabels)
49
50 # plot train data and classified test data
51 train_x = mini_train[:, 0]
52 train_y = mini_train[:, 1]
53 fig = plt.figure()
54 plt.scatter(train_x[np.where(mini_train_label == 0)], train_y[np.where(mini_train_label == 0)], color='red')
55 plt.scatter(train_x[np.where(mini_train_label == 1)], train_y[np.where(mini_train_label == 1)], color='blue')
56 plt.scatter(train_x[np.where(mini_train_label == 2)], train_y[np.where(mini_train_label == 2)], color='yellow')
57 plt.scatter(train_x[np.where(mini_train_label == 3)], train_y[np.where(mini_train_label == 3)], color='black')
58
59 test_x = mini_test[:, 0]
60 test_y = mini_test[:, 1]
61 outputLabels = np.array(outputLabels)
62 plt.scatter(test_x[np.where(outputLabels == 0)], test_y[np.where(outputLabels == 0)], marker='x', color='red')
63 plt.scatter(test_x[np.where(outputLabels == 1)], test_y[np.where(outputLabels == 1)], marker='x', color='blue')
64 plt.scatter(test_x[np.where(outputLabels == 2)], test_y[np.where(outputLabels == 2)], marker='x', color='yellow')
65 plt.scatter(test_x[np.where(outputLabels == 3)], test_y[np.where(outputLabels == 3)], marker='x', color='black')
66
67 # save diagram as png file
68 # plt.show()
69
70 plt.savefig('minikmn.png')
71
Run minikmn
S:\Anaconda3\envs\pytorch\python.exe "F:/PythonProject/ECE579 Intro to DL/HW1/minikmn.py"
random test points are: [[12 15]
[ 0  3]
[ 3  7]
[ 9 19]
[18  4]
[ 6 12]
[ 1  6]
[ 7 14]
[17  5]
[13  8]]
knn classified labels for test: [1, 2, 2, 0, 3, 0, 2, 0, 3, 3]
Process finished with exit code 0
```

Problem 3 (KNN for handwriting digit recognition)

I choose 20 testing images and train them on all 60000 training datasets, the accuracy is 0.95



```
File Edit View Navigate Code Refactor Run Tools Git Window Help ECE579 Intro to DL - Q3_MNIST.ipynb
Q3_MNIST.ipynb HW1 Q3_MNIST.ipynb Q1_Practice the computation of KNN
Managed: http://localhost:8888 Python 3 Trusted
In 58 1 import time
2 from torchvision import datasets, transforms
3 import numpy as np
4 from tqdm import tqdm

In 59 1 def load_data(preprocess = False):
2     pipeline = transforms.ToTensor()
3     dataset_train = datasets.MNIST(root='Mnist', train=True, transform=pipeline, download=True)
4     dataset_test = datasets.MNIST(root='Mnist', train=False, transform=pipeline, download=True)
5     imgs_train = dataset_train.data.numpy()
6     imgs_train = np.reshape(imgs_train, (imgs_train.shape[0], -1)) # Flatten each image
7
8     imgs_test = dataset_test.data.numpy()
9     imgs_test = np.reshape(imgs_test, (imgs_test.shape[0], -1)) # Flatten each image
10    if preprocess:
11        # for img in tqdm(imgs_train, desc='Train Data Preprocess'):
12        #     for r in range(784):
13        #         img[r, :] = 255 if img[r, :] >= 128 else 0
14        # for img in tqdm(imgs_test, desc='Test Data Preprocess'):
15        #     for r in range(784):
16        #         img[r, :] = 255 if img[r, :] >= 128 else 0
17        imgs_train[imgs_train < 127] = 0
18        imgs_train[imgs_train != 0] = 255
19        imgs_test[imgs_test < 127] = 0
20        imgs_test[imgs_test != 0] = 255
21    labl_train = dataset_train.targets.numpy()
22    labl_test = dataset_test.targets.numpy()
23    return imgs_train, labl_train, imgs_test, labl_test

In 61 1 def get_distance_list(k, test_image, train_images, train_labels):
2     res = []
3     for i in range(len(train_images)):
4         dist = np.linalg.norm(test_image - train_images[i])
5         # print(train_labels)
6         res.append([dist, train_labels[i]])
7     res = sorted(res, key=lambda t: t[0])
8     return res[:k]

In 62 1 def KNNClassify(k, num_test, preprocess):
2     result = []
3     cnt = 1
4     dist_lst = []
5     imgs_train, labl_train, imgs_test, labl_test = load_data(preprocess)
6     imgs_test, labl_test = imgs_test[:num_test], labl_test[:num_test]
7     # imgs_train, labl_train = imgs_train[:10], labl_train[10]
8     # print('imgs_train', imgs_train.shape)
9     # print('labl_train', labl_train.shape)
10    # print('imgs_test', imgs_test.shape)
11    # print('labl_test', labl_test.shape)
12    for i in tqdm(range(num_test)):
13        # print(f'The {cnt}th image test starting...')
14        # cnt += 1
15        dist_lst = get_distance_list(k, imgs_test[i], imgs_train, labl_train)
```

Git Run Debug TODO Problems Terminal Python Console Python Packages Jupyter
Jupyter Server started at http://localhost:8888 // Open in Browser (today 10:30) <no default server> 4:49 LF UTF-8 4 spaces Python 3.6 (pytorch) main

```
ECE579 Intro to DL - Q3_MNIST.ipynb
Q3_MNIST.ipynb
Q2_miniknn.py
Q1_Practice the computation of KNN.py
Managed: http://localhost:8888 Python 3 Trusted
Remote host Database SQL view

In 68 1 def KNNClassify(k, num_test, preprocess):
2     result = []
3     cnt = 1
4     imgs_train, labl_train, imgs_test, labl_test = load_data(preprocess)
5     imgs_test, labl_test = imgs_test[:num_test], labl_test[:num_test]
6     # imgs_train, labl_train = imgs_train[:10], labl_train[10]
7     # print('imgs_train', imgs_train.shape)
8     # print('labl_train', labl_train.shape)
9     # print('imgs_test', imgs_test.shape)
10    # print('labl_test', labl_test.shape)
11    for i in tqdm(range(num_test)):
12        # print(f'The {cnt}th image test starting...')
13        # cnt += 1
14        dist_lst = get_distance_list(k, imgs_test[i], imgs_train, labl_train)
15        k_labels = []
16
17        for dist, label in dist_lst:
18            k_labels.append(label)
19        result.append(max(k_labels, key=k_labels.count))
20
21    print('Prediction->', result)
22    return result

In 69 1 def evaluate():
2     k = 20
3     num_test = 20
4     num_correct = 0
5     imgs_train, labl_train, imgs_test, labl_test = load_data()
6     # print('imgs_train', imgs_train.shape)
7     # print('labl_train', labl_train.shape)
8     # print('imgs_test', imgs_test.shape)
9     # print('labl_test', labl_test.shape)
10    start_time = time.time()
11    outputLabels = KNNClassify(k, num_test, preprocess=True)
12    for idx, pred_label in enumerate(outputLabels):
13        num_correct += (pred_label == labl_test[idx])
14    accuracy = num_correct / num_test
15    print("---classification accuracy for knn on mnist: %s ---" % accuracy)
16    print("---execution time: %s seconds ---" % (time.time() - start_time))

In 70 1 evaluate()

100%|██████████| 20/20 [00:05<00:00, 3.81it/s]
Prediction-> [7, 2, 1, 0, 4, 1, 4, 9, 0, 9, 0, 6, 9, 0, 1, 5, 9, 7, 3, 4]
---classification accuracy for knn on mnist: 0.95 ---
---execution time: 5.423188924789429 seconds ---

KNNClassify0
Git Run Debug TODO Problems Terminal Python Console Python Packages Jupyter
Jupyter Server started at http://localhost:8888 // Open in Browser (today 10:30)
<no default server> 42:12 LF UTF-8 4 spaces Python 3.6 (pytorch) main
```