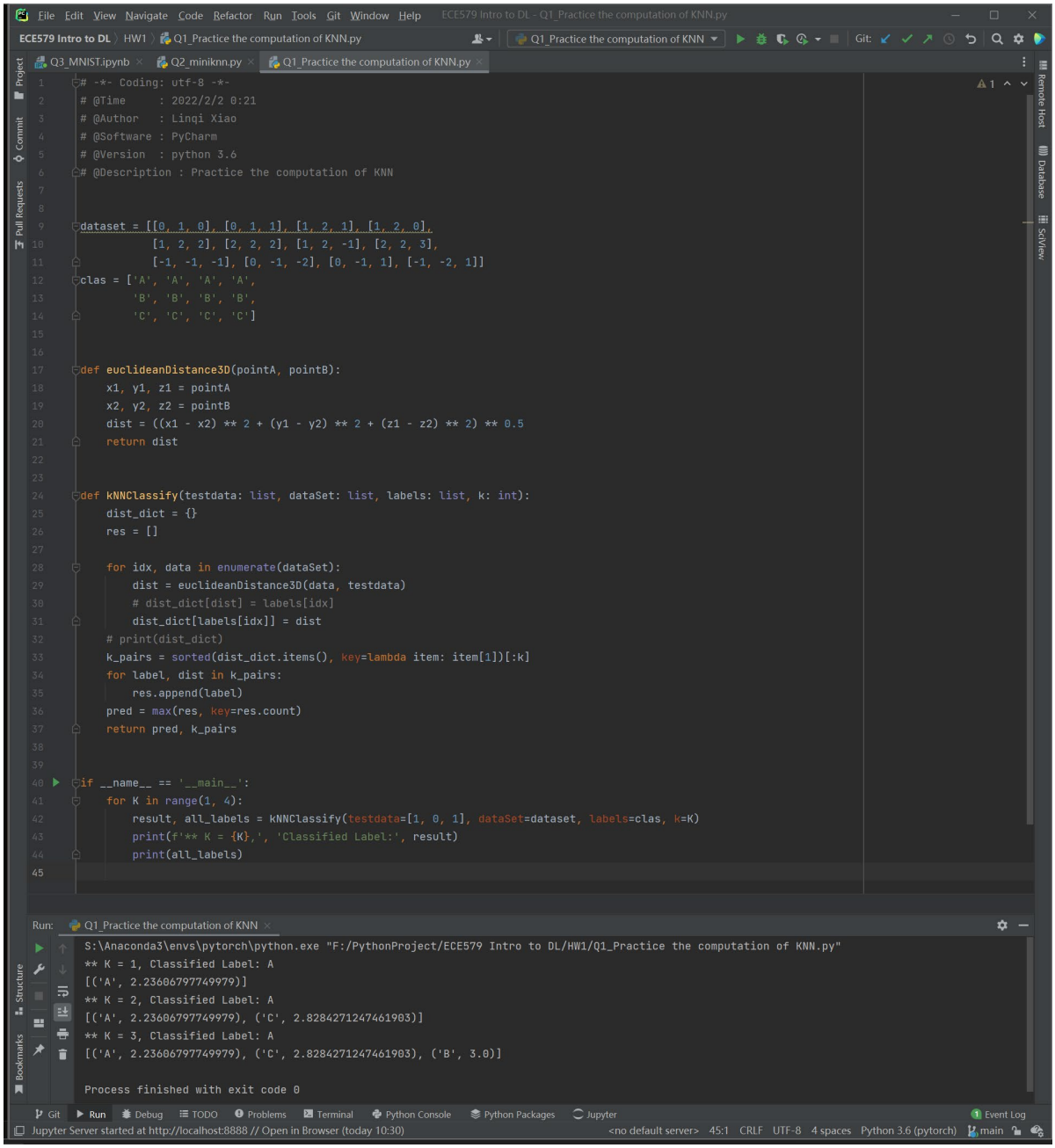
Homework Assignment 1

Name: Linqi Xiao NetID: lx130

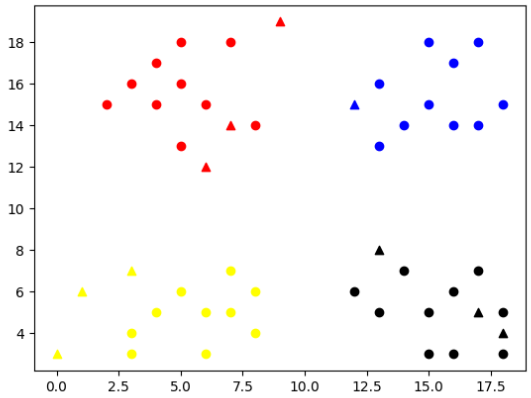
Problem 1 (Practice the computation of KNN)

When K = 1, classified label is ‘A’
When K = 2, classified label is ‘A’
When K = 3, classified label is ‘A’

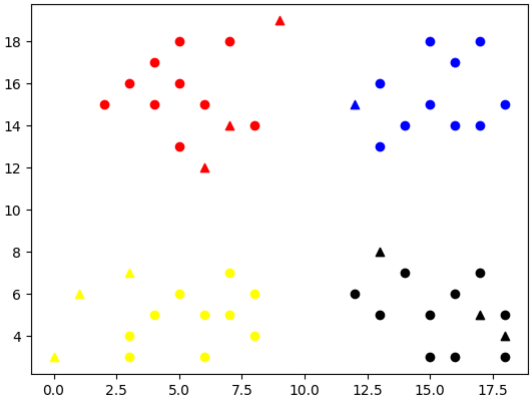


Problem 2 (KNN for simple data)

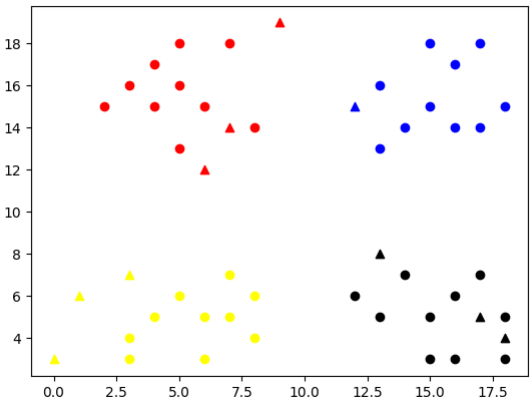
K = 1

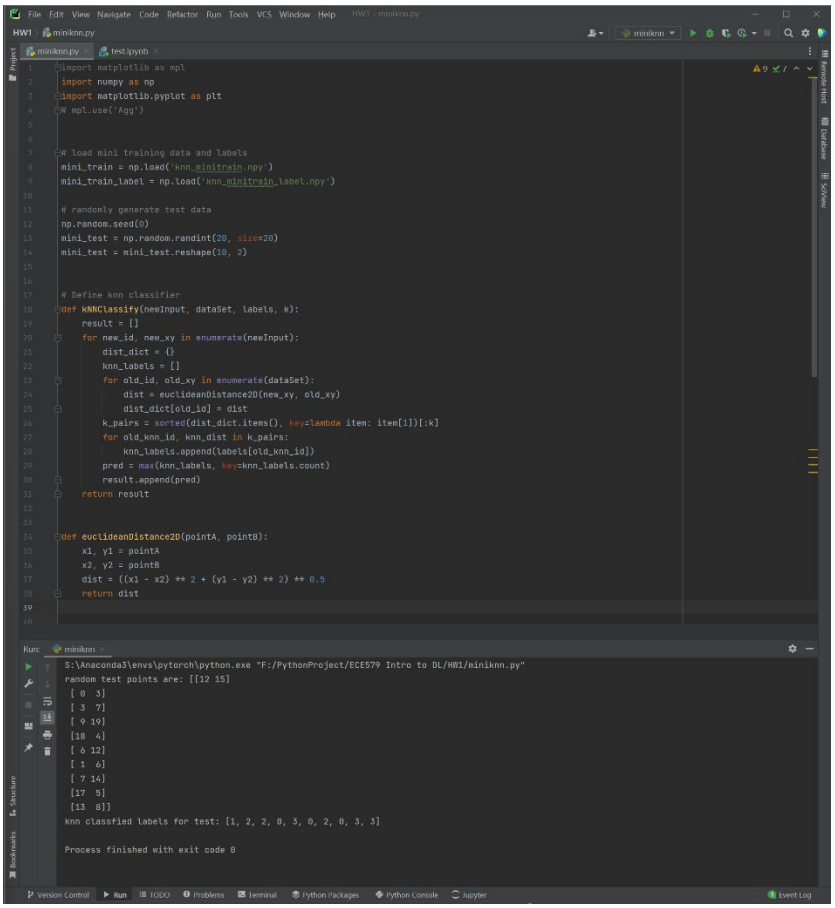


K = 5



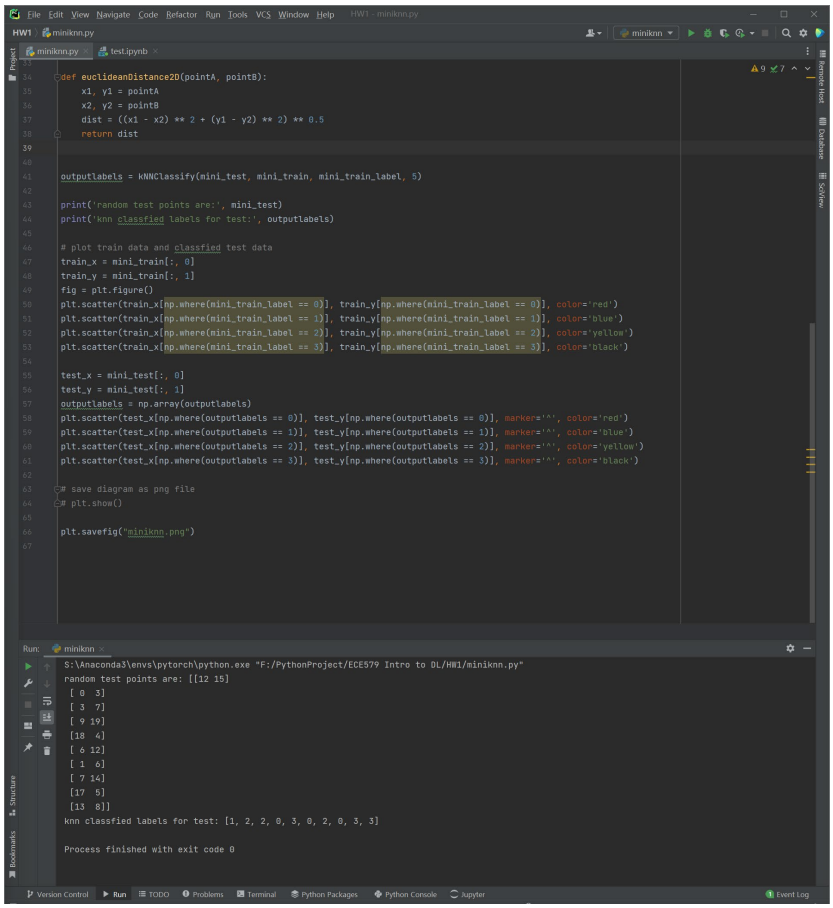
K = 10





```
import matplotlib as mpl
import numpy as np
import matplotlib.pyplot as plt
# mpl.use('Agg')


# load mini training data and labels
mini_train = np.load('knn_minitrain.npy')
mini_train_label = np.load('knn_minitrain_label.npy')

# randomly generate test data
np.random.seed(0)
mini_test = np.random.randint(20, size=20)
mini_test = mini_test.reshape(10, 2)


# Define knn classifier
def kNNClassify(newInput, dataSet, labels, k):
    result = []
    for new_id, new_xy in enumerate(newInput):
        dist_dict = {}
        knn_labels = []
        for old_id, old_xy in enumerate(dataSet):
            dist = euclideanDistance2D(new_xy, old_xy)
            dist_dict[old_id] = dist
        k_pairs = sorted(dist_dict.items(), key=lambda item: item[1])[:k]
        for old_knn_id, knn_dist in k_pairs:
            knn_labels.append(labels[old_knn_id])
        pred = max(knn_labels, key=knn_labels.count)
        result.append(pred)
    return result


def euclideanDistance2D(pointA, pointB):
    x1, y1 = pointA
    x2, y2 = pointB
    dist = ((x1 - x2) ** 2 + (y1 - y2) ** 2) ** 0.5
    return dist
```

```
S:\Anaconda3\envs\pytorch\python.exe "F:/PythonProject/ECE579 Intro to DL/HW1/miniknn.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 classfied labels for test: [1, 2, 2, 0, 3, 0, 2, 0, 3, 3]

Process finished with exit code 0
```

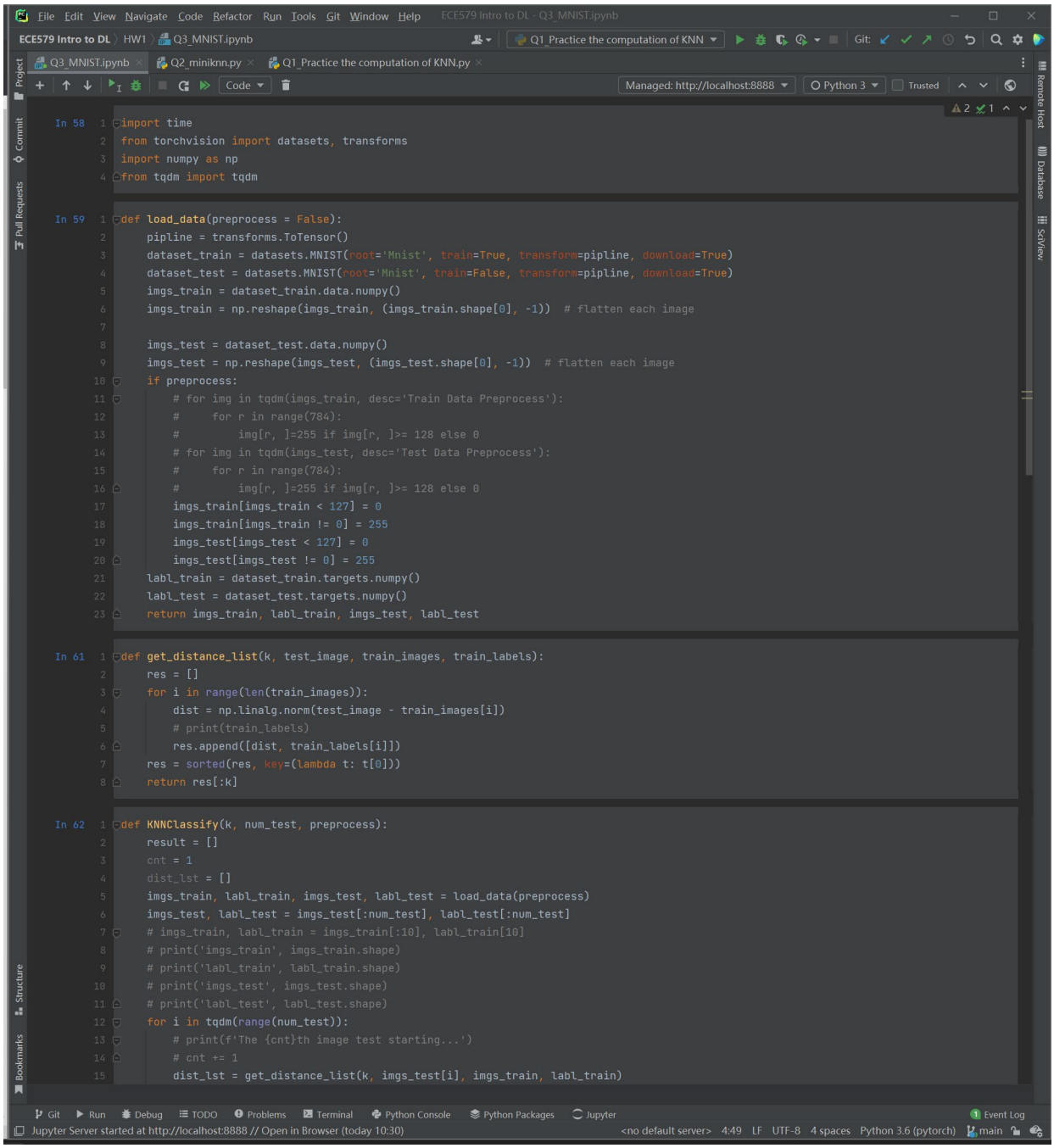


```
def euclideanDistance2D(pointA, pointB):
    x1, y1 = pointA
    x2, y2 = pointB
    dist = ((x1 - x2) ** 2 + (y1 - y2) ** 2) ** 0.5
    return dist


outputlabels = kNNClassify(mini_test, mini_train, mini_train_label, 5)

print('random test points are:', mini_test)
print('knn classfied labels for test:', outputlabels)

# plot train data and classfied test data
train_x = mini_train[:, 0]
train_y = mini_train[:, 1]
fig = plt.figure()
plt.scatter(train_x[np.where(mini_train_label == 0)], train_y[np.where(mini_train_label == 0)], color='red')
plt.scatter(train_x[np.where(mini_train_label == 1)], train_y[np.where(mini_train_label == 1)], color='blue')
plt.scatter(train_x[np.where(mini_train_label == 2)], train_y[np.where(mini_train_label == 2)], color='yellow')
plt.scatter(train_x[np.where(mini_train_label == 3)], train_y[np.where(mini_train_label == 3)], color='black')

test_x = mini_test[:, 0]
test_y = mini_test[:, 1]
outputlabels = np.array(outputlabels)
plt.scatter(test_x[np.where(outputlabels == 0)], test_y[np.where(outputlabels == 0)], marker='^', color='red')
plt.scatter(test_x[np.where(outputlabels == 1)], test_y[np.where(outputlabels == 1)], marker='^', color='blue')
plt.scatter(test_x[np.where(outputlabels == 2)], test_y[np.where(outputlabels == 2)], marker='^', color='yellow')
plt.scatter(test_x[np.where(outputlabels == 3)], test_y[np.where(outputlabels == 3)], marker='^', color='black')

# save diagram as png file
# plt.show()

plt.savefig("miniknn.png")
```

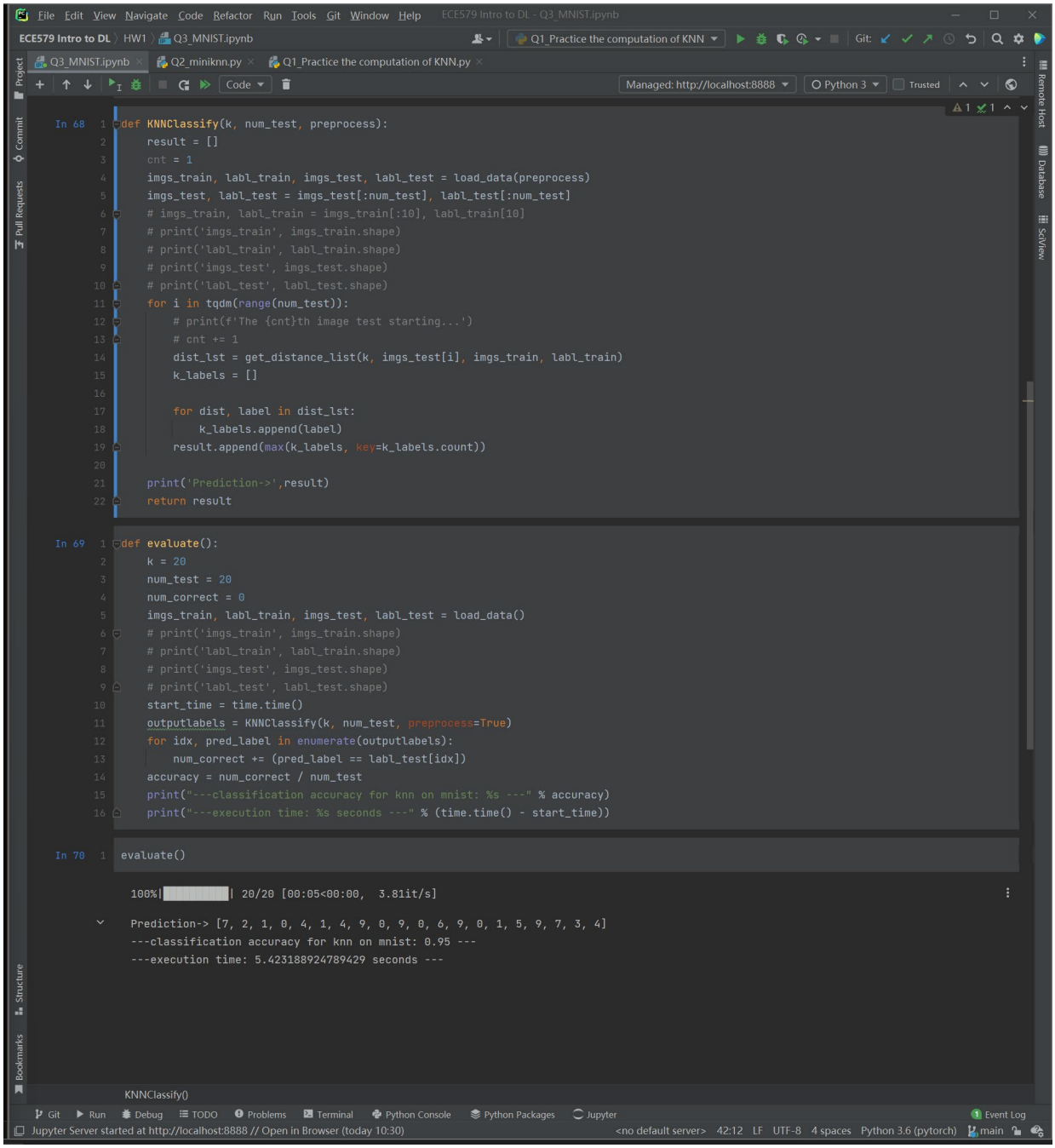
```
S:\Anaconda3\envs\pytorch\python.exe "F:/PythonProject/ECE579 Intro to DL/HW1/miniknn.py"
random test points are: [[12 15]
 [ 0  3]
 [ 3  7]
 [ 9 19]
 [18  4]
 [ 6 12]
 [ 1  6]
 [ 7 14]
 [17  5]
 [13  8]]
knn classfied labels for test: [1, 2, 2, 0, 3, 0, 2, 0, 3, 3]

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



```
In 58
import time
from torchvision import datasets, transforms
import numpy as np
from tqdm import tqdm

In 59
def load_data(preprocess = False):
    pipline = transforms.ToTensor()
    dataset_train = datasets.MNIST(root='Mnist', train=True, transform=pipline, download=True)
    dataset_test = datasets.MNIST(root='Mnist', train=False, transform=pipline, download=True)
    imgs_train = dataset_train.data.numpy()
    imgs_train = np.reshape(imgs_train, (imgs_train.shape[0], -1))  # flatten each image

    imgs_test = dataset_test.data.numpy()
    imgs_test = np.reshape(imgs_test, (imgs_test.shape[0], -1))  # flatten each image
    if preprocess:
        # for img in tqdm(imgs_train, desc='Train Data Preprocess'):
        #     for r in range(784):
        #         img[r, ]=255 if img[r, ]>= 128 else 0
        # for img in tqdm(imgs_test, desc='Test Data Preprocess'):
        #     for r in range(784):
        #         img[r, ]=255 if img[r, ]>= 128 else 0
        imgs_train[imgs_train < 127] = 0
        imgs_train[imgs_train != 0] = 255
        imgs_test[imgs_test < 127] = 0
        imgs_test[imgs_test != 0] = 255
    labl_train = dataset_train.targets.numpy()
    labl_test = dataset_test.targets.numpy()
    return imgs_train, labl_train, imgs_test, labl_test

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

In 62
def KNNClassify(k, num_test, preprocess):
    result = []
    cnt = 1
    dist_lst = []
    imgs_train, labl_train, imgs_test, labl_test = load_data(preprocess)
    imgs_test, labl_test = imgs_test[:num_test], labl_test[:num_test]
    # imgs_train, labl_train = imgs_train[:10], labl_train[10]
    # print('imgs_train', imgs_train.shape)
    # print('labl_train', labl_train.shape)
    # print('imgs_test', imgs_test.shape)
    # print('labl_test', labl_test.shape)
    for i in tqdm(range(num_test)):
        # print(f'The {cnt}th image test starting...')
        # cnt += 1
        dist_lst = get_distance_list(k, imgs_test[i], imgs_train, labl_train)
```


File Edit View Navigate Code Refactor Run Tools Git Window Help
ECE579 Intro to DL - Q3_MNIST.ipynb
ECE579 Intro to DL 〉 HW1 〉 Q3_MNIST.ipynb
Q1_Practice the computation of KNN
Git:
Q3_MNIST.ipynb
Q2_miniknn.py
Q1_Practice the computation of KNN.py
Code
Managed: http://localhost:8888
Python 3
Trusted
In 68
def KNNClassify(k, num_test, preprocess):
result = []
cnt = 1
imgs_train, labl_train, imgs_test, labl_test = load_data(preprocess)
imgs_test, labl_test = imgs_test[:num_test], labl_test[:num_test]
# imgs_train, labl_train = imgs_train[:10], labl_train[10]
# print('imgs_train', imgs_train.shape)
# print('labl_train', labl_train.shape)
# print('imgs_test', imgs_test.shape)
# print('labl_test', labl_test.shape)
for i in tqdm(range(num_test)):
# print(f'The {cnt}th image test starting...')
# cnt += 1
dist_lst = get_distance_list(k, imgs_test[i], imgs_train, labl_train)
k_labels = []
for dist, label in dist_lst:
k_labels.append(label)
result.append(max(k_labels, key=k_labels.count))
print('Prediction->',result)
return result
In 69
def evaluate():
k = 20
num_test = 20
num_correct = 0
imgs_train, labl_train, imgs_test, labl_test = load_data()
# print('imgs_train', imgs_train.shape)
# print('labl_train', labl_train.shape)
# print('imgs_test', imgs_test.shape)
# print('labl_test', labl_test.shape)
start_time = time.time()
outputlabels = KNNClassify(k, num_test, preprocess=True)
for idx, pred_label in enumerate(outputlabels):
num_correct += (pred_label == labl_test[idx])
accuracy = num_correct / num_test
print("---classification accuracy for knn on mnist: %s ---" % accuracy)
print("---execution time: %s seconds ---" % (time.time() - start_time))
In 70
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 ---
KNNClassify()
Git Run Debug TODO Problems Terminal Python Console Python Packages Jupyter
Event Log
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