CS 5785 Applied Machine Learning
Assignment 1

Kelly Wang
and Yian Mo
09/12/2017

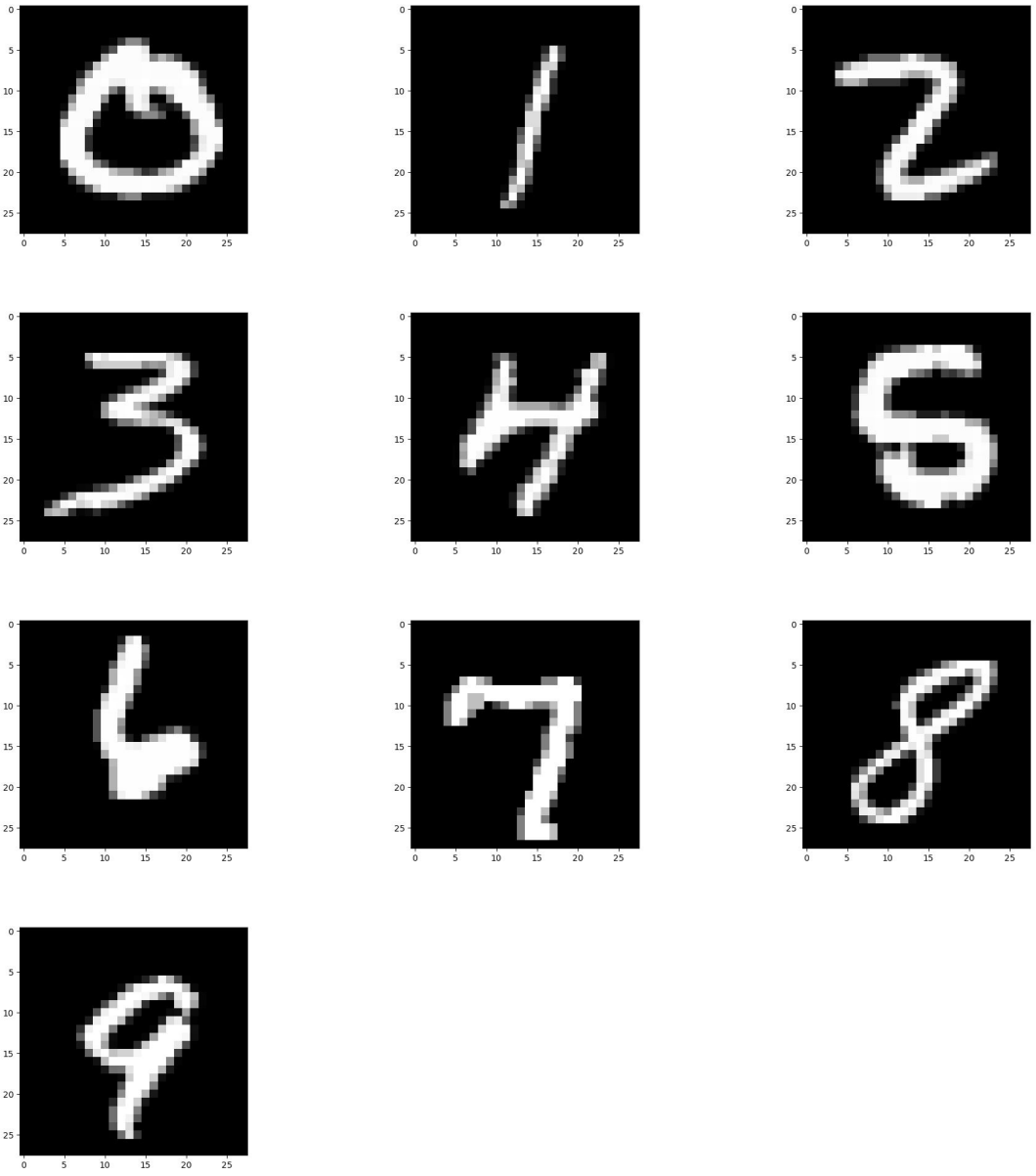
Summary

During this assignment, we explored the use of KNN algorithm and Logistic Regression on the dataset MNIST and Titanic from Kaggle challenges. We implemented our own KNN classifier and used it to train the datasets and predict the test data. We also learnt many useful functions from libraries such as Sklearn, Panda, Seaborn to help us simplify our code.

Procedure

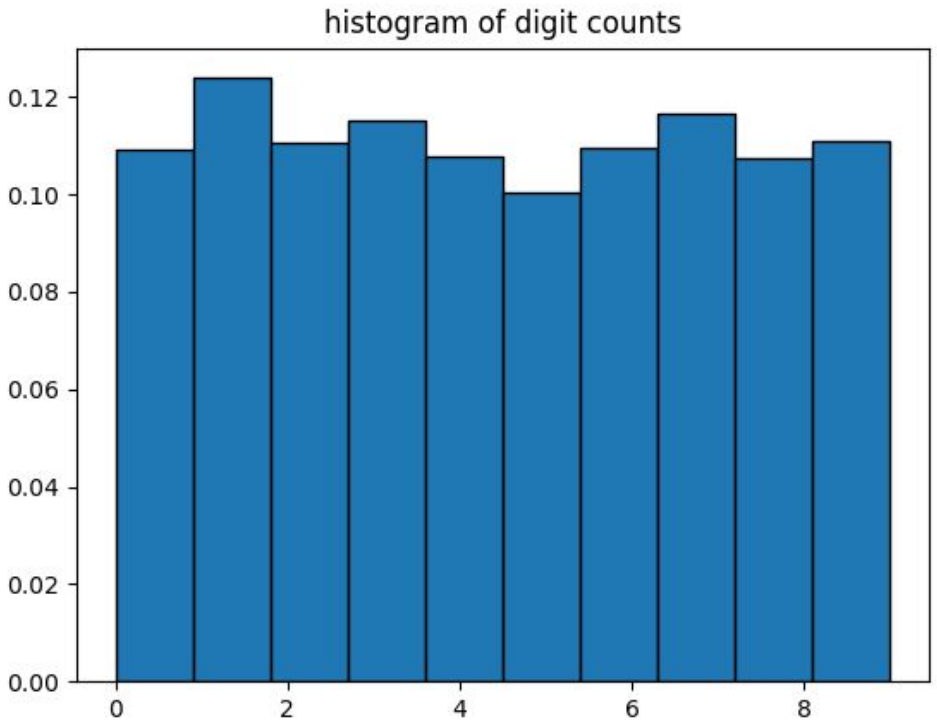
1. Digital Recognizer

(b) Sample digits

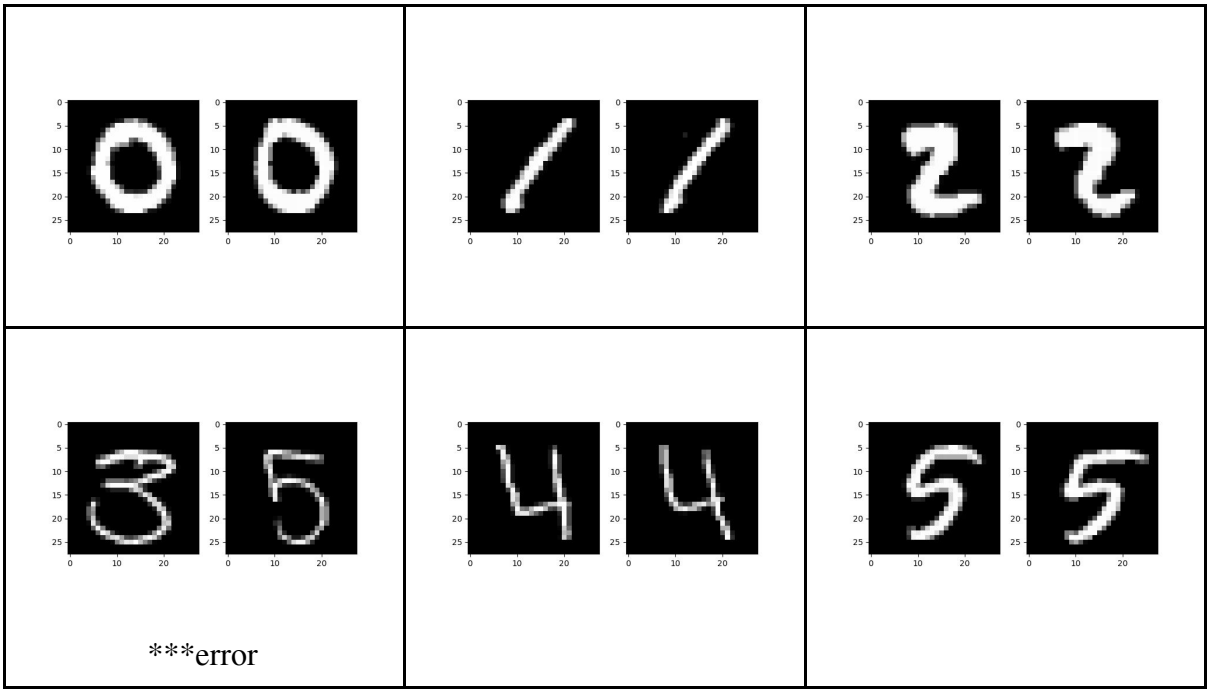


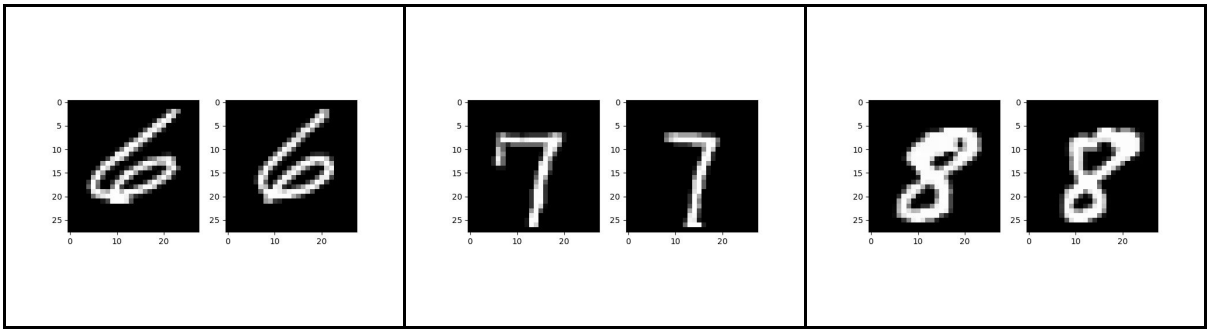
(c) The prior probability of the classes in the training data is calculated via the number of that digits appeared in the data divided by the total number of samples available. And the result is 0.09838095, 0.11152381, 0.09945238, 0.10359524, 0.09695238, 0.09035714, 0.0985, 0.10478571, 0.0967381, 0.09971429 for digits 0 to 9 respectively.

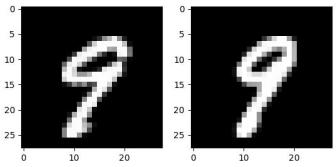
Figure below is a normalized histogram of digit counts. As it can be seen, the diagram is quite even.



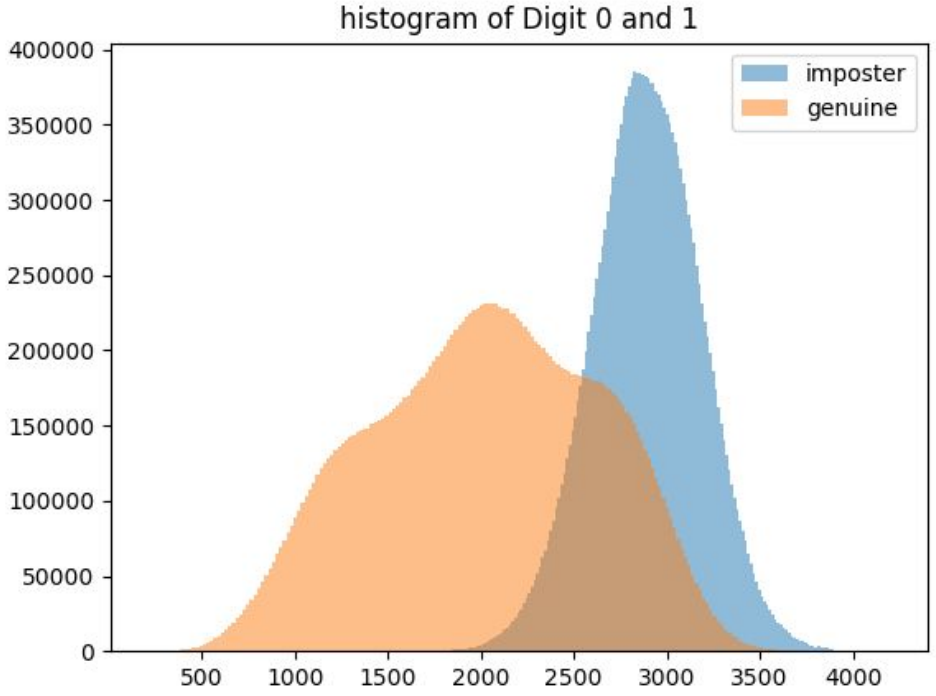
(d) See function findBestMatch() and plotNearSample(). Below displays a set of sample digits and their corresponding best match. The result shows that there is an error appeared for digit 3.





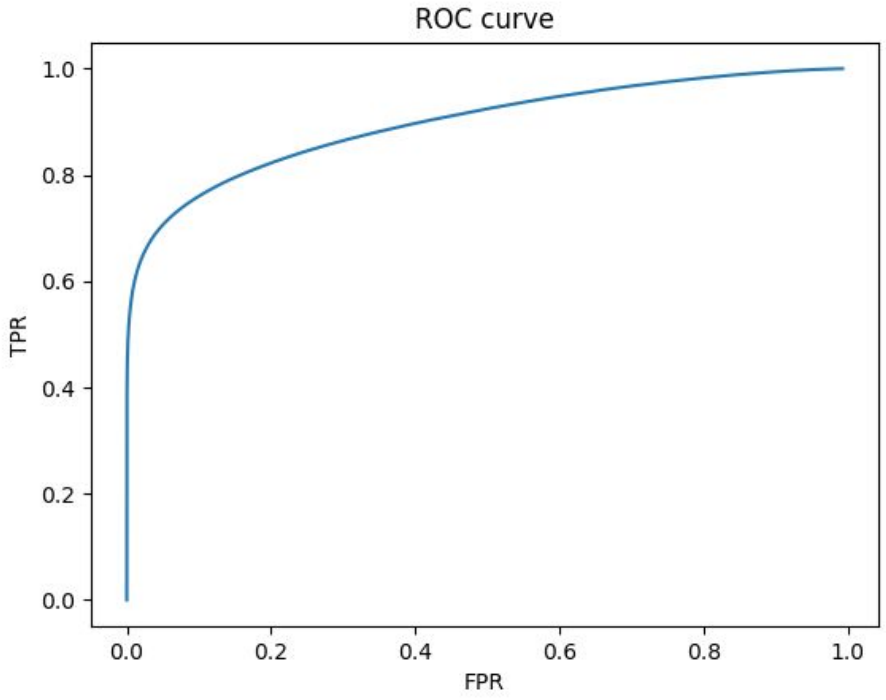


(e) See function questionE(trainLabel,trainData, questionelist)
We recorded all the indices of digit 0 and digit 1 in the training dataset, and calculated the L2 norm distance for all the possible combinations which are then subsequently stored in either impostor or genuine distance sets accordingly. The resulting histogram is displayed below. The x-axis is the distances between pairs of digits. It shows that the distances in the impostor cases are generally larger than the genuine ones, with some overlapping cases in between. Hence, choosing a proper threshold value is important.



(f) See function generate_roc(ImposterDis,GenuineDis) in script digitRecognizer.py
An ROC curve is generated via calculating the true positive rate and false positive rate while shifting the threshold bar across the histogram. The eer (equal error rate) is the intersection of the roc curve and the diagonal line from (0,1) to (1,0) and it is found to be 0.185. At this

point, the false positive rate equals to the false negative rate. The error rate of a classifier that simply guesses randomly is 0.498.



(g) See KNN.py

The KNN function is the main function. The getNeighbors function is used to calculate the Euclidean distance and find the k nearest neighbor. The getReponse() function is to find the majority votes and return the predicted class. The getAccuracy function is used to calculate the accuracy rate after calculating predictions.

(h) We performed 3 fold cross-validation on the K-NN classifier. The resulting accuracy for three folds are 0.96636675, 0.96428061, and 0.96585226 respectively. Averaging it gives the final accuracy, which is 0.965499876161.

(i) Confusion matrix is

```
array([[4109,    0,    5,    0,    0,    5,   10,    0,    2,    1],
       [   0, 4659,    8,    1,    1,    1,    4,    7,    1,    2],
       [  31,   48, 3989,   13,    4,    3,    4,   74,    7,    4],
       [   6,   12,   34, 4202,    0,   39,    2,   21,   18,   17],
       [   3,   45,    1,    0, 3897,    0,   15,    4,    0,  107],
       [  15,    4,    2,   75,    4, 3618,   43,    2,    5,   27],
       [  26,    7,    0,    0,    5,   16, 4081,    0,    2,    0],
       [   2,   54,   10,    3,   12,    0,    0, 4268,    0,   52],
       [  24,   49,   26,   67,   20,   74,   17,   11, 3732,   43],
       [  16,   10,    5,   34,   46,   13,    2,   59,    7, 3996]])
```

If we look at the confusion matrix, we can see that, along the diagonal, the entries are all quite big. This makes sense as it shows that digits would mostly be recognized correctly as themselves rather than others. However, there are a few tricky digits to classify. For example, digits 4 and 9 are miss-classified to be each other for many times. Same thing happens for digits 3 and 5.

(j)

2. Titanic

We glanced through all the parameters and first omitted the parameters that we found irrelevant for predicting whether a person survived or not, which includes a person's name, ticket number, and passenger ID number. These are factors that don't have any influence on person's possibility to survive. The next step is to fill out any empty entries in the column that we think is relevant to the situation. From the data, it can be seen that some entries inside the age are empty, so a function to input the missing value was created. Inside this function, the missing ages are determined by the average of the ages of all the person in that particular class. The sex variable is also quite important as women are more likely to be rescued then men, so a dummy variable is used to represent the sex by using 1 to indicate the person is a male and 0 to represent female. Last, to reduce the training time, some related variables can be omitted. By creating a heatmap between fare and PClass, it is proved that they are not independent of each other, hence the fare variable was left out during the training.

The processed training data was then used to train the logistic regression model and the prediction is returned as:

```
array([1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1])
```

Uploading the result onto Kaggle, we got an accuracy of 0.77033.

Conclusion:

We used different models to fit different kinds of datasets and the results returned are almost as what we expected. The KNN algorithm used to classify the MNIST digits has a fairly high performance, reaching an accuracy of 96%. The Titanic problem, which is hard to tackle as it has got more features, is solved via logistic regression, and the final prediction resulted in an accuracy of 77%.

WRITTEN EXERCISES

1. Proof:

var [X −Y] = cov(X -Y, X-Y) = E(X^2 -2XY+ Y^2) − E(X-Y)E(X-Y) = E(X^2) - $(E(X))^2$ + E(Y^2)+ $(E(Y))^2$ + 2(E(XY) - E(X)E(Y)) = var[X]+var[Y] -2cov(X,Y)

2. (a) Let A= actually defective
   B= actually not defective
   C= test shows defective
   D = test shows not defective

$$P(A \mid C) = \frac{P(C|A)\,P(A)}{P(C)} = \frac{P(C|A)\,P(A)}{P(C|A)P(A) + P(C|B)P(B)}$$

$$= \frac{0.95\times\frac{1}{100000}}{0.95\times\frac{1}{100000} +0.05\times(1-\frac{1}{100000})} = 0.000189965$$

(b) Total production per year = 10 million

$$P(B \cap C) = P(C|B)P(B) = 0.05 \times (1 - \frac{1}{100000})$$

# of good widgets being thrown away= total production $\times$ P(B ∩ C) =500000-5 =499995

$$P(A \cap D) = P(D \mid A)\,P(A) = 0.05 \times \frac{1}{100000}$$

# of bad widgets shipped to customers= total production $\times$ P(A ∩ D) =5

3. (a)When k equals to n, since two classes have the same number of training points, it's a tie. As the KNN would normally randomly assign a class to break the tie. The average prediction error in this case is ½. If k = 1, it would always be classified right as itself is included in the training set, hence the prediction error is zero. Overall, as k varies from n to 1, the prediction error will gradually decrease.

   (b)The graph should look like "U". When the k is very small, the error rate should be very high because there aren't enough neighbors to calculate majority. Then when the k increases, the error rate decreases. When k equals N/2, the error rate became the largest since the majority votes became the other wrong class.

   (c)When the k increases, the computational requirements increases exponentially. When the k is very small, it may become inaccurate because of inadequate number of neighbors. Based on our experiments we found that when k=3 to k=5, it reaches the best tradeoff between the runtime and accuracy. So I would suggest using k=3.

   (d)We can give the weights to different neighbors while the closer ones have greater weights. For example, we can assign weight w=1/d, d is the distance.

   (e) Two similar points may have long distance between them because of the high dimensions.
   Matrix operation very expensive.
   The prediction may largely be influenced by the class with highest frequency.