Due Date: 5/26/2021

Introduction

Computer vision is a field of computer science that involves training computers to understand digital media such as images or videos. This subject of interest dates back several decades in time, but in today's rapidly evolving and technologically advanced environment, there are a number of different methodologies designed for this task. The following report depicts one such example, using the well-known MINST dataset, to recognize handwritten digits.

Data

The MINST dataset consists of a large collection of 28 x 28 pixel images, in which each image reflects a single handwritten digit between 0-9. Each individual pixel is represented by a positive integer within the range of 0-255, corresponding with the lightness-darkness of that pixel. As such, there are a total of 784 pixels for each image. The goal is to code a machine learning algorithm that is able to effectively determine the digit for a given image based on the image's pixels.

Methodology

The data is initially explored and some data preprocessing is performed. Noisy images, i.e. images with many outlier pixels, are removed prior to model training. several classification models are coded and trialed. These models include C50 decision tree, naive bayes, k nearest neighbors, support vector machine, and random forest. Based on the results, one of the models will be selected as the final production model and will be used to determine the unkown labels of the testing data.

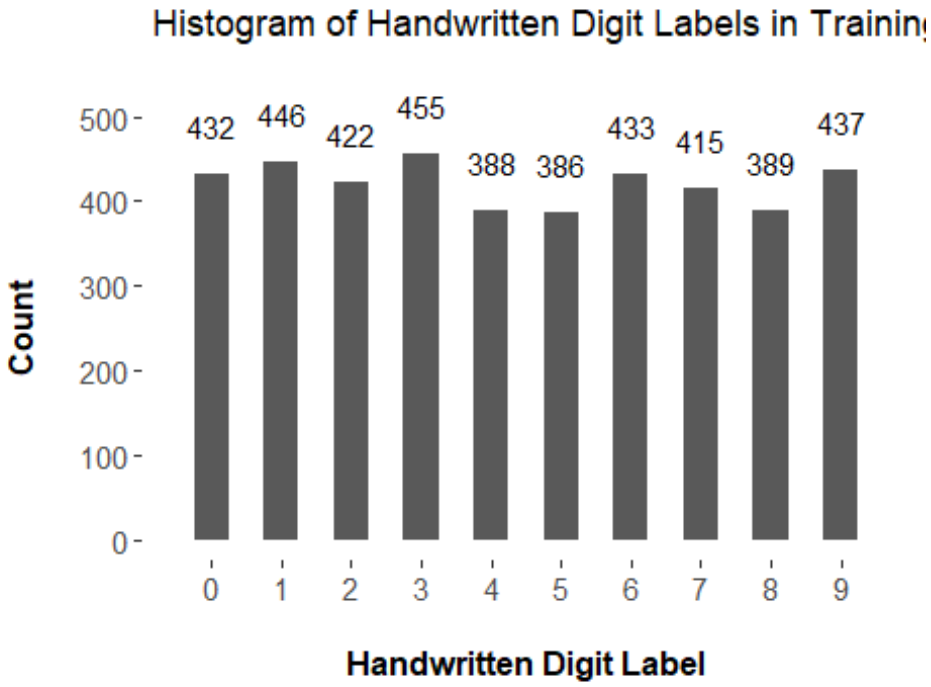
Exploratory Data Analysis

What are the dimensions of the data?

```
## Original Training and Testing Data
##
## Training Data Dimensions
## Rows = 4203 Cols = 785
## Size = 25 Mb
##
## Testing Data Dimensions
## Rows = 28000 Cols = 784
## Size = 168 Mb
```

How many of each digit are there in the training data?

```
##  Digit Count
##      0   432
##      1   446
##      2   422
##      3   455
##      4   388
##      5   386
##      6   433
##      7   415
##      8   389
##      9   437
##  Total  4203
```



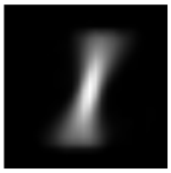
Observations

-there is a fairly uniform distribution of each digit

-the digit 1 has the most examples and the digit 5 the least

-no action needed, but keep in mind for subsequent analysis

What do some of the handwritten digits look like?


Observations

-given the blurriness of the images, there are likely some noisy images

-the blurrier parts have a greater variation in their pixels

-the images would be clearer if all of the data were closer together

Data Preprocessing

Are there any outliers?

```
# a pixel is an outlier if it falls outside:
# Q1 - (1.5 * IQR) or Q3 + (1.5 * IQR)
```

label	avgoutliers	labelcount	relativenoise
0	24	432	0.85
1	19	446	0.65
2	31	422	1.14
3	30	455	1.02
4	32	388	1.28
5	30	386	1.20
6	26	433	0.92
7	29	415	1.07
8	25	389	0.99
9	25	437	0.88

Observations

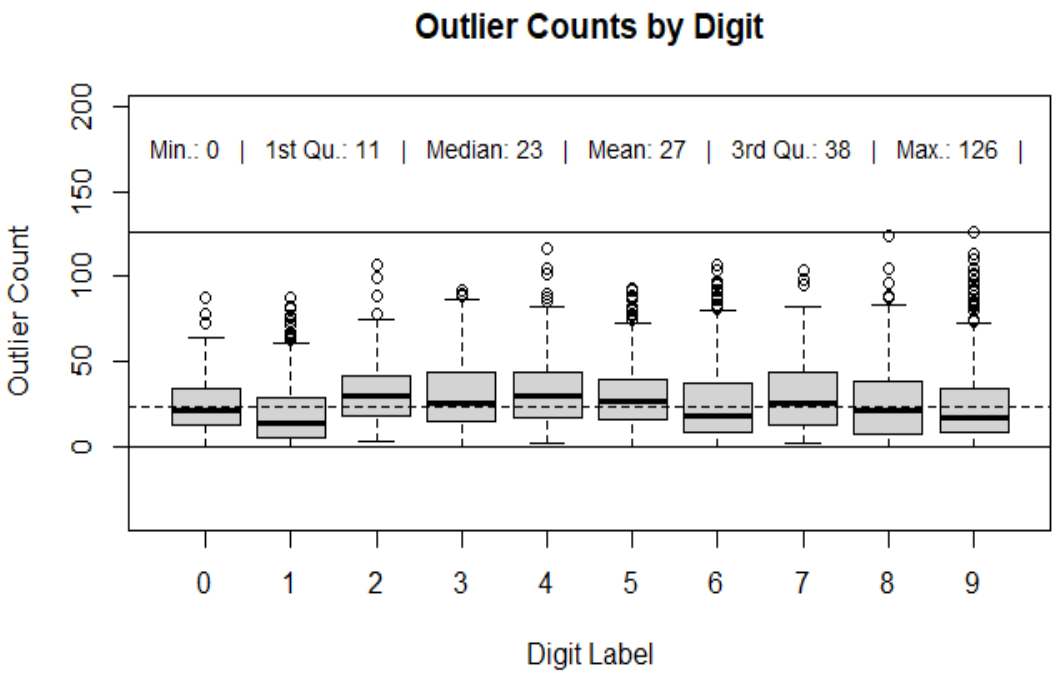
-this discovery uncovers some hidden noise in the dataset

-the noise level varies quite a bit depending on the digit

-1's have the most training examples, and the lowest relative noise

-5's have the least training examples, and the second highest noise

-the digits with lower relative noise will be more accurately predicted



Observations

-virtually every image has some outlier pixels. Very few perfect images

-this noise can be mitigated by ommitting the noisiest images

-what is considered a noisy image varies depending on the digit

-all outlier images will be removed from the training dataset

-potentially could cut more noise but do not want to lose too much data

Remove Outlier Images

What are the dimensions of the data after removing the outliers?

```
## Training Data Dimensions
## Rows = 4106 Cols = 785
## Size = 25 Mb
##
## Outlier Data Dimensions
## Rows = 97 Cols = 785
## Size = 1 Mb
##
## Testing Data Dimensions
## Rows = 28000 Cols = 784
## Size = 168 Mb
```

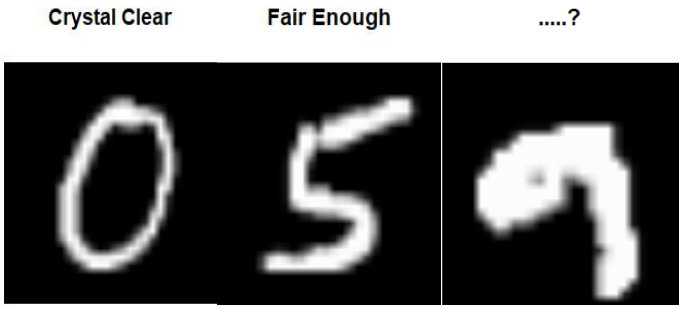
Observations

-only the most extreme cases of noise were taken out of the data

-the dimensionality of the data is still too high for efficient model computation

-principal component analysis will be performed to reduce the dimensionality

What do the noisy images look like versus the not noisy images?



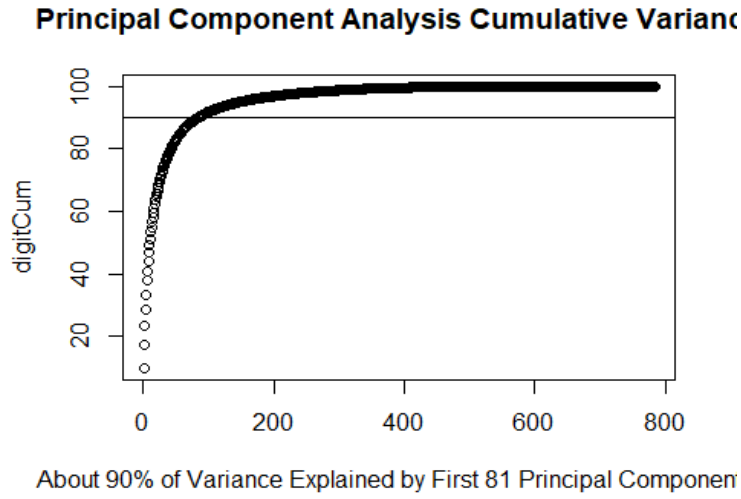
Observations

-the first image is very clear what digit it is

-the second image is fairly clear what digit it is

-the third image is not clear what image it is

Perform Principal Component Analysis



What are the dimensions of the data after performing PCA?

```
## Compressed 784 Columns (28 x 28) Into 81 Columns (9 x 9)
##
## Training Data Dimensions
## Rows = 4106 Cols = 82
## Size = 3 Mb
##
## Outlier Data Dimensions
## Rows = 97 Cols = 82
## Size = 0 Mb
##
## Testing Data Dimensions
## Rows = 28000 Cols = 82
## Size = 18 Mb
```

Size of the training data was reduced by:

```
## [1] "Size Reduction = -89.85%"
```

Size of the testing data was reduced by:

```
## [1] "Size Reduction = -89.56%"
```

Data Modeling

The first model will be a C50 decision tree model with a five fold cross validation. Model is too complex to visualize but analysis of results is provided

```
## Average =  73.86%

## Cross Validation 1 = 70.89%
## Cross Validation 2 = 76.98%
## Cross Validation 3 = 75.64%
## Cross Validation 4 = 73.08%
## Cross Validation 5 = 72.72%

##    TempPredict
##      0  1  2  3  4  5  6  7  8  9
##   0 68  0  2  2  0  2  3  0  1  0
##   1  0 71  3  0  0  0  0  0  2  0
##   2  3  1 67  4  1  1  6  5  1  1
##   3  1  1  3 70  1 13  1  2  5  2
##   4  0  1  0  0 60  1  1  0  1 16
##   5  9  0  2  7  3 43  5  1 11  2
##   6  6  0  6  0  2  0 74  1  2  1
##   7  0  0  0  3  4  3  0 63  3  4
##   8  3  1  6  9  1  4  3  2 45  6
##   9  0  2  1  4 11  3  0  4  2 36
```

Observations

-3s and 5s, 4s and 9s, 7s and 9s, got confused a lot

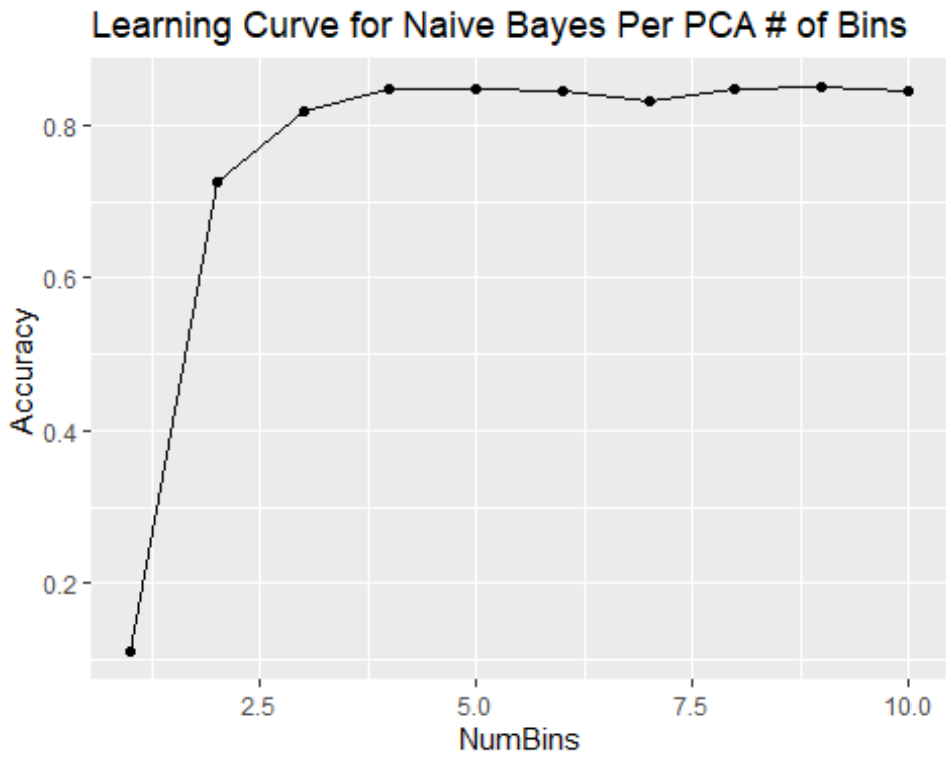
-the decision algorithm is sensitive to noise

-this explains why it did best on the 1 and worst on the 5

-it also was not relatively computationally efficient

The next model will be naive bayes. Discretized the PCA dimensions into bins 1 - 10 and compared model performance for each bin size.

```
## Model Complete | # of bins = 1 | Accuracy = 11.02 %
## Model Complete | # of bins = 2 | Accuracy = 72.71 %
## Model Complete | # of bins = 3 | Accuracy = 81.88 %
## Model Complete | # of bins = 4 | Accuracy = 84.7 %
## Model Complete | # of bins = 5 | Accuracy = 84.7 %
## Model Complete | # of bins = 6 | Accuracy = 84.46 %
## Model Complete | # of bins = 7 | Accuracy = 83.35 %
## Model Complete | # of bins = 8 | Accuracy = 84.82 %
## Model Complete | # of bins = 9 | Accuracy = 85.07 %
## Model Complete | # of bins = 10 | Accuracy = 84.58 %
```



```
##
##  Confusion Matrix for 10 Bin Model

##    TempPredict
##       0  1  2  3  4  5  6  7  8  9
##   0 75  0  1  1  0  4  1  2  1  0
##   1  0 84  0  0  0  1  0  0  0  0
##   2  3  0 66  5  3  1  1  1  2  1
##   3  0  0  2 76  0  6  2  1  2  1
##   4  0  0  1  0 61  1  3  3  1  6
##   5  1  0  0  4  2 59  3  1  1  3
##   6  1  0  3  0  1  3 73  0  2  0
##   7  1  0  0  0  4  3  1 71  1  1
##   8  1  0  3  5  0  4  0  0 63  0
##   9  1  0  0  3  5  3  0  6  2 63
```

Observations

-the naive bayes model was more accurate than the decision tree

-it was also computationally faster than the decision tree

-any number of bins between 5-10 should be fine. No benefit after that.

The next model will be a k nearest neighbors model. K values 1 - 10 will be used and the models will be compared.

```
## Model Complete | K = 1 | Accuracy = 100% | RunTime = 0 Minutes
## Model Complete | K = 2 | Accuracy = 94.2% | RunTime = 0 Minutes
## Model Complete | K = 3 | Accuracy = 95.65% | RunTime = 0 Minutes
## Model Complete | K = 4 | Accuracy = 92.75% | RunTime = 0 Minutes
## Model Complete | K = 5 | Accuracy = 93.48% | RunTime = 0 Minutes
## Model Complete | K = 6 | Accuracy = 92.75% | RunTime = 0 Minutes
## Model Complete | K = 7 | Accuracy = 91.3% | RunTime = 0 Minutes
## Model Complete | K = 8 | Accuracy = 89.86% | RunTime = 0 Minutes
## Model Complete | K = 9 | Accuracy = 91.3% | RunTime = 0 Minutes
## Model Complete | K = 10 | Accuracy = 92.75% | RunTime = 0 Minutes
```

Observations

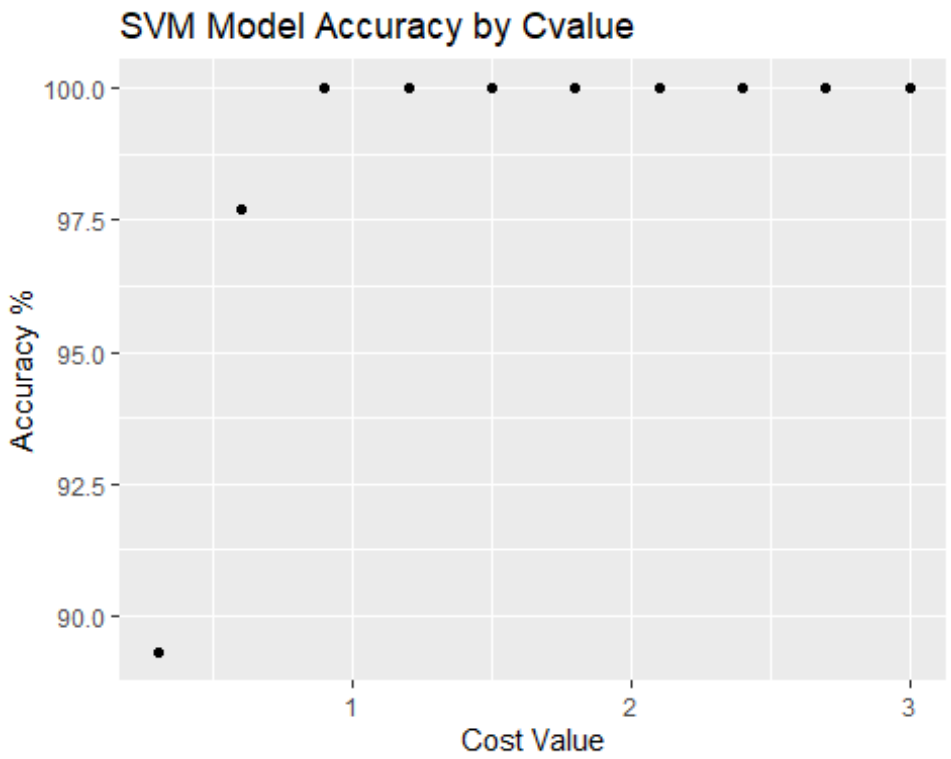
-accuracy is significantly higher for the knn model

-interesting that 100% accurate at k = 1

-there is likely some overtraining occurring

The next model will be support vector machine model. Several different cost values will be used and the models will be compared.

```
## Model Complete | C = 0.3 | Accuracy = 89.31% | RunTime = 0.01 Minutes
## Model Complete | C = 0.6 | Accuracy = 97.71% | RunTime = 0.01 Minutes
## Model Complete | C = 0.9 | Accuracy = 100% | RunTime = 0.01 Minutes
## Model Complete | C = 1.2 | Accuracy = 100% | RunTime = 0.01 Minutes
## Model Complete | C = 1.5 | Accuracy = 100% | RunTime = 0.01 Minutes
## Model Complete | C = 1.8 | Accuracy = 100% | RunTime = 0.01 Minutes
## Model Complete | C = 2.1 | Accuracy = 100% | RunTime = 0.01 Minutes
## Model Complete | C = 2.4 | Accuracy = 100% | RunTime = 0.01 Minutes
## Model Complete | C = 2.7 | Accuracy = 100% | RunTime = 0.01 Minutes
## Model Complete | C = 3 | Accuracy = 100% | RunTime = 0.01 Minutes
```



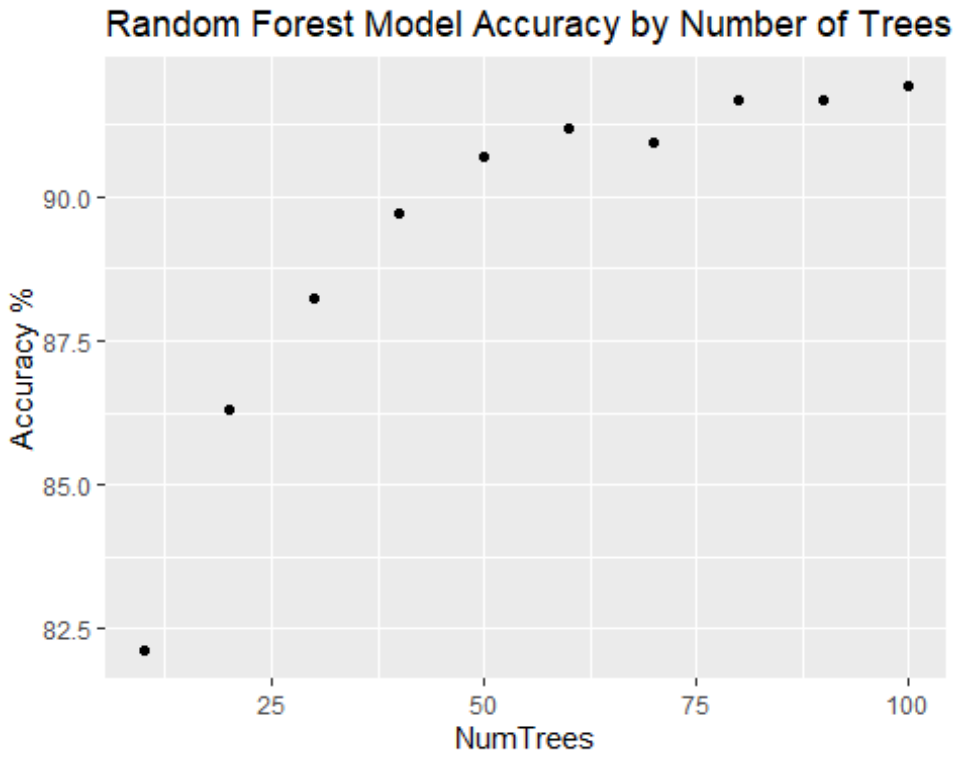
Observations

-the SVM model is very accurate

-once the cost is increased to 0.9, the accuracy becomes 100%

The next model will be random forest. Several different number of trees will be used and the models will be compared.

```
## Model Complete | NTrees = 10 | Accuracy = 82.13% | RunTime = 0.01 Minutes
## Model Complete | NTrees = 20 | Accuracy = 86.29% | RunTime = 0.01 Minutes
## Model Complete | NTrees = 30 | Accuracy = 88.25% | RunTime = 0.02 Minutes
## Model Complete | NTrees = 40 | Accuracy = 89.72% | RunTime = 0.02 Minutes
## Model Complete | NTrees = 50 | Accuracy = 90.7% | RunTime = 0.03 Minutes
## Model Complete | NTrees = 60 | Accuracy = 91.19% | RunTime = 0.03 Minutes
## Model Complete | NTrees = 70 | Accuracy = 90.94% | RunTime = 0.04 Minutes
## Model Complete | NTrees = 80 | Accuracy = 91.68% | RunTime = 0.04 Minutes
## Model Complete | NTrees = 90 | Accuracy = 91.68% | RunTime = 0.05 Minutes
## Model Complete | NTrees = 100 | Accuracy = 91.92% | RunTime = 0.05 Minutes
```



Observations

-the random forest models take longer to train for more trees

-the accuracy incrementally increases with number of trees

-the accuracy does start to level off after 50 trees

Conclusions

At first when I developed the models, I had not done anything about the outliers in the data. I eventually came to my senses that it would be helpful to do so, and I am glad that I did, because as it turns out this did increase the accuracy by several percentage points on each model.

The decision tree and naive bayes models did not perform well enough for this task. I am not confident that either of these models are the best model for the task at hand.

The accuracy for the remaining models were similar, with the best performance coming from support vector machine. The knn model did reach 100% accuracy with a kvalue of 1, but I am not sure if this makes sense or not. Therefore, SVM model was the best.

I do plan on submitting this project to kaggle, which will be my first ever kaggle submission. There are a few things that I hope to modify before doing so that I did not have time to do before this homework assignment was due.

I would like to see if there is a model that can predict the outliers seperately. I will need to do some further data exploration to justify if this would make sense or not. I will also clean up my code and rmd output a little bit more

Overall, I had a lot of fun with this assignment and learned alot. I believe that this assignment helped me grow as a data scientist. I started to organize my code better and use comments more effectively to explain what the code is doing. I am happy with the results that I have gotten in this assignment.