Write up for programming exercise

SYDE 675

Programming Assignment 4

Zhiqi Bei

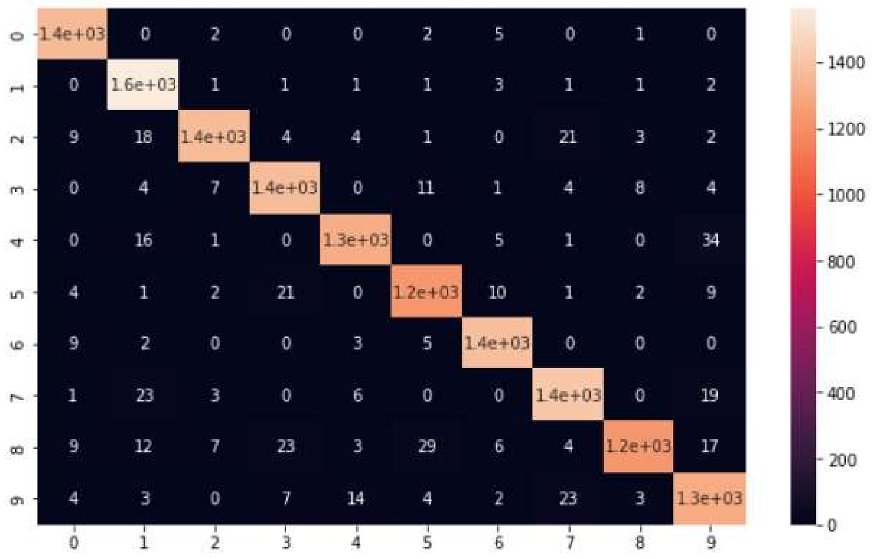
Wenhan Liu

Write up for exercise 1

For this Assignment, the train_data was first loaded using numpy and its dimension and the dimension of each image were checked. Then, by using a for loop and reshape function, the image of each digit was printed. Next, by setting the range of x-axis from 0 to 9 and range of y-axis from 0 to 0.12, a normalized histogram was plotted with the use of the weight parameter in plt.hist function. Then, by finding the index of each digit in the dataset, the distance of each digit in dig_list with every other digit in the dataset was calculated, and from the distance list, the datapoint with smallest distance was found and displayed along with the sample datapoint. Then, by finding the zeros and ones data in the dataset, the zero to zero, one to one, and zero to one distance were calculted and resulted in genuine and impostor distances, and a histogram about them was then plotted. Next, with the use of a for loop, the ROC Curve was plotted, and the two different error rates were shown. Then, a KNN was implemented, the distance between datapoint and every other data in the dataset was calculated, and the k smallest were chosen, next by equally voting, the final decision was made, and this was done by using 3 for loops. Finally, by separating the training_data to train and label, and 3-fold cross validation was performed.

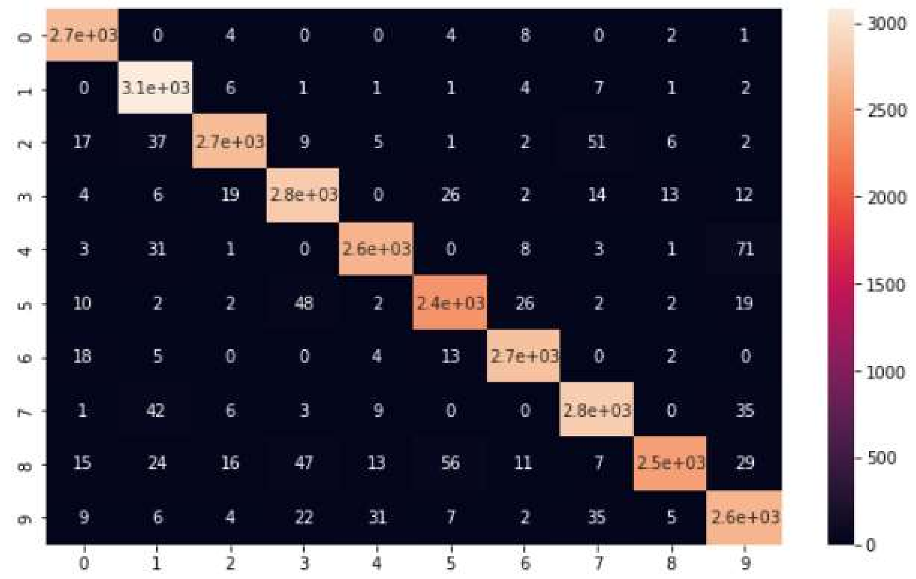
The accuracy along with the heatmap for the 3 trials are as follows:

```
NO.1 CV Accuracy is: 96.6%, Time Consumed: 3315.19s
```



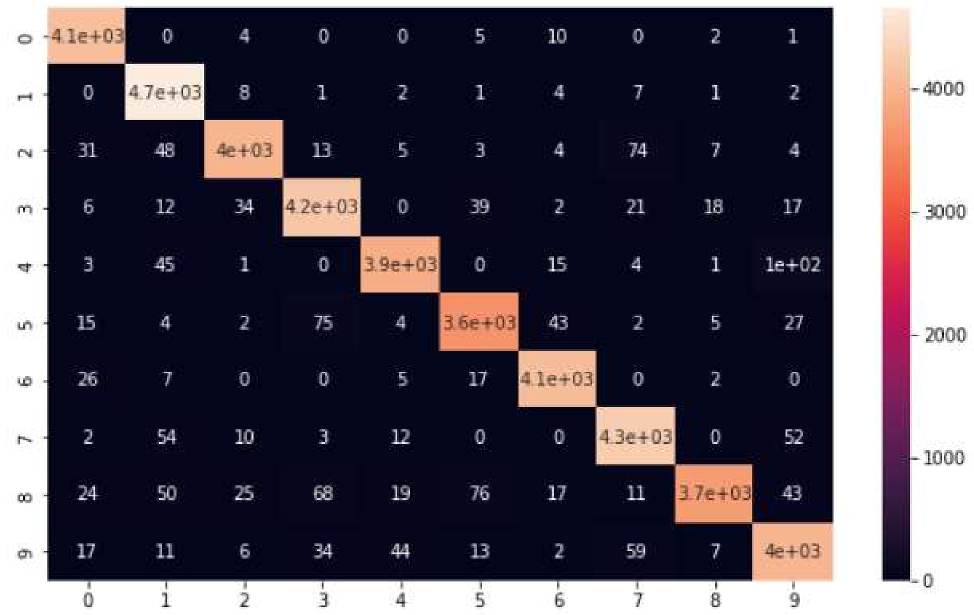
	0	1	2	3	4	5	6	7	8	9
0	1.4e+03	0	2	0	0	2	5	0	1	0
1	0	1.6e+03	1	1	1	1	3	1	1	2
2	9	18	1.4e+03	4	4	1	0	21	3	2
3	0	4	7	1.4e+03	0	11	1	4	8	4
4	0	16	1	0	1.3e+03	0	5	1	0	34
5	4	1	2	21	0	1.2e+03	10	1	2	9
6	9	2	0	0	3	5	1.4e+03	0	0	0
7	1	23	3	0	6	0	0	1.4e+03	0	19
8	9	12	7	23	3	29	6	4	1.2e+03	17
9	4	3	0	7	14	4	2	23	3	1.3e+03

```
----------------
NO.2 CV Accuracy is: 96.4%, Time Consumed: 3268.08s
```



	0	1	2	3	4	5	6	7	8	9
0	2.7e+03	0	4	0	0	4	8	0	2	1
1	0	3.1e+03	6	1	1	1	4	7	1	2
2	17	37	2.7e+03	9	5	1	2	51	6	2
3	4	6	19	2.8e+03	0	26	2	14	13	12
4	3	31	1	0	2.6e+03	0	8	3	1	71
5	10	2	2	48	2	2.4e+03	26	2	2	19
6	18	5	0	0	4	13	2.7e+03	0	2	0
7	1	42	6	3	9	0	0	2.8e+03	0	35
8	15	24	16	47	13	56	11	7	2.5e+03	29
9	9	6	4	22	31	7	2	35	5	2.6e+03

```
----------------
NO.3 CV Accuracy is: 96.6%, Time Consumed: 3287.70s
```

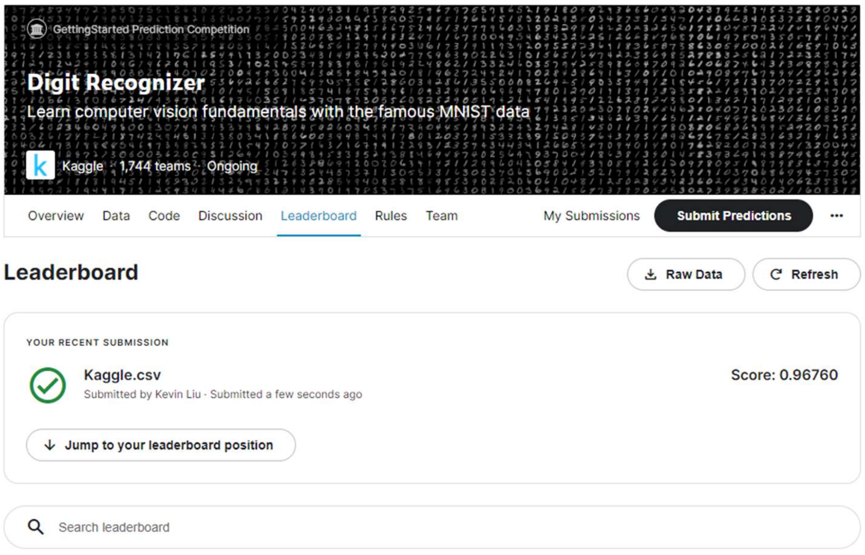


	0	1	2	3	4	5	6	7	8	9
0	4.1e+03	0	4	0	0	5	10	0	2	1
1	0	4.7e+03	8	1	2	1	4	7	1	2
2	31	48	4e+03	13	5	3	4	74	7	4
3	6	12	34	4.2e+03	0	39	2	21	18	17
4	3	45	1	0	3.9e+03	0	15	4	1	1e+02
5	15	4	2	75	4	3.6e+03	43	2	5	27
6	26	7	0	0	5	17	4.1e+03	0	2	0
7	2	54	10	3	12	0	0	4.3e+03	0	52
8	24	50	25	68	19	76	17	11	3.7e+03	43
9	17	11	6	34	44	13	2	59	7	4e+03

```
----------------
Average Accuracy is: 96.5%, Total Time Consumed: 9872.53s
```

With each datapoint in the testing set, KNN was used to perform the prediction and the accuracy was calculted. From the heatmap that got plotted, we could see that the digit 5 is particularly tricky to classify.

And finally, the Kaggle results are as shown below:

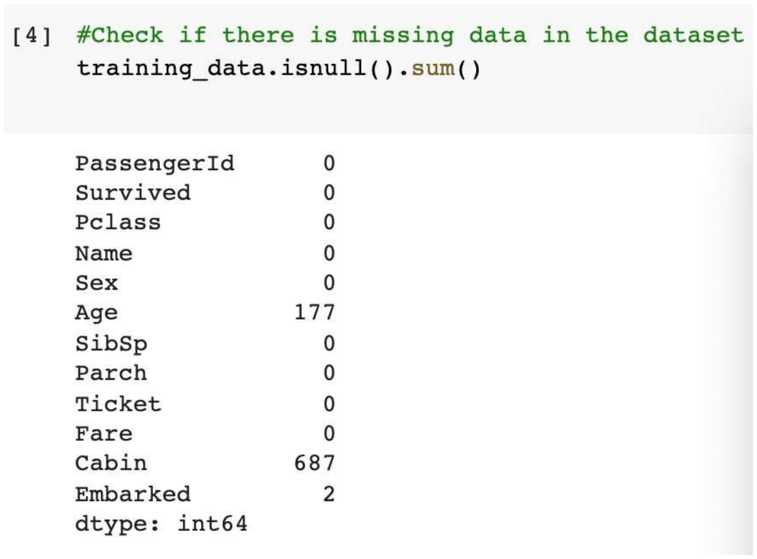

GettingStarted Prediction Competition
Digit Recognizer
Learn computer vision fundamentals with the famous MNIST data
Kaggle · 1,744 teams · Ongoing
Overview
Data
Code
Discussion
Leaderboard
Rules
Team
My Submissions
Submit Predictions
Leaderboard
Raw Data
Refresh
YOUR RECENT SUBMISSION
Kaggle.csv
Submitted by Kevin Liu · Submitted a few seconds ago
Score: 0.96760
Jump to your leaderboard position
Search leaderboard

Write up for exercise 2

Loading data

First is loading the data, we name the data from train.csv as training_data, and the data from test.csv as testing_data.

After loading the data, we can find that there are a lot of missing data in the training dataset.

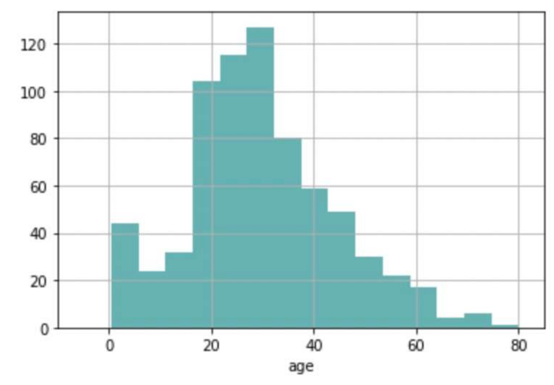


```
[4] #Check if there is missing data in the dataset
    training_data.isnull().sum()

    PassengerId      0
    Survived         0
    Pclass           0
    Name             0
    Sex              0
    Age            177
    SibSp            0
    Parch            0
    Ticket           0
    Fare             0
    Cabin          687
    Embarked         2
    dtype: int64
```

“Age” has 177 missing data, “Cabin” has 687 missing data, and “Embarked” has 2 missing data, so next we need to process this data.

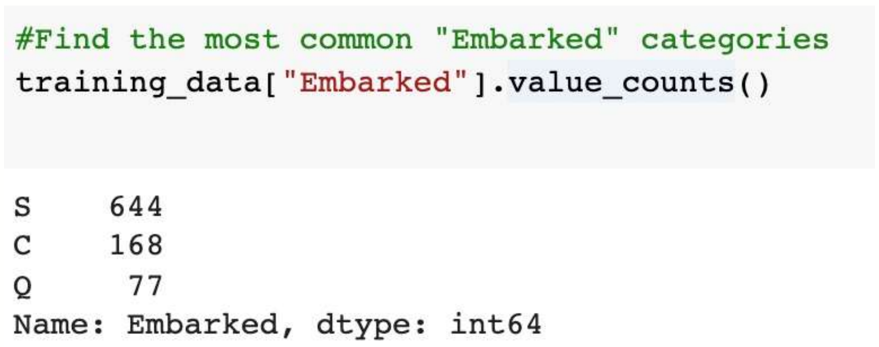
Process the missing “Age” data in the training set



By visualizing the “Age” data, we can see from the above graph, the skewness is not 0, then using the median of age to fill the missing data is better than using the mean. Therefore, we choose to use the median to fill in missing data, which is 28.

Process the missing “Cabin” data in the training set
Since the “cabin” is missing too much data, we decided to remove this feature

Process the missing “Embarked” data in the training set

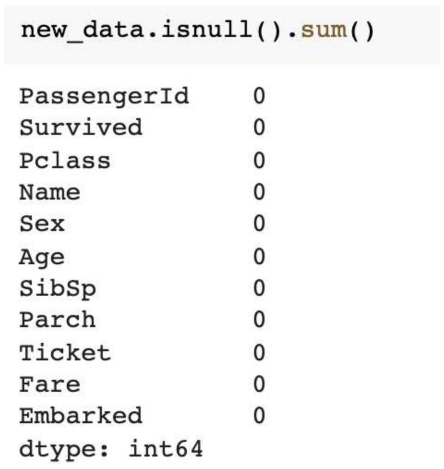


```
#Find the most common "Embarked" categories
training_data["Embarked"].value_counts()
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

We observed that there are 3 boarding ports, and most of the people are from port S. Therefore, we choose “S” to fill the missing “Embarked” data.

Check the final training dataset



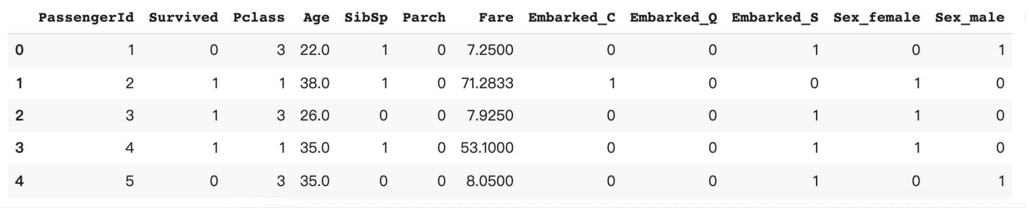
```
new_data.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

After processing all the missing data, we check the new taring data to see if there still has missing data. There is no missing data, so we go to the next step.

Process other features in training set

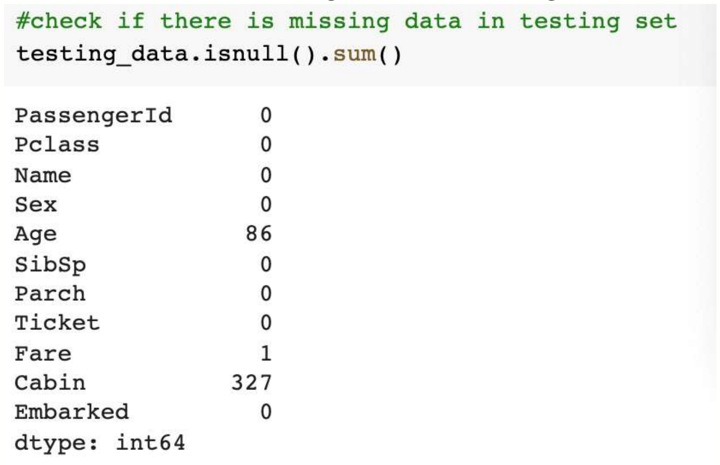
Most people have different names and ticket number, so we decided to remove those data. And using one-hot encoding convert "Embarked" and "Sex" to number. Then the final training data set is:



	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Embarked_C	Embarked_Q	Embarked_S	Sex_female	Sex_male
0	1	0	3	22.0	1	0	7.2500	0	0	1	0	1
1	2	1	1	38.0	1	0	71.2833	1	0	0	1	0
2	3	1	3	26.0	0	0	7.9250	0	0	1	1	0
3	4	1	1	35.0	1	0	53.1000	0	0	1	1	0
4	5	0	3	35.0	0	0	8.0500	0	0	1	0	1

Process the missing data in testing set

There are also have missing data in the testing set.

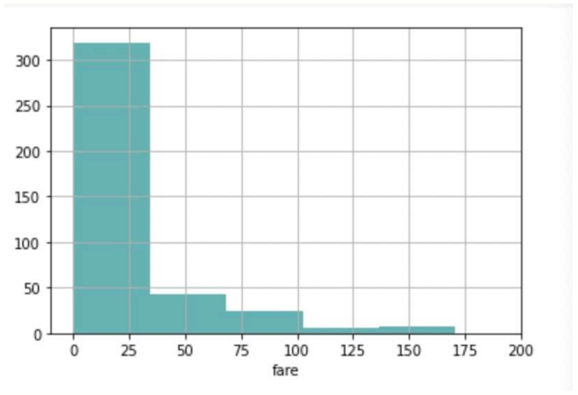


```
#check if there is missing data in testing set
testing_data.isnull().sum()
```

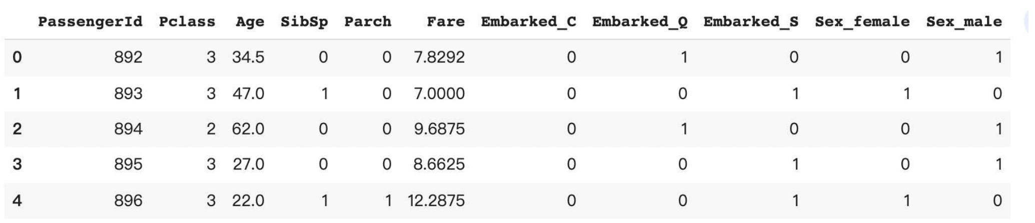
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

"Age" has 86 missing data, "Fare" has 1 missing data, and "Cabin" has 327 missing data. Similarly, as the training set, we use the median to fill the missing data in "Age" and remove the "Cabin" feature.

For the missing “Fare” data, we visualize the “Fare” data and observe that the skewness is not 0, so we choose the median of fare to fill the missing data.

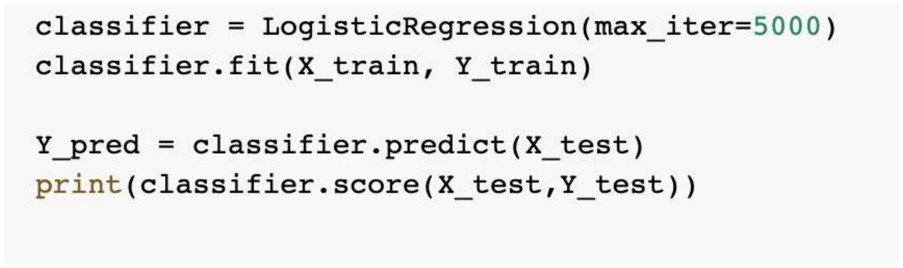


At last, we also remove the “Name” and “Ticket” from the testing set. The final testing set is as below:



	PassengerId	Pclass	Age	SibSp	Parch	Fare	Embarked_C	Embarked_Q	Embarked_S	Sex_female	Sex_male
0	892	3	34.5	0	0	7.8292	0	1	0	0	1
1	893	3	47.0	1	0	7.0000	0	0	1	1	0
2	894	2	62.0	0	0	9.6875	0	1	0	0	1
3	895	3	27.0	0	0	8.6625	0	0	1	0	1
4	896	3	22.0	1	1	12.2875	0	0	1	1	0

Logistic regression
Then we can do the logistic regression, we first train the classifier using the training data, we split the training set into X_train, X_test, Y_train, Y_Test by 8:2, 8 is the training and 2 is the testing.



```
classifier = LogisticRegression(max_iter=5000)
classifier.fit(X_train, Y_train)

Y_pred = classifier.predict(X_test)
print(classifier.score(X_test,Y_test))
```

```
0.7988826815642458
```

The prediction accuracy is 0.79

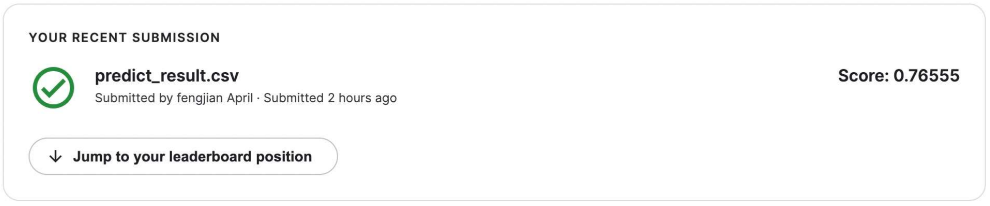
Train the classifier using all the training data
Then we can use all the training data to train the classifier and test it using the testing data from the test.csv and write down the result to an csv file.

```
new_classifier = LogisticRegression(max_iter=5000)
new_classifier.fit(X,Y)
X_test = new_test[['Pclass','Age','SibSp','Parch','Fare','Embarked_C','Embarked_Q','Embarked_S','Sex_female','Sex_male']]
Y_pred = classifier.predict(X_test)


result = pd.DataFrame({'PassengerId':new_test['PassengerId'],'Survived':Y_pred})
result.head()
result.to_csv('./titanic/predict_result.csv',index=False)
```

Submit to Kaggle

At last, we submit this file to Kaggle, and below is the score.



YOUR RECENT SUBMISSION

predict_result.csv
Submitted by fengjian April · Submitted 2 hours ago

Score: 0.76555