National University of Singapore
Faculty of Engineering
Department of Electrical and Computer Engineering

Project Report

Spam Data Recognition and Classification

Liu Sai A0177222J

Supervised By

Dr. Yeo Boon Thye Thomas

1. Summary

1.1 Introduction

Spam mail is any email that meets the following three criteria:

Anonymity: The address and identity of the sender are concealed,

Mass Mailing: The email is sent to large groups of people,

Unsolicited: The email is not requested by the recipients.

Spam email can be not only annoying but also dangerous to consumers. Many email spam messages are commercial in nature but may also contain disguised links that appear to be for familiar websites but in fact lead to phishing web sites or sites that are hosting malware. As people using more and more email, they'll consistently be frustrated by having to fight numerous spam emails. With data becoming more and more important today, spam email detection is of great importance for personal or financial security, even for the whole world.

1.2 Objectives

In this project, we'll design classifiers that can distinguish spam email from normal email to help solving this problem, including Beta-bernoulli Naïve Bayes classifier, Gaussian Naïve Bayes classifier, Logistic Regression classifier, K-Nearest Neighbors classifier. The results are demonstrated by graph and chart with data. Some principles and reasons are also discussed.

2. Data Processing

The data is an email spam dataset, consisting of 4601 email messages with 57 features. Feature descriptions are found in:

https://web.stanford.edu/~hastie/ElemStatLearn/datasets/spam.info.txt

We have divided the data into a training set (3065 emails) and test set (1536 emails) with accompanying labels (1 = spam, 0 = not spam).

Three means for data processing is used in this project, they are shown below:

Binarization: binarize features: I (xij > 0)

Z-normalization: standardize each column so they have 0 mean and unit variance. Use empirical mean and empirical variance.

Log transform: transform each feature using log (xij + 0.1) (assume natural log)

3. Beta-bernoulli Naïve Bayes

3.1 Algorithm

For this classifier, we use binarized data.

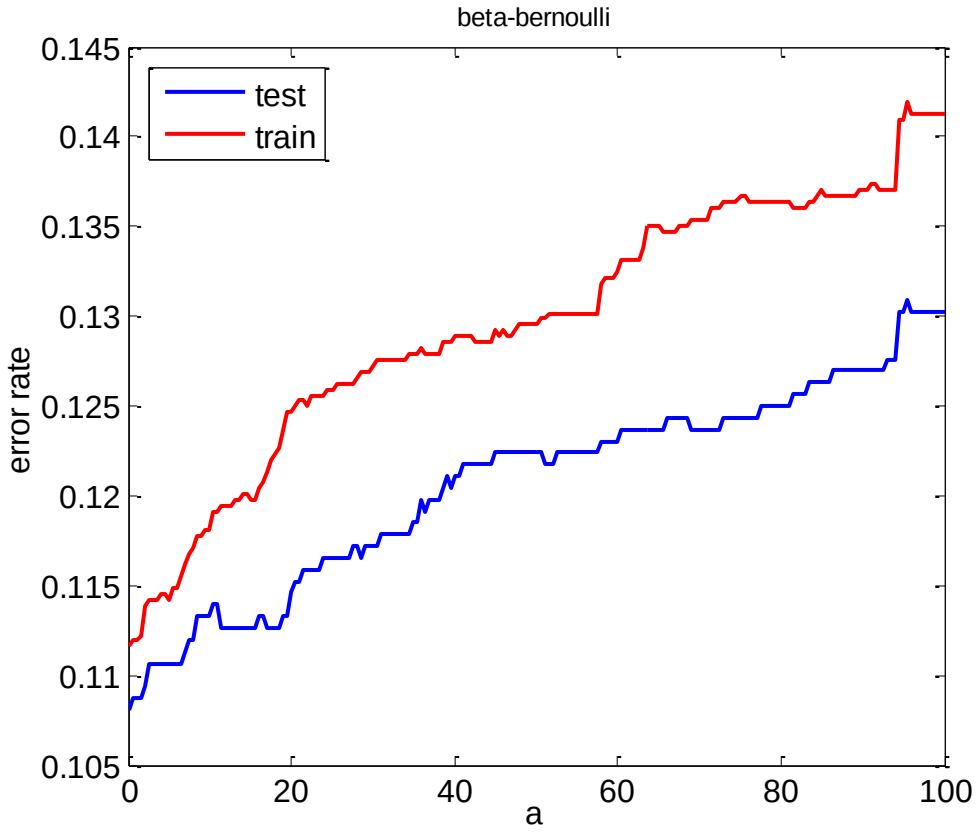
Pseudocode:

Algorithm1: Beta-bernoulli Naïve Bayes
Input: Spam Data Output: error_rate

1. Load data
2. Data processing
3. Calculate posterior of y (π_c) with train data (in this project, we use ML)
4. Calculate posterior predictive distribution of x (Beta-binomial) with train data.
5. Test with test set.
6. Get error rate by using test label.

3.2 Results

After training, the results are shown below:



3.3 Remarks

- Remark 1:

The error rate increases with the a increasing, and the error rate of test set is lower than the training set.

- Remark 2:

a	1	10	100
Test	10.8724%	11.3281%	13.0208%
Train	11.1909%	11.8108%	14.1272%

4. Gaussian Naïve Bayes

4.1 Algorithm

For Gaussian Naïve Bayes classifier, we use continuous data from z-normalization and log transform.
Pseudocode:

Algorithm2: Gaussian Naïve Bayes

Input: Spam Data
Output: error_rate

1. Load data
2. Data processing
3. Calculate posterior of y (π_c) with train data (in this project, we use ML)
4. Calculate posterior predictive distribution of x (Gaussian Distribution) with train data.
5. Test with test set.
6. Get error rate by using test label.

4.2 Results

After training, the results are shown below:

	z-normalization train	z-normalization test	Log transform train	Log transform test
Error rate	17.68%	20.25%	16.12%	18.42%

5. Logistic Regression

5.1 Algorithm

Algorithm3: Logistic Regression

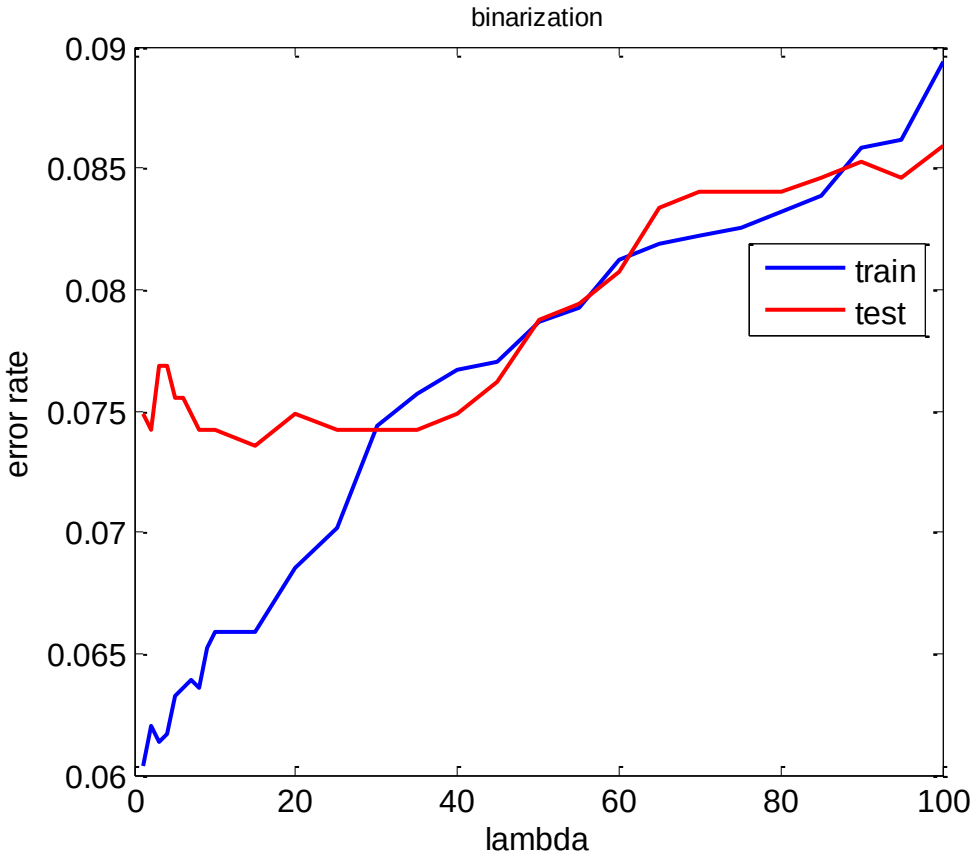
Input: Spam Data
Output: error_rate

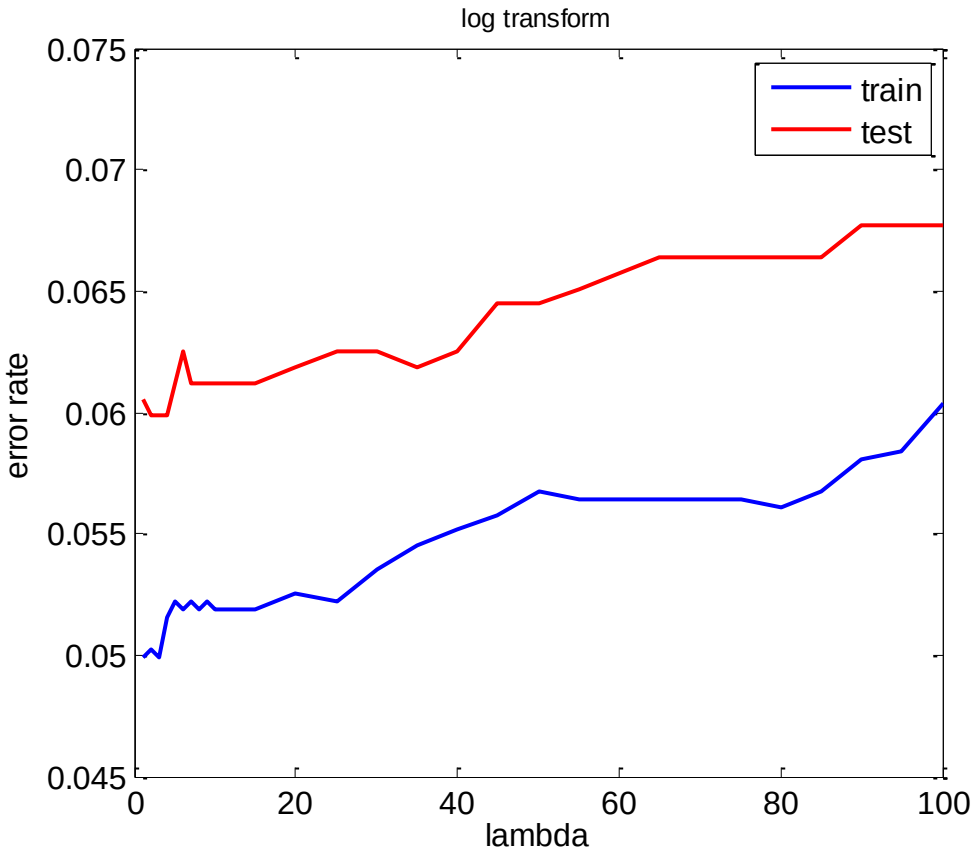
1. Load data
2. Data processing
3. Initialize weights w
4. Compute first order (g) and second order derivatives (H)

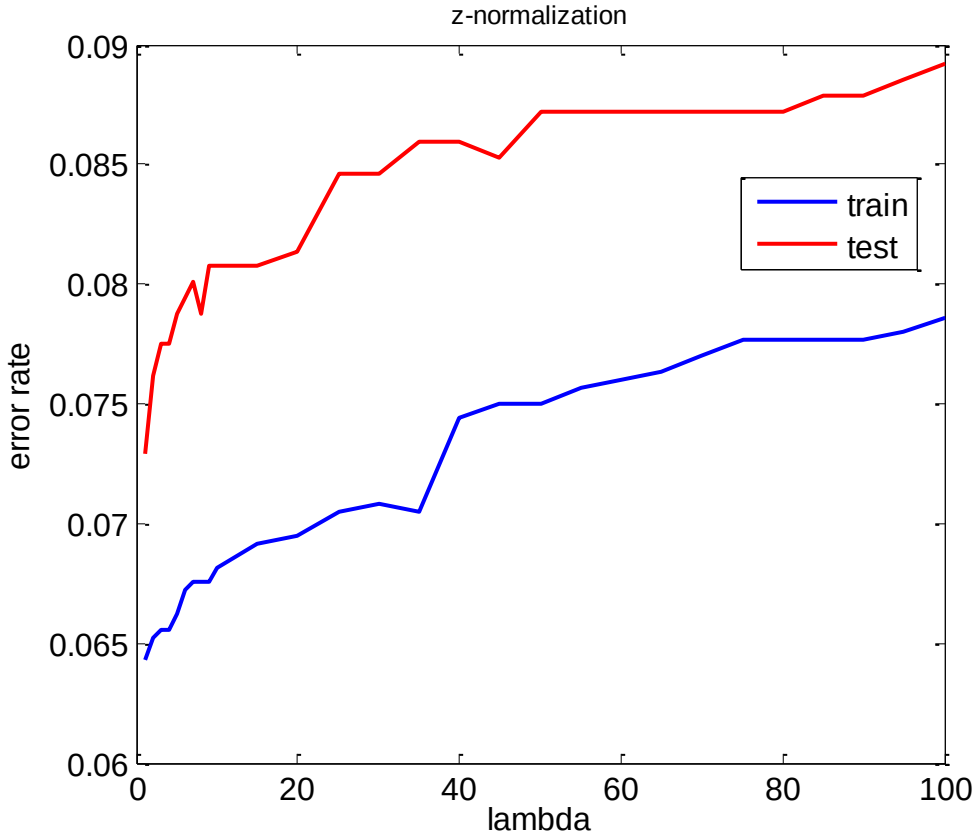
   $g = X^T(\mu-y)$, $H=X^TSX$ ($\mu_i = sigm(w^Tx_i)$, $S = diag(\mu_i(1-\mu_i))$)
5. Update the weights $w = w+d$ ($d = -H^{-1}g$) until the model converge
6. Test with test set
7. Get error rate by using test label.

5.2 Results

After training, the results are shown below:


binarization
0.09
0.085
0.08
0.075
0.07
0.065
0.06
error rate
0
20
40
60
80
100
lambda
train
test


log transform
0.075
0.07
0.065
0.06
0.055
0.05
0.045
error rate
0
20
40
60
80
100
lambda
train
test



5.3 Remarks

- Remark 1:

The training error increases with the λ increasing. For z-normalization and log transform the error rate of test are higher than that of train.

- Remark 2:

Although the difference between different preprocessing are slight, there do have the one performs better. After comparing the results shown above, I conclude that for same data, the result got from log transforming has the lowest training error which means that the log transform performs better than the other two means of preprocessing.

- Remark 3:

λ		1	10	100
binarization	Train	6.0359%	6.5905%	8.9396%
	Test	7.4870%	7.4219%	8.5937%
log transform	Train	4.9918%	5.1876%	6.0359%
	Test	6.0547%	6.1198%	6.7708%
z-normalization	Train	6.4274%	6.8189%	7.8630%
	Test	7.2917%	8.0729%	8.9193%

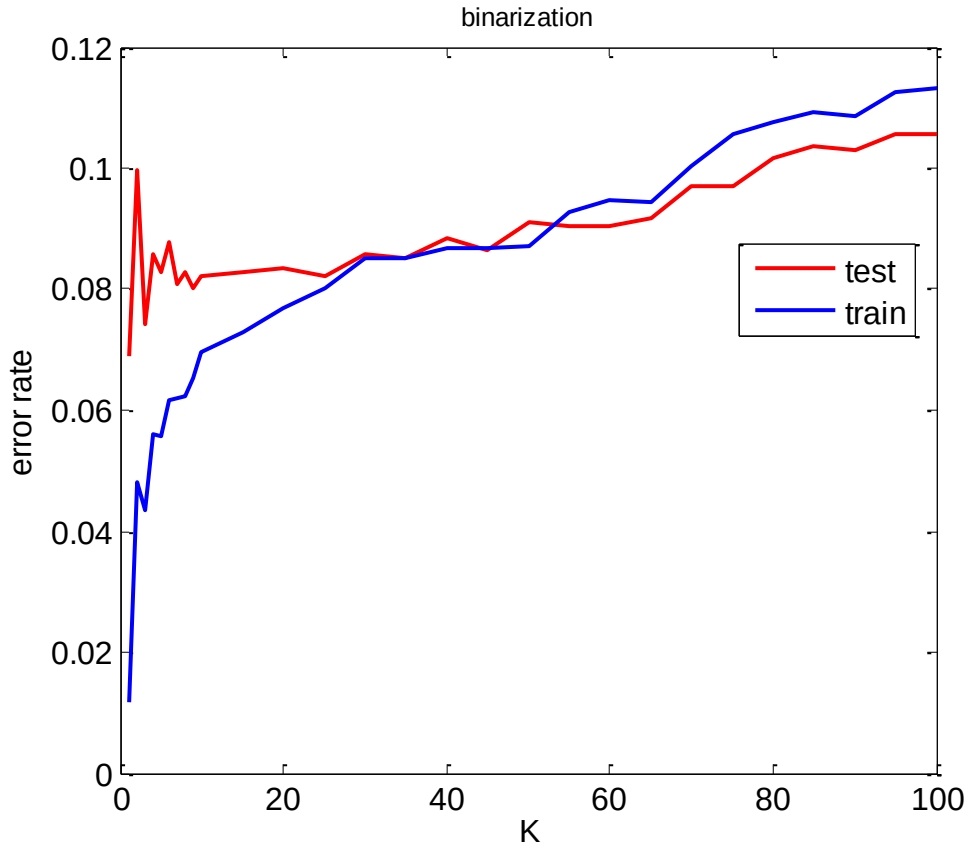
6. K-Nearest Neighbors

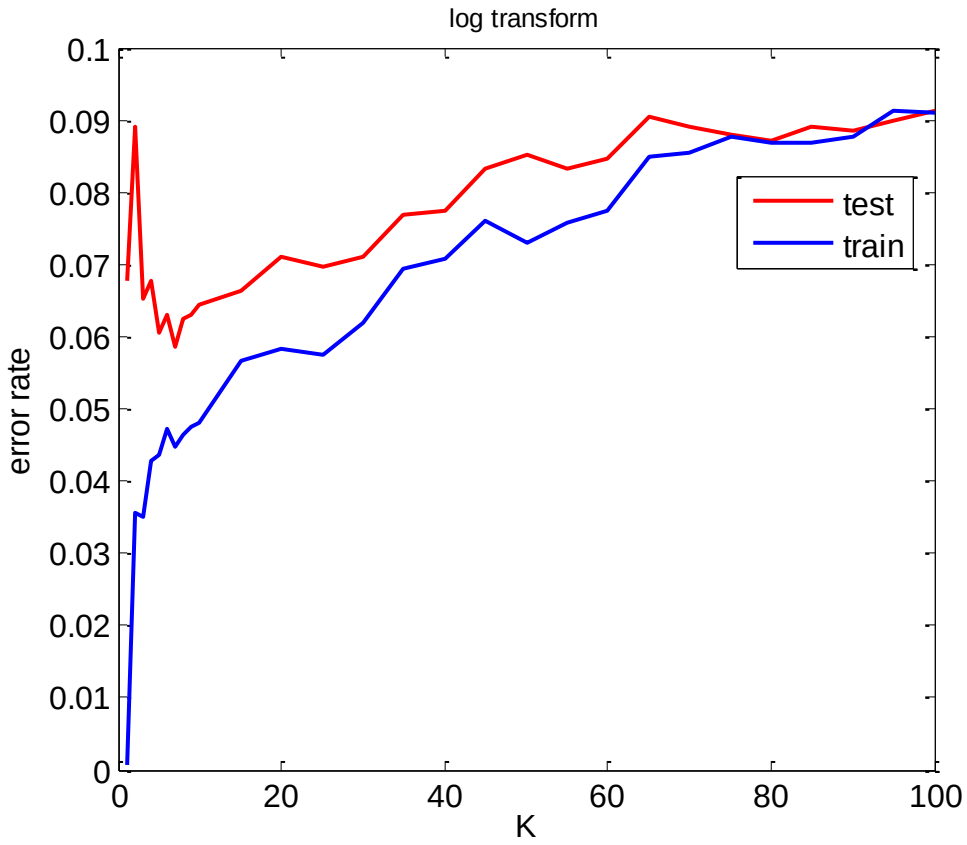
6.1 Algorithm

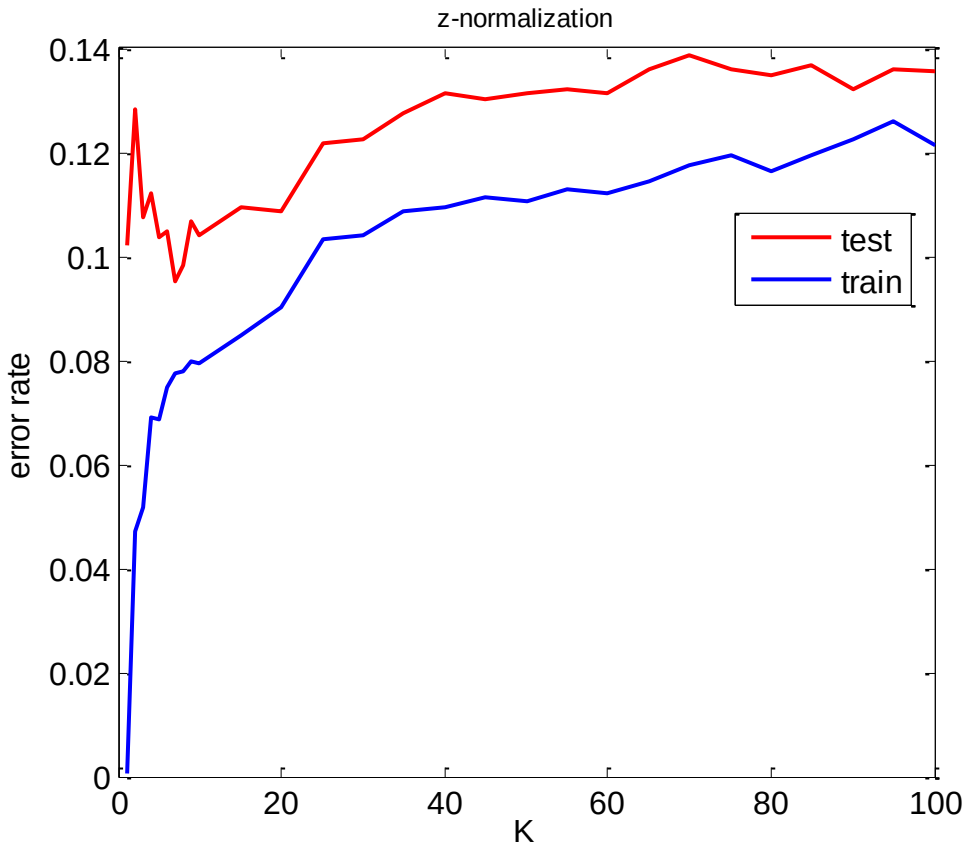
Algorithm4: K-Nearest Neighbors
Input: Spam Data Output: error_rate 1. Load data 2. Data processing 3. Calculate Euclidean distance for continuous data (z-normalization, log-transform), calculate Hamming distance for binarized data. 4. Rank the distances into ascending order. 5. Choose K smallest distance and compute P(y=c|x) 6. Test with test label 7. Get error rate

6.2 Results

For each kind of data processing, the results are shown below:




log transform
error rate
K
test
train
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
0
20
40
60
80
100


z-normalization
error rate
K
test
train
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0
20
40
60
80
100

6.3 Remarks

- Remark 1:

As K increase, the training error rate increases. The training error is not 0 when K = 1, because in my code, there is a judging condition shown below:

Judging Condition:

```
if(((y1count>=y0count)&&(ytest(i)==1))||((y1count<y0count)&&(ytest(i)==0)))
  correct=correct+1;
end
```

y1count and y0count is counting the k nearest points within the range. There is a larger or equal symbol. That means that when two points are of the same distance to that point, namely A and B. A is supposed to be the right one with label 1, and B labelled by 0. When we are sorting the point in ascending order, B is sorted before A. This condition is y1count>y0count and ytest(i)=0, the correct would not add with 1 consequently. The error rate is not 0 now.

- Remark 2:

Although the difference between different preprocessing are slight, there do have the one performs better. After comparing the results shown above, I conclude that for same data, the result got from log transforming has the lowest training error which means that the log transform performs better than the other two means of preprocessing.

- Remark 3:

K		1	10	100
binarization	train	1.1746%	6.9494%	11.3214%
	test	6.9010%	8.2031%	10.5469%
log transform	train	0.0653%	4.7961%	9.1028%
	test	6.7708%	6.4453%	9.1146%
z-normalization	train	0.0653%	7.9282%	12.1370%
	test	10.2214%	10.4167%	13.5417%

7. Survey

I spent about 3 days to do this assignment. I finished writing every code quickly, but it is time-consuming to debug. For example, I spent about 2/3 of my time to debug my KNN model. I even thought to abandon coding with Matlab, and switch to python. Magically, the result shows up. That reminds me of an experience when I was doing my FYP. After a long period of working, I finally got a perfect array contains only the number 5! I really think the fascination of studying deeply about a question is the time you get your expected result. Thank you for the chance.