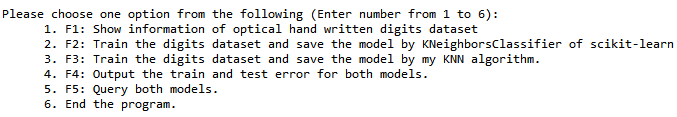
**Checklist:**

|  |  |  |
| --- | --- | --- |
| F1: | 1. Successfully load the dataset | √ |
|  | 1. Display the dataset information, including the number of data entries, the number of classes, the number of data entries for each class, the minimum and maximum values for each feature, and the train dataset and test dataset split | √ |
| F2: | 1. Successfully call library functions to train a model. | √ |
|  | 1. There needs to be a corresponding saved model in your submission. | √ |
| F3: | 1. You have an implementation of an algorithm that is able to train a model. | √ |
|  | 1. There needs to be a corresponding saved model in your submission. | √ |
| F4: | 1. Output the train and test errors for f2’s model | √ |
|  | 1. Output the train and test errors for f3’s model | √ |
| F5: | 1. Allow users to query two models by changing the input | √ |

***1. Detailing how to run your program, including the software dependencies***

**how to run my program：**



This is the user interface of my program. User should choose a number from 1 to 6, any other numbers or digits will not be accepted. The first option intends to show the dataset information. The second option means training the dataset by calling the KNN algorithm from the machine learning library scikit-learn and saving the model after the training process. The third option means training the dataset by my own KNN algorithm. The model will be saved afterwards. The fourth option outputs the training and testing accuracy of my saved KNN model and sklearn knn model. The fifth option will query the model by asking the user to input the index of the test dataset. Finally, upon get the input ‘6’, the program will end.

**software dependencies:**

The program is written by python. It can run in Spyder(python 3.7) and sklearn 0.21. It uses Scikit-learn algorithm from the machine learning libraries. Numpy, pickle, time, matplotlib are all imported in my program. Classifification is chosen.

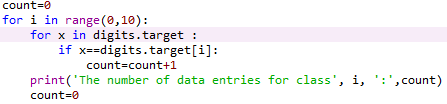
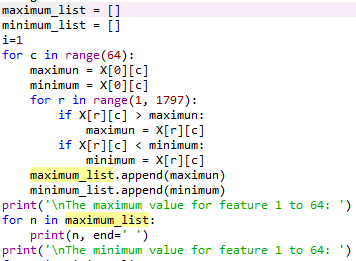
***2. Explaining how the functionalities and additional requirements are implemented***

**F1:** Call the method: ShowInfo() in my program

1. the number of data entries：n\_samples,n\_features=digits.data.shape

n\_samples is the number of data entries

1. the number of classes: digits.target\_names.size is the number of classes
2. the number of data entries for each class: In this for loop, add 1 to count if the x in digits dataset is the same with i in range(0,10). Count is the number of data entries for each class from 0-9



1. the minimum and maximum values for each feature: use a for loop to compare the maximum and minimum value for each feature, and meanwhile add that mim/max value into maximum\_list and minimum\_list
2. the train dataset and test dataset split:

Split the dataset by putting 80% of data into the train dataset and 20% into the test dataset

X\_train\_samples, X\_train\_features=X\_train.data.shape

X\_test\_samples=n\_samples-X\_train\_samples

X\_train\_samples/ X\_test\_samples is the number of training/testing data entries.

**F2:** Call the method: trainSklearnKNN() in my program

1. call scikit-learn library functions to train a model



This is the scikit-learn library’s KNN classifier. I import it at the beginning of the program.

Firstly, use shuffle function: x,y = shuffle(x,y), to disrupt the order of the original digits.

Secondly, I call the scikit-learn library’s KNN classifier to train the data.

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(X\_train,y\_train) // put the training data into the classifier

knn.score(X\_train,y\_train) // The training accuracy

knn.score(X\_test, y\_test) // The testing accuracy

Thirdly, the method trainSklearnKNN() is created to train the dataset with KNeighborsClassifier. I first chose k=3, the training accuracy is 0.9923451635351427 and the testing accuracy is 0.9861111111111112. Then I iterate the value of k from 1 to 9 and drew the graph (see Graph 2 in F4). This process takes around 5 seconds.

1. save the model so that it can be called later:

save the model into file SKlearnKnnModel

**F3:** Call the method: trainMyKNN() in my program

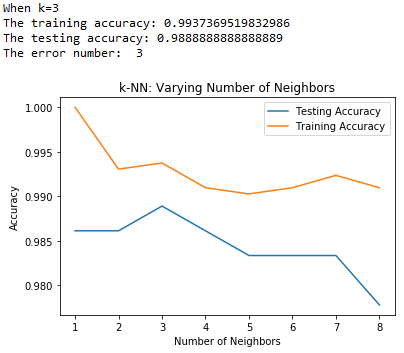
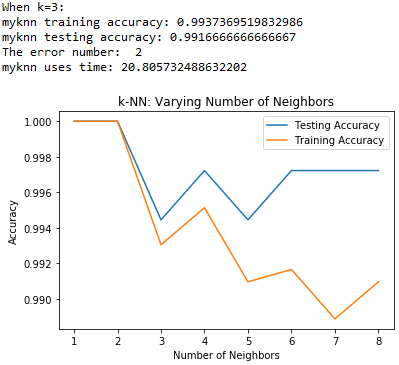
1. train a model with my own KNN algorithm by myself

Firstly, I created a class named KNNClassfier. In this class, the method predict(self,X\_test) is used to predict the digits in the training dataset. Euclidean Distance is used in this method to predict the target of each digit. The variable Output is set as the return value of this method. In the score(self,x,y) method, the return value Output of the method predict(self,X\_test) is compared with the real target of each digit. The method score(self,x,y) will return the accuracy of my knn algorithm.

Secondly, the method trainMyKNN() is created to train the dataset with my KNNClassfier. I first chose k=3, the training accuracy is 0.9909533750869868 and the testing accuracy is 0.9916666666666667. Then I iterate the value of k from 1 to 9 and drew the graph (see Graph 1 in F4). This process takes around 30 seconds.

1. save the model so that it can be called later: The same with that in F2

**F4:** The train accuracy and test accuracy will be printed after calling LoadSKLearnModel() and LoadMyKNNModel()

1. compare the train error and test error of the two models

d

Graph 1: my knn algorithm Graph 2: SKlearn knn algorithm

Graph 1 is not presented in the program because it takes 2 minutes to iterate from k=1 to k=8. In my program, I showed the accuracy of my knn algorithm when k=3. As we can see from the graph, overfitting doesn’t exist and the average accuracy is higher than 0.98. It is suitable to train the digits dataset with knn algorithm. The error number for each model is also presented.

**F5:** Call the method: QueryMyknnModel() and QuerySklearnknnModel()

1. enable the user to query the saved models with an index of the test dataset
2. Query the saved model of sklearn knn algorithm.

I wrote this part of functionality in the method of QuerySklearnknnModel (). The saved file is loaded at the beginning of this method.

Firstly, the user is asked to input a number. Input validation is included in the method. Secondly, because of the fact that knn doesn’t have model, the predict method in SKlearn library is called to predict the target of the selected digit. In the end, the image of this digit will be printed and the real target will be provided.

1. Query the saved model of my knn algorithm.

I wrote this part of functionality in the method of QueryMyknnModel().

The same with the way to query the model with sklearn knn algorithm.

**Additional Requirements:**

I created a user interface for user to choose from different options so that they can run my code directly, i.e., see the results of functionalities f1, f4, and f5 by loading the saved models, without calling the training functionalities f2 and f3. Clear instructions on how to train the two models are also provided in this user interface.

***3. Providing the details of your implementation, including e.g., the meaning of parameters and variables, the idea of your algorithm, etc.***

1. the idea of my algorithm:

I chose Optical recognition of handwritten digits dataset, which is trained by k nearest neighbor algorithm. Scikit-learn algorithm from the machine learning libraries is called to train this machine learning model. I also created my own knn algorithm to train the dataset. Euclidean Distance is used to predict the target of each digit. Overfitting doesn’t occur in my program, as can be shown in the graphs displayed in section 2, F4. The average training and testing accuracy are all above 0.98, which is comparably high with decision tree learning and naive Bayes. The order of the original digits is disrupted at the beginning.

1. the meaning of parameters and variables:

KNeighborsClassifier means the classifier for sklearn knn algorithm

x = digits.data, x means all the data from digits dataset

y = digits.target, y means the target of each data from digits dataset

X\_train is the training data, X\_test is the testing data, y\_train is the target for the training data, y\_test is the target of the testing data

myknn\_start\_time means the start time for my knn algorithm to run

myknn\_end\_time means the time for my knn algorithm to finish