Homework 1

Haomiao Han (hh696), Hyein Baek (hb437)
CS5785 Applied Machine Learning

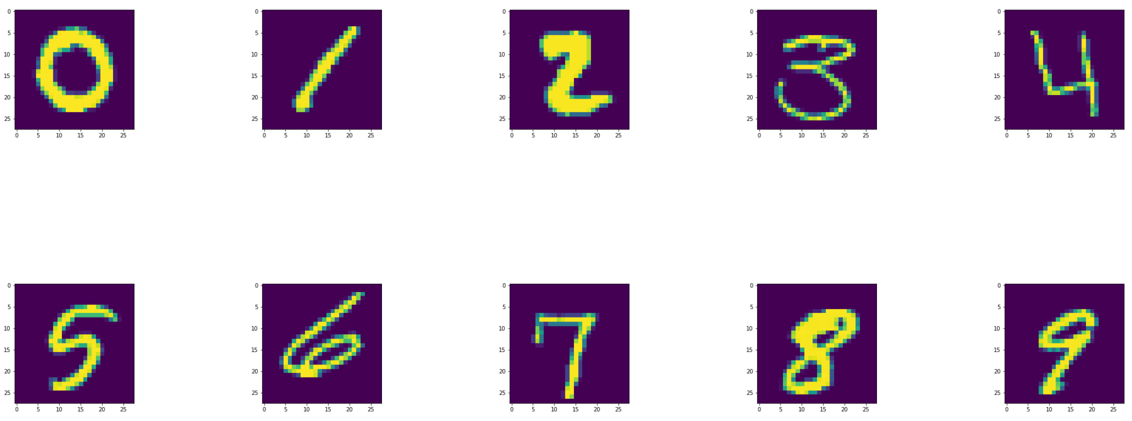
September 17, 2019

1 Programming Exercises

- Programming Exercises, Question 1

  (a) We used `pandas` to load and split the dataset.

  (b) We first defined a function, `show_digit`, to display a single digit given an input (0-9). We then called this function with all numbers from 0 to 9 as inputs, thereby displaying 10 digits. Currently, the function will only find and display the first occurrence of each digit. Additionally, an optional line of code that calls `sklearn`'s `utils.shuffle` function is included. This function will randomly shuffle the dataset so that we can see different digit images each time we run this program. Below are the images of the digits.



  (c) We counted the occurrences of each digit within the dataset, then plotted a histogram. We can see that the distribution of the digits is roughly even, with 1 having the most occurences throughout the dataset and 5 having the least occurences. The histogram is shown below:

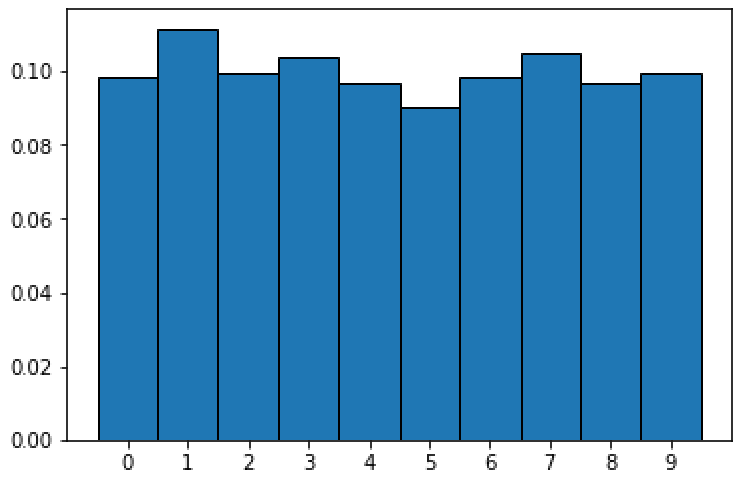
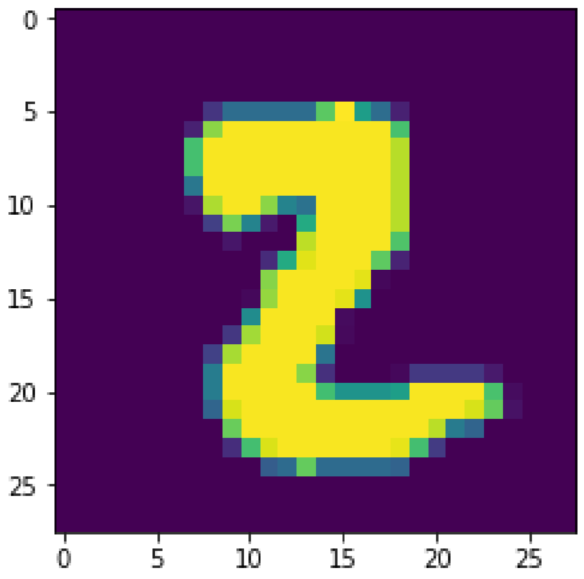
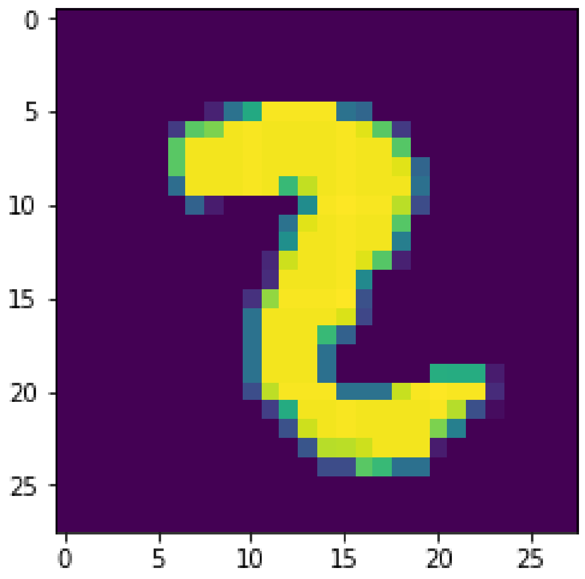


Figure 3: Normalized histogram of digit counts

(d) We picked the digits used in part (b), then used `sklearn`'s `neighbors.NearestNeighbors` function to find the 2 nearest neighbors for each digit. The reason why we are finding 2 nearest neighbors is that since each of the digits themselves are already inside the training set, its nearest neighbor must be itself, and its 2nd nearest neighbor is the digit that is most similar to itself. From the results, we can see that the most similar image to the image of digit 3 is actually a 5. All other digit images are matched with the same digit. As an example, we included the images of a digit 2 image and its nearest neighbor below. The distance between the 2 images is 1380.88. (The full results can be displayed by running the .ipynb file.)



(a) Image of digit 2, at index 16



(b) Image of digit 2, at index 9536

(e) We used `sklearn`'s `metrics.pairwise_distances` function to calculate the pairwise distances among 0 digits and among 1 digits (”genuine matches”) as well as the pairwise distances between 0 digits and 1 digits (”imposter matches”). We then generated a histogram using the distances above. From the histogram, we can see that ”imposter matches” generally have longer distances than ”genuine matches”. The histogram is shown below.

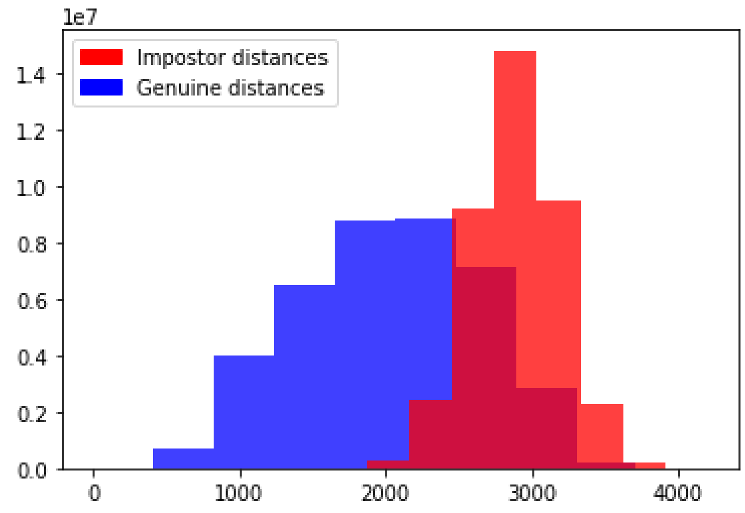


Figure 5: Histogram of ”genuine matches” and ”imposter matches”

(f) We first calculated the False Positive Rates and True Positive Rates of the prediction based on different thresholds, then plotted the ROC curve using these rates. The curve is shown below. The Equal Error Rate is around 0.2, as demonstrated by the intersection of the ROC curve (blue) and the orange dashed line. The Equal Error Rate for random prediction would be 0.5, as demonstrated by the intersection of the ROC curve for random prediction (green dashed line) and the orange dashed line.

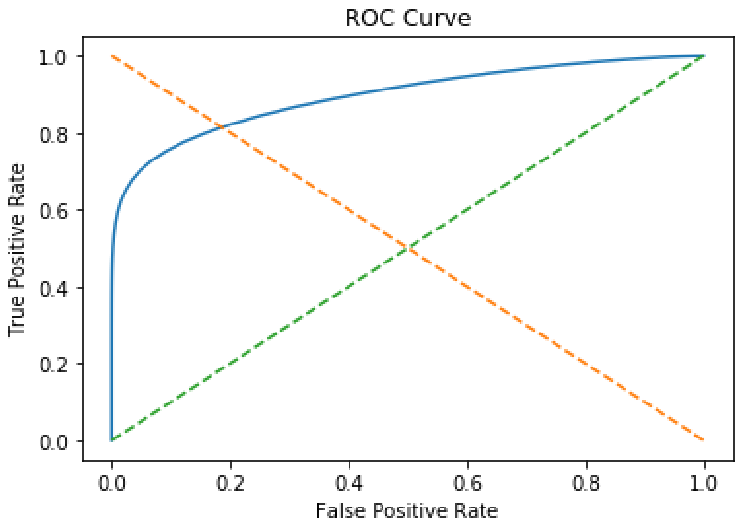


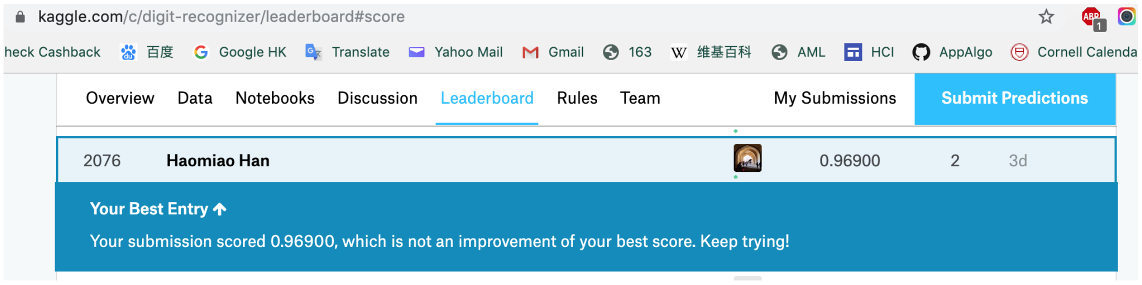
Figure 6: ROC curve

(g) We used `scipy`'s `cdist` function to calculate the distances between each point in test set and each point in training set. Then, using the calculated distances, we find the nearest `k` neighbors and predict the label of our target point given the labels of its neighbors. The total training and prediction time is around 5-10 minutes on our machines.

(h) Using our KNN algorithms, we achieved an accuracy rate of around 0.962.

(i) We created a confusion matrix using `sklearn`'s `metrics.confusion_matrix` function. We then calculated the correct and incorrect predictions for each digit (0-9). According to the data, digit 8 is more difficult to classify than other digits, with an accuracy rate of 0.917. Digit 5, with an accuracy rate of 0.944, is the second most difficult to classify.

(j) We used all of the training set to train our model and predicted on the test set. According to Kaggle, our predictions have an accuracy rate of 0.969. Below is the proof:


kaggle.com/c/digit-recognizer/leaderboard#score
heck Cashback 百度 Google HK Translate Yahoo Mail Gmail 163 维基百科 AML HCI AppAlgo Cornell Calenda
Overview Data Notebooks Discussion Leaderboard Rules Team My Submissions Submit Predictions

2076 Haomiao Han 0.96900 2 3d

Your Best Entry

Your submission scored 0.96900, which is not an improvement of your best score. Keep trying!

- Programming Exercises, Question 2

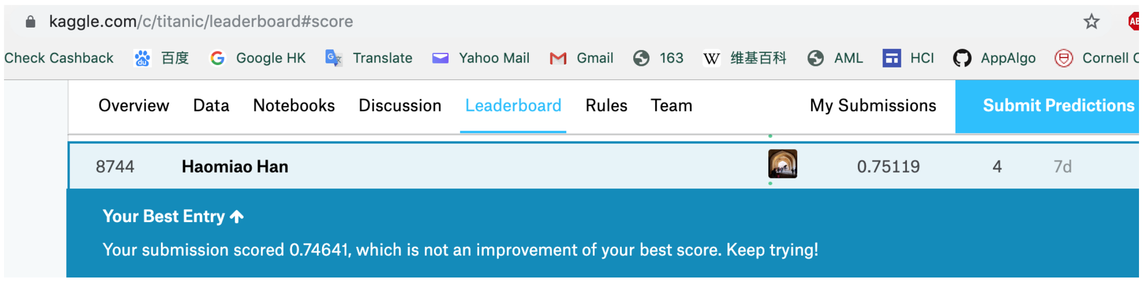
(a) We used `pandas` to load and split the dataset.

(b) We decided to drop the following columns in the dataset for prediction purposes: `PassengerId`, `Name`, `Ticket`, `Cabin`. `PassengerId` is simply used as an index and is unrelated to whether the passenger has survived or not (which is our prediction goal). Similarly, `Name` and `Ticket` (which contains the ticket number) are also unrelated to the prediction process. While `Cabin` may have explained the survival rate to some extent, we discovered that this column has a large percentage of missing data - 687 out of 891 cells (around 77.1%) are missing in the training set. Given the fact that we already knew the cabin class each passenger was in (through `Pclass`), it wouldn't make sense to fill the empty cells in this column. Therefore, this column was also dropped.

We then filled the empty cells in `Age` and `Fare` with the median value of all existing cells in each column, and in `Embarked` with the next valid cell ("backfill"). After that, we encoded the values in `Sex` and `Embarked` columns into numbers {0,1,2}, as the values are originally in a string format.

After that, we used `sklearn`'s `linear_model.LogisticRegression` to build a logistic regression model, using 75% of the training data as the training set and 25% as the validation set. Our average accuracy on the validation set is around 0.8.

(c) We used all of the training set to train our model and predicted on the test set. According to Kaggle, our predictions have an accuracy rate of 0.751. Below is the proof:


kaggle.com/c/titanic/leaderboard#score
Check Cashback 百度 Google HK Translate Yahoo Mail Gmail 163 维基百科 AML HCI AppAlgo Cornell C
Overview Data Notebooks Discussion Leaderboard Rules Team My Submissions Submit Predictions

8744 Haomiao Han 0.75119 4 7d

Your Best Entry

Your submission scored 0.74641, which is not an improvement of your best score. Keep trying!

2 Written Exercises

- Written Exercises, Question 1

$$\begin{aligned} var[X-Y] &= E[(X-Y)^2] - (E[X-Y])^2 \\ &= E[X^2] - E[2XY] + E[Y^2] - (E[X])^2 + 2E[X]E[Y] - (E[Y])^2 \end{aligned} \tag{1}$$

$$var[X] = E[X^2] - (E[X])^2 \tag{2}$$

$$var[Y] = E[Y^2] - (E[Y])^2 \tag{3}$$

$$\begin{aligned} 2cov[Y] &= 2E[XY] - 2E[X]E[Y] \\ &= E[2XY] - 2E[X]E[Y] \end{aligned} \tag{4}$$

Given (2), (3) and (4), we get:

$$\begin{aligned} &var[X] + var[Y] - 2cov[Y] \\ &\quad = E[X^2] - (E[X])^2 + E[Y^2] - (E[Y])^2 - E[2XY] + 2E[X]E[Y] \end{aligned} \tag{5}$$

Given that the terms on the right side of equation (1) and (5) are the same, the original equation is proved.

- Written Exercises, Question 2

(a)

Let $P(A|B)$ be the probability of a widget that is actually defective to be tested defective, $P(B|A)$ be the probability of a widget that is tested defective to be actually defective, $P(B)$ to be the probability of a widget to be actually defective, and $P(notB)$ to be the probability of a widget to be actually not defective.

Thus, we have:

$$P(A|B) = 0.95 \tag{6}$$
$$P(A|notB) = 0.05 \tag{7}$$
$$P(B) = 1/100000 = 0.00001 \tag{8}$$
$$P(notB) = 0.99999 \tag{9}$$

According to Bayes' Theorem, we have:

$$\begin{aligned} P(B|A) &= \frac{P(A|B) * P(B)}{P(A|B) * P(B) + P(A|notB) * P(notB)} \\ &= \frac{0.95 * 0.00001}{0.95 * 0.00001 + 0.05 * 0.99999} \\ &= \frac{0.0000095}{0.050009} \\ &= 0.000190 \end{aligned} \tag{10}$$

which is 0.019%.

(b)

All defective widgets: 10,000,000 * 0.00001 = 100

Actual Defective & Predicted Defective: 100 * 0.95 = 95

Actual Defective & Predicted Not Defective: 100 * 0.05 = 5

All non-defective widgets: 10,000,000 * 0.99999 = 9,999,900

Actual Not Defective & Predicted Not Defective: 9,999,900 * 0.95 = 9,499,905

Actual Not Defective & Predicted Defective: 9,999,900 * 0.05 = 499,995

Thus, 5 bad widgets are shipped to customers and 499,995 good widgets are thrown each year.

- Written Exercises, Question 3

  (a) As each point we are predicting exists in the training dataset, its nearest neighbor is itself. Thus, when k=0, the error rate would be close to 0. When k=n, given that 50% of the data in the dataset belong to one class and the rest belong to the other class, the error rate would be close to 0.5.

  (b) In this circumstance, we would need to find an optimal k value where the average error is minimized, which means that selecting any value larger than that optimal k value or any value smaller than that optimal k value would result in a larger average error. This could be demonstrated by the Figure 1 below.

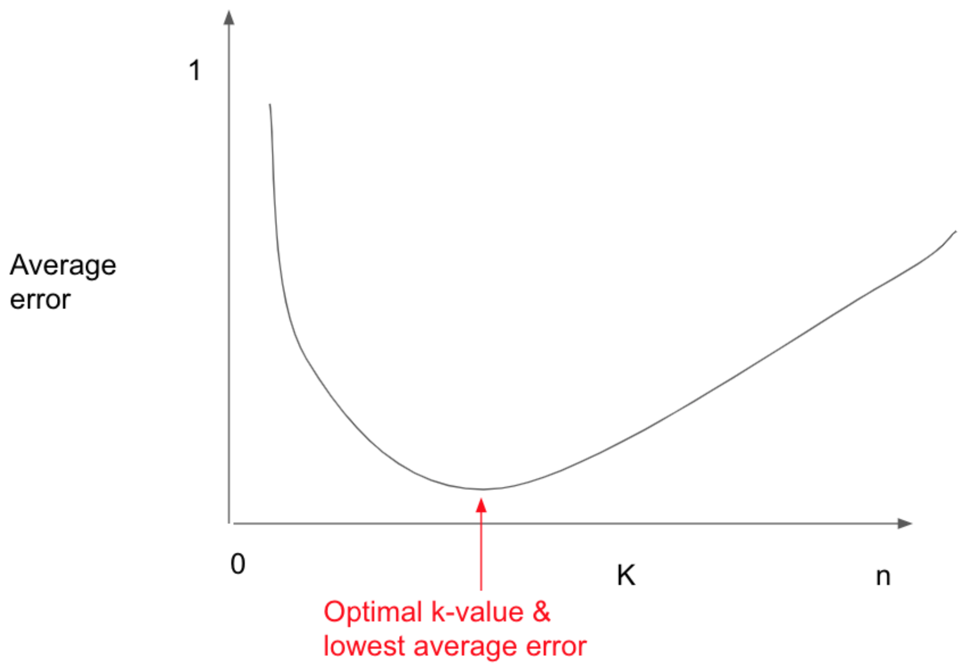


Figure 7: Rough sketch of a k-values and average error curve

(c) We could assign different "weights" to each point's vote, so that points closer to the point we are predicting have more "weights" than points farther to our point. For example, when we are deciding the label of our target point by vote, the vote of each of the nearest 10 points weighs 2 (i.e. their votes are counted twice) whereas the vote of any other point weighs 1. This will skew the prediction process so that the labels of the closer points will matter more.

(d)

1. "Curse of dimensionality" - as the input dimension increases, even the distances between 2 similar points will be large and the distribution of distances will have less spread, making the prediction less accurate.

2. The running time of knn will increase significantly as the dimension of the input data increases.

(e)

i.

Training data:

	Predicted Blue	Predicted Red
Actual Blue	8	3
Actual Red	3	6

Test data:

	Predicted Blue	Predicted Red
Actual Blue	4	1
Actual Red	2	3

ii.

Training data:

	Predicted Blue	Predicted Red
Actual Blue	6	5
Actual Red	4	5

Test data:

	Predicted Blue	Predicted Red
Actual Blue	3	2
Actual Red	4	1

iii.

For LogReg: Accuracy (training) = 14/20 = 0.7; Accuracy (test) = 7/10 = 0.7

For KNN: Accuracy (training) = 11/20 = 0.55; Accuracy (test) = 4/10 = 0.4

We can see that in both training and test dataset, logistic regression performs better than KNN in terms of accuracy. This might be because of the fact that k=1 is too low for this dataset so that the prediction is not general enough (as it only depends on the label of its nearest neighbor, and there are a relatively large amount of "outliers" in the dataset). Incrementing the k value might increase the accuracy for KNN model in this case.

3 References

- (Display images with matplotlib) `https://stackoverflow.com/questions/44755458/display-pixels-values-as-an-image`
- (Graphing histograms with matplotlib) `https://stackoverflow.com/questions/27083051/matplotlib-xticks-not-lining-up-with-histogram`
- (Equal Error Rate) `https://www.cs.ubc.ca/~murphyk/Teaching/CS340-Fall07/ROC.pdf`
- (Equal Error Rate) `https://stackoverflow.com/questions/28339746/equal-error-rate-in-python`
- (Bayes' Theorem) `https://betterexplained.com/articles/an-intuitive-and-short-explanation-of-bayes-theorem/`

- (kNN prediction error) `https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/`
- (Curse of dimensionality) `http://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html`