HANDWRITTEN ZIPCODE ANALYSIS

AUGUST 27, 2021

KO TSZ NGA (VALERIE)
valerie.ktn@gmail.com

Introduction

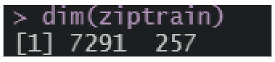
This project aims at comparing machine learning algorithms (ie. linear regression and KNN) on their performance in predicting the handwritten digits which were scanned from envelopes by the U.S. Postal Service.

Background of the dataset

According to https://web.stanford.edu/~hastie/ElemStatLearn//datasets/zip.info.txt , the first column of the dataset is our dependent variable - digit ID(ranging from 0-9) and the remaining 256 columns are the grayscale value, which are our independent variables.

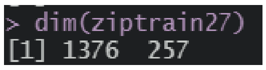
Data Preparation

The original 'ziptrain' dataset consist of 7291 rows and 257 columns



```
> dim(ziptrain)
[1] 7291  257
```

After narrowing the scope to only consider digit ID having the value either 2 or 7, we subset the 'ziptrain' to form 'ziptrain27' which only contains 1376 rows with the number of columns unchanged

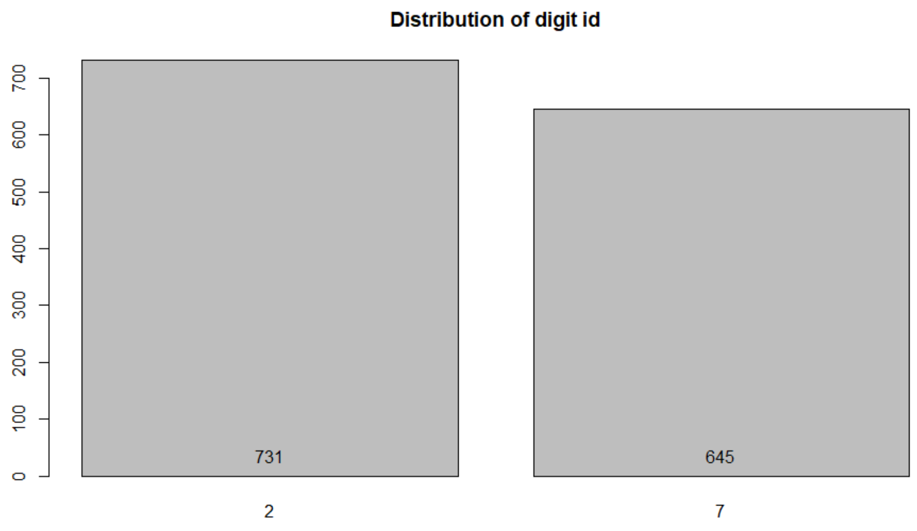


```
> dim(ziptrain27)
[1] 1376  257
```

Exploratory data analysis

1. digit ID distribution

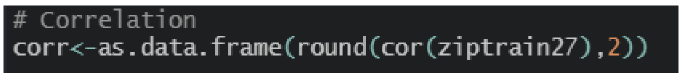
To further investigate on the digit ID distribution on 'ziptrain27', the following bar chart is created.



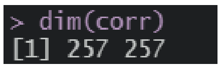
As we can see, digit id = 2 is slightly more than that of 7.

2. Correlation

I am also interested to know the correlation between the variables. To find that out, I used the 'corr' function and produced the below data frame.



```
# Correlation
corr<-as.data.frame(round(cor(ziptrain27),2))
```

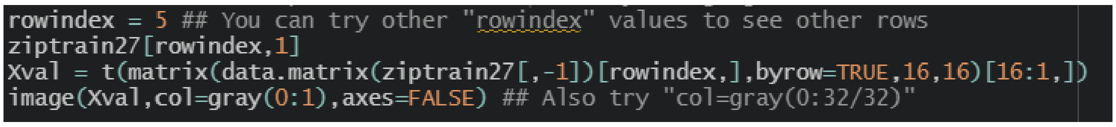


```
> dim(corr)
[1] 257 257
```

As we can see, there are 257 columns and rows respectively. Given that there are too many data correlation to be taken into account, it is not really meaningful to further visualise it and draw inference.

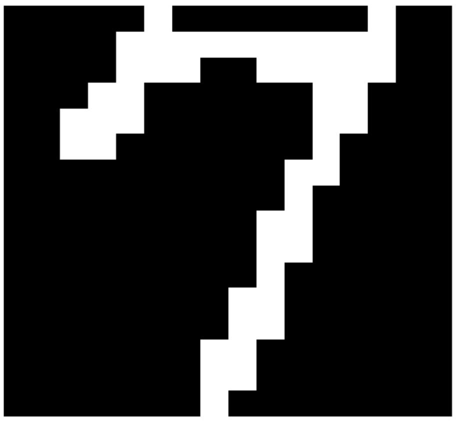
3. Data visualisation

There are 256 variables for grayscale value. That is hard to understand what implication they carry. Therefore, a few data point is selected for visualisation with the below code syntax:

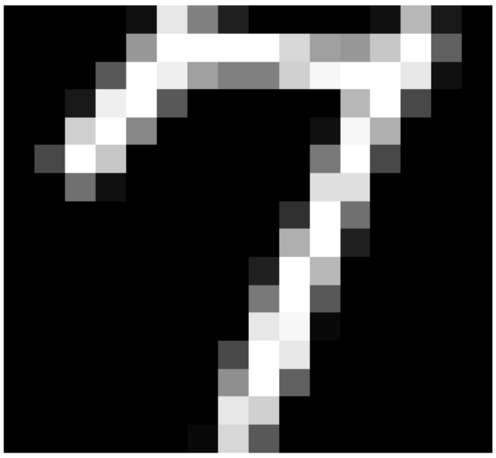


```
rowindex = 5 ## You can try other "rowindex" values to see other rows
ziptrain27[rowindex,1]
Xval = t(matrix(data.matrix(ziptrain27[,-1])[rowindex,],byrow=TRUE,16,16)[16:1,])
image(Xval,col=gray(0:1),axes=FALSE) ## Also try "col=gray(0:32/32)"
```

That produce the below image:

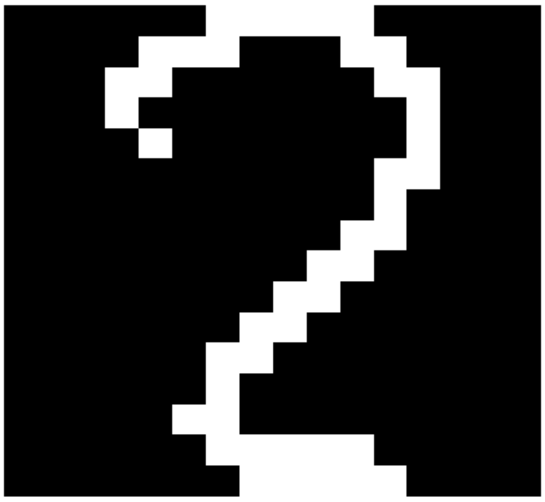


Changing the col parameter in 'image' function produces the below image:



Both images depict a letter similar to 7, which is the value on the digit_id column on row 5. However, the image with 'col=gray(0:32/32)' produce a more blurred image and hence the parameter of 'col=gray(0:1)' is more preferred.

Another example is drawn from row15 of the 'ziptrain27' data. This time, a letter similar to 2 is depicted.



In general, the variation of the 256 grayscale values determines the image and hence help to predict the digit_id of an instance.

Methodology

2 models are built, namely 3.1) linear regression and 3.2) KNN.

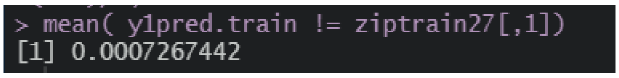
Result

a. Linear regression model

The training data is both used in model training and making prediction. ‘pred1.train’ store the vector of prediction made by the model. However, they are not represented by either 2 or 7. Transformation has to be done with below equation:

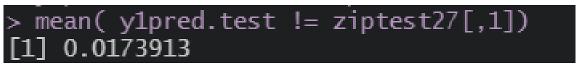
2 + 5*(pred1.train >= 4.5) , which produce a new vector ‘y1pred.train’ which only includes value of 2 or 7.

The expected training error is



```
> mean( y1pred.train != ziptrain27[,1])
[1] 0.0007267442
```

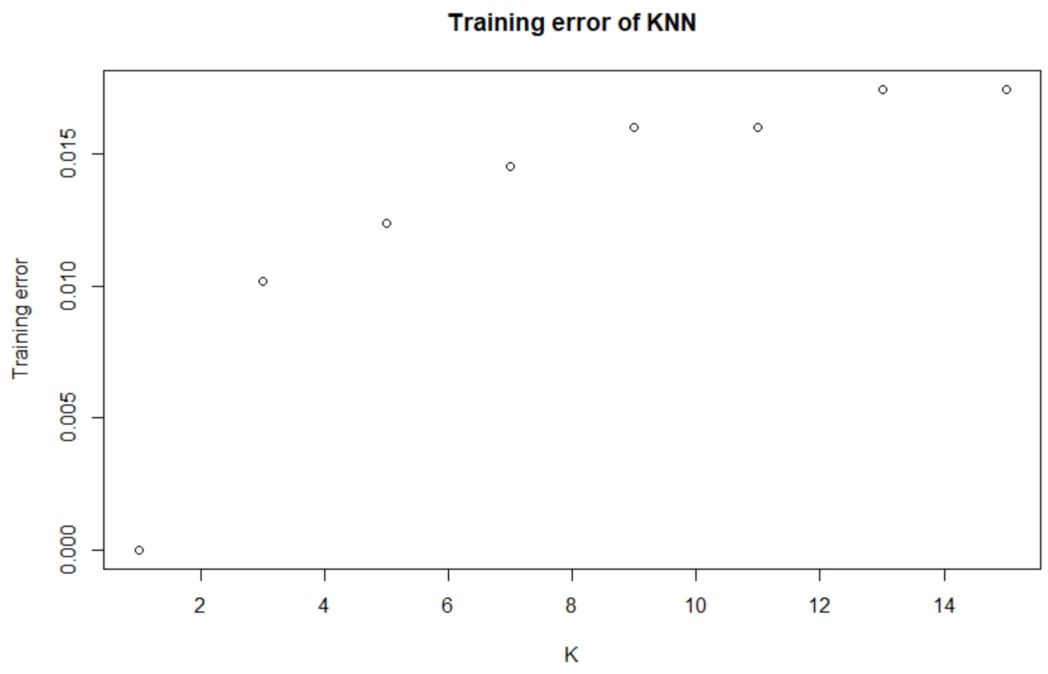
I re-run the linear regression model with testing data to obtain the below testing error.



```
> mean( y1pred.test != ziptest27[,1])
[1] 0.0173913
```

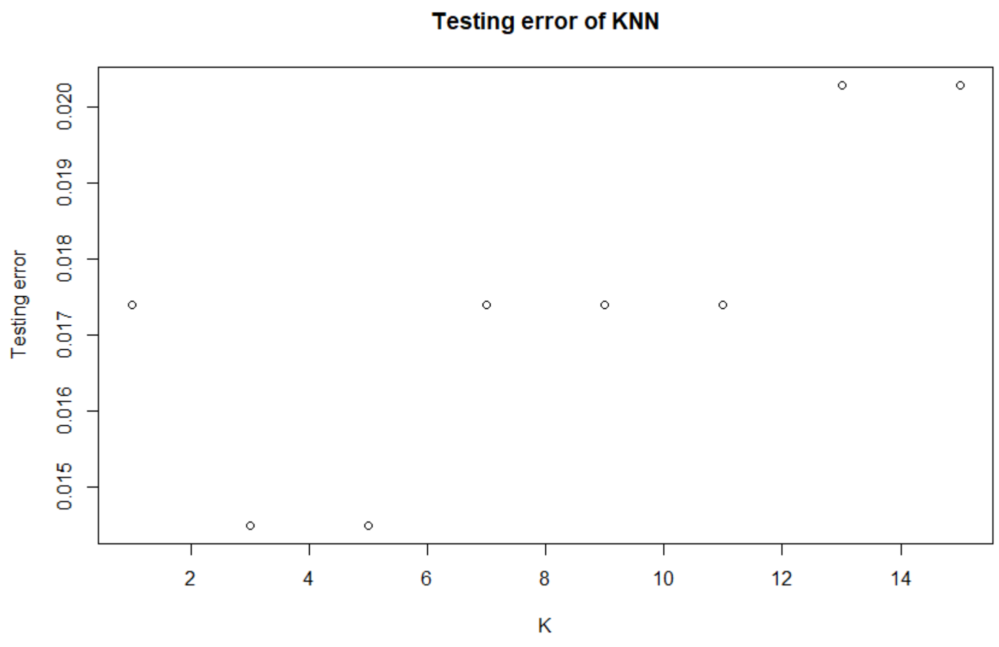
b. KNN

KNN with k=1, 3, 5, 7, 9, 11, 13, and 15 is used and the below graph displays the training error of the respective models.



We can see that the training error is the lowest with the smallest K (ie. K=1). The training error increase with K.

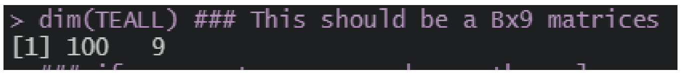
I re-run the KNN with k=1, 3, 5, 7, 9, 11, 13, and 15 and calculate the testing error of the respective models.



This time, unlikely the training error, the testing error is the smallest when k=3 and 5. That makes sense because the model with K=1 is biased for the training data, in other words, it is not generalised enough to predict the new data in testing set. In short, K=3 or K=5 are the better option.

Monte Carlo Cross-Validation

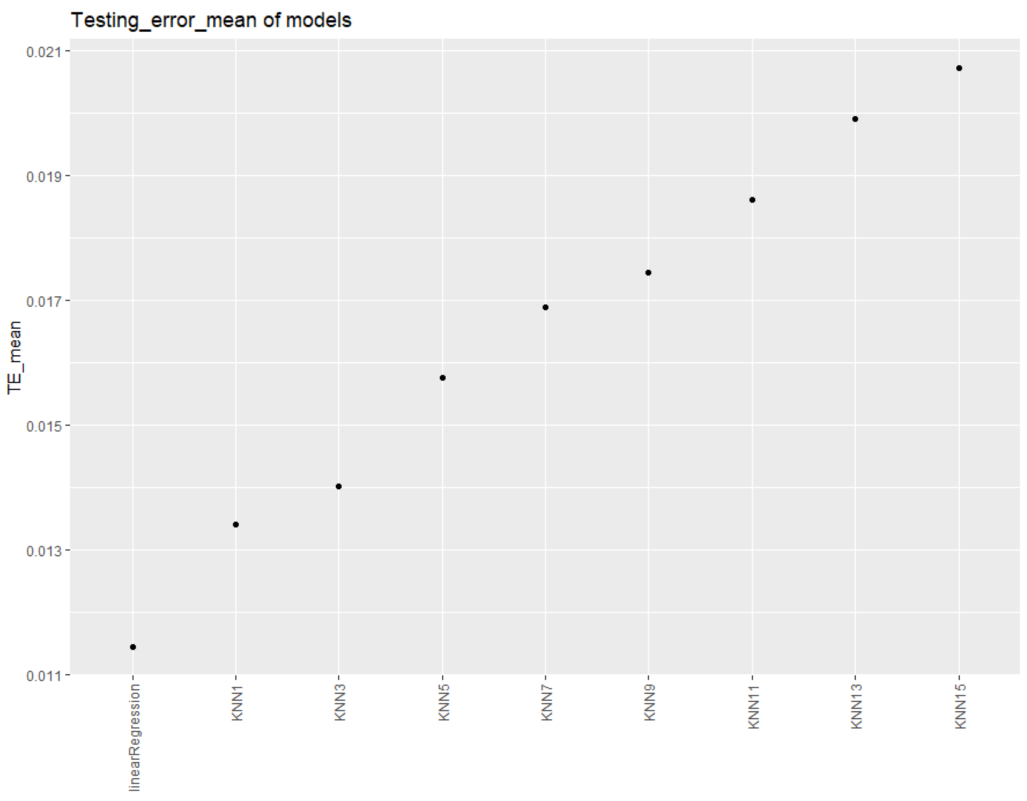
To validate the result, Monte Carlo cross validation on the full dataset is carried out. 100 loops will be run. 9 models include 1 linear regression model and 8 addition KNN model from k=1,3,5,7,9,11,13,15 will be built. Hence, the dimension of the output matrix includes 100 rows and 9 columns.

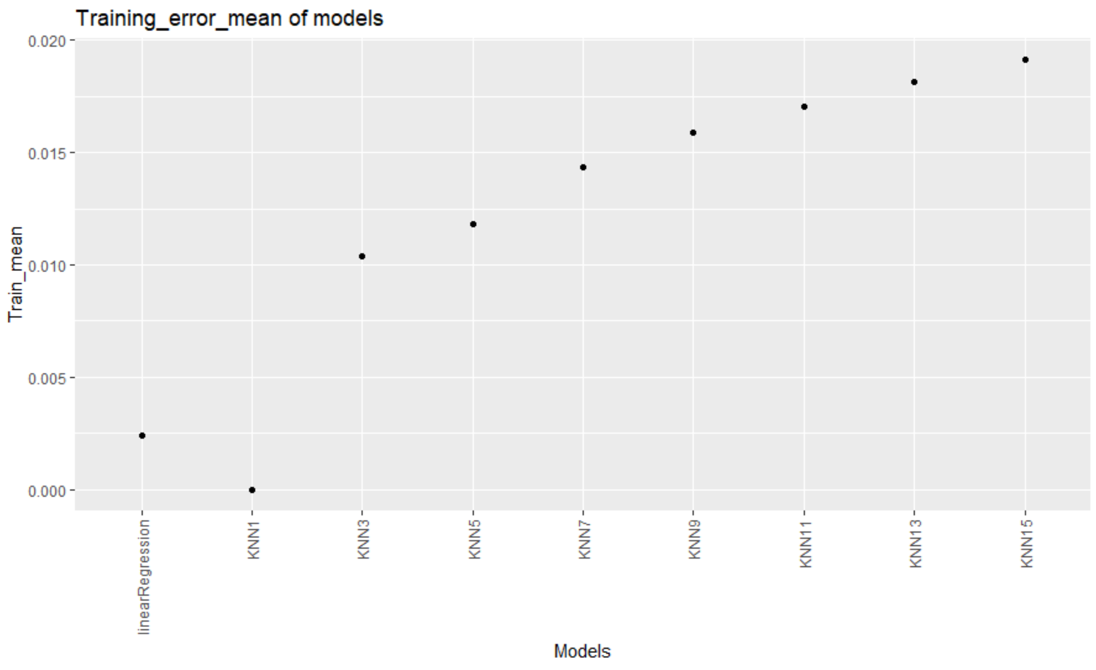


```
> dim(TEALL) ### This should be a Bx9 matrices
[1] 100   9
```

The sample mean, sample variance of the testing errors and training errors are calculated and visualised.

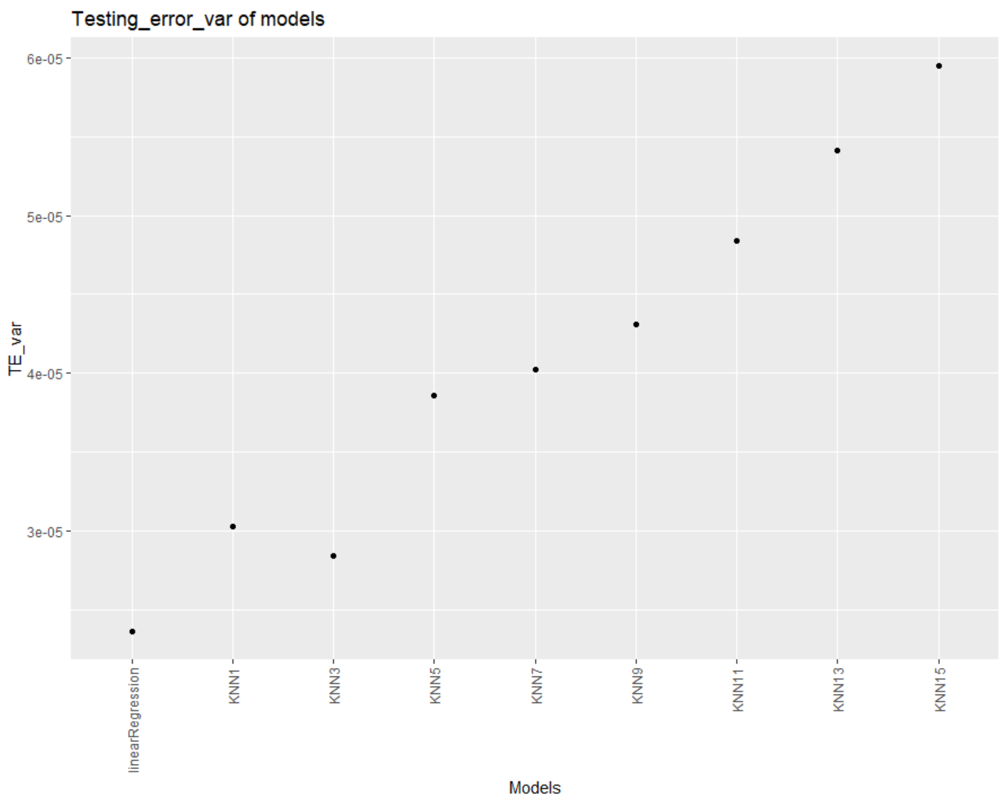
1. **Comparison for sample mean**

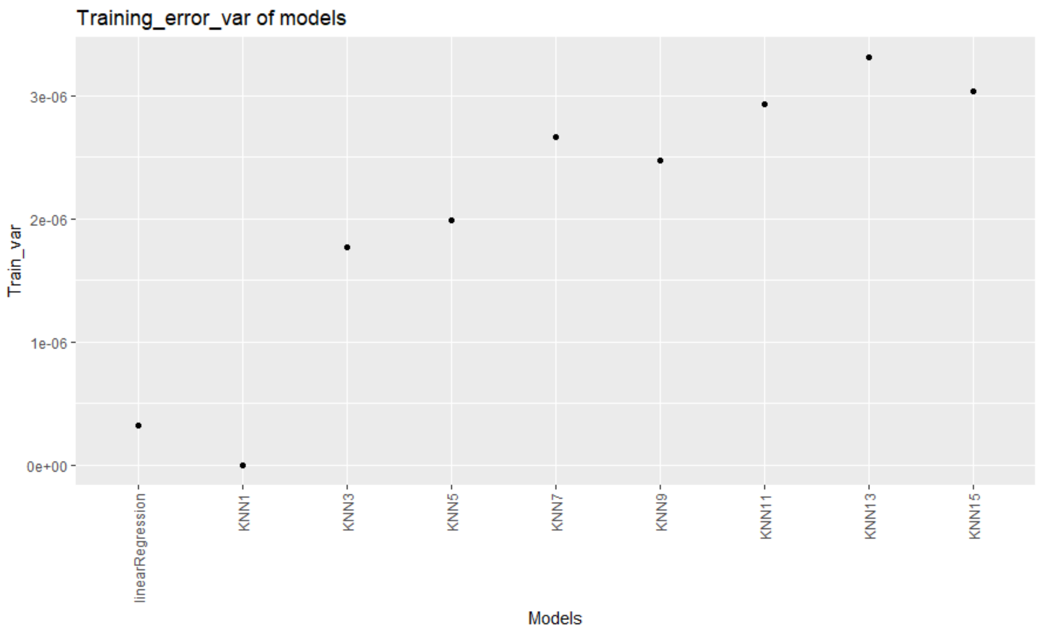




As shown in the above graphs, the testing error mean is the lowest in linear regression while the training error mean is the lowest for KNN model with K=1 (which gives a value of 0). The reason can be because there is bias for KNN model with K=1 with only training data is being used. Its predictability decreases when using testing data instead.

2. **Comparison for sample variance**





As shown in the above graphs, the testing error variance is the lowest in linear regression while that of the training error is the lowest for KNN model with K=1 (which gives a value of 0). The reason is again due to the bias involved in KNN model with K=1 with only training data is being used.

Conclusion

The model tested with testing data should be used as there are less bias and hence more generalised for prediction. It is obvious to see that **linear regression perform better**, as it gives the lowest testing error mean and variance when compared with all the KNN models. For KNN, **the optimal value for K should be 3**, it because it gives the lowest testing error variance among the KNN model.