Team Marshmallow

STOR 565

May 3, 2022

Kyungjin Sohn, SooHyun Kim, Winnie Ren, Yang Xiao, Ziang Li

How to Recognize Fake News

Abstract

This research looks into how to classify fake news using text characteristics. The data set used in this research includes 44,898 observations (23,481fake and 21,417 true). After preprocessing and cleaning the data, we implement two feature extraction methods: bag of n-grams and sentiment analysis. Using the features extracted we build 10 models to classify fake and true news (LDA, QDA, KNN: k = 1, KNN: k = 184, SVC (linear), SVM (polynomial), decision trees, bagging, random forest, logistic, neural networks). From our models, we find that the random forest had the best performance with an accuracy of 0.931, sensitivity of 0.914, and a specificity of 0.947. Looking at the variable importance of the random forest model, we find that news articles including words that are similar to trust are an important indicator of fake news. Also, the average and the 1st quartile of text sentiment scores are important. Based on this finding, we suggest that when individuals are reading news articles, they should be on the lookout for words such as agree, associate, certify, reassure, and verified, since these words are an indication that the article is trustworthy. In addition, individuals should be more alert when reading news articles with a negative voice.

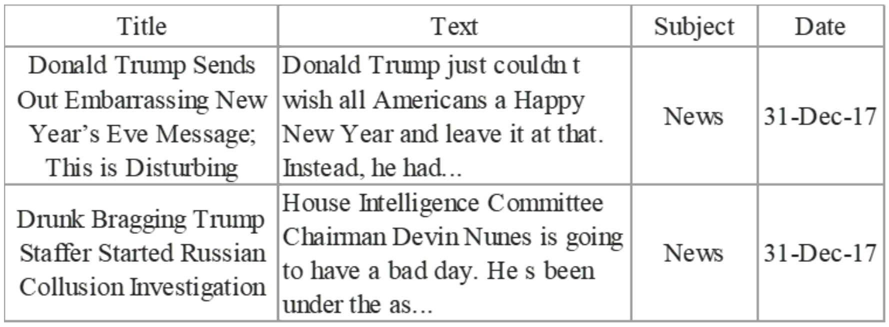
1. Introduction

At some point of our lives, we have all come across a news article that is fake. However, whether we realize that the article is fake or not, that is the question. Fake news, defined as news that is intentionally crafted to convey misleading information or totally fabricated information and presented in a way that mimics conventional news, propagates the spread of misinformation and disinformation. Especially, the Covid-19 pandemic was accompanied by the circulation of misinformation, myths, and conspiracy theories about the disease. In 2021, it is estimated that nearly 8 in 10 adults have experienced confusion regarding COVID-19 (Budgar, 2022). However, it is not just limited to confusion. It has been reported that fake news has led to death from alcohol poisoning and cleaning products (Spring, 2020).

The issue of fake news is not only limited to the COVID-19 pandemic, but is widely present in other time periods as well. One of the most common is during great political activities, like elections. To prevent such confusion and severe ramifications that may follow from fake news, this project aims to look at aspects of textual data to determine whether a news article is truthful or not. Therefore, our goal is to build a fake news detection model that will be able to recognize whether a particular article is real or fake by using machine learning techniques.

In the following section, we will first discuss the data used in this project detailing how the data was cleaned. Next, a simple EDA will explain how textual characteristics differ from true and fake news. In the following section, feature extraction methods, bag of n-gram and sentiment analysis, will be discussed. The classification models and their performances will be discussed. Finally, we will conclude with the interpretation of our findings and possible future work.

Table 1. Data Structure



Title	Text	Subject	Date
Donald Trump Sends Out Embarrassing New Year's Eve Message; This is Disturbing	Donald Trump just couldn t wish all Americans a Happy New Year and leave it at that. Instead, he had…	News	31-Dec-17
Drunk Bragging Trump Staffer Started Russian Collusion Investigation	House Intelligence Committee Chairman Devin Nunes is going to have a bad day. He s been under the as…	News	31-Dec-17

2. Data

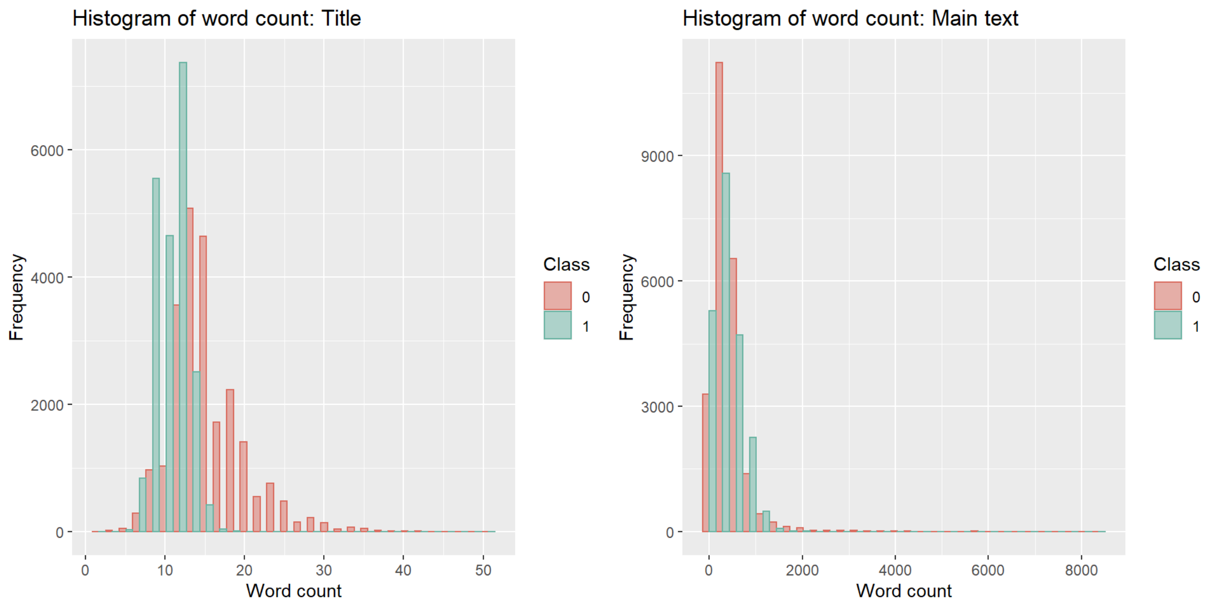
Our dataset on real versus fake news comes from Kaggle (Bisaillon, 2020). There, we found two files– one containing the data for real news articles and the other containing data for fake news articles. Each article in the dataset was run through a fact-checking website called PolitiFact, which determined the accuracy of the claims made in the articles using its Truth-O-Meter.

Our data contains 23,481 fake and 21,417 real news with four columns each: title, main text, category, and the date. Regarding the subject of the observation, the observations of the real news dataset had two subject categories: politics and world news. The fake news articles had five subject categories, which are the following: News, politics, left-news, Government News, and US News. Table 1 shows a sample structure of the data.

2.1 Data Preprocessing

When inspecting our dataset, we found that the majority of the observations of the real news dataset contained the following prefix in its main text: “Location (Reuters) -”. We suspected that this prefix would interfere with our fake news classification model, therefore, we made the decision to remove it from the main text.

Figure 1. Histogram of Title and Main Text Word Count for Fake and Real News

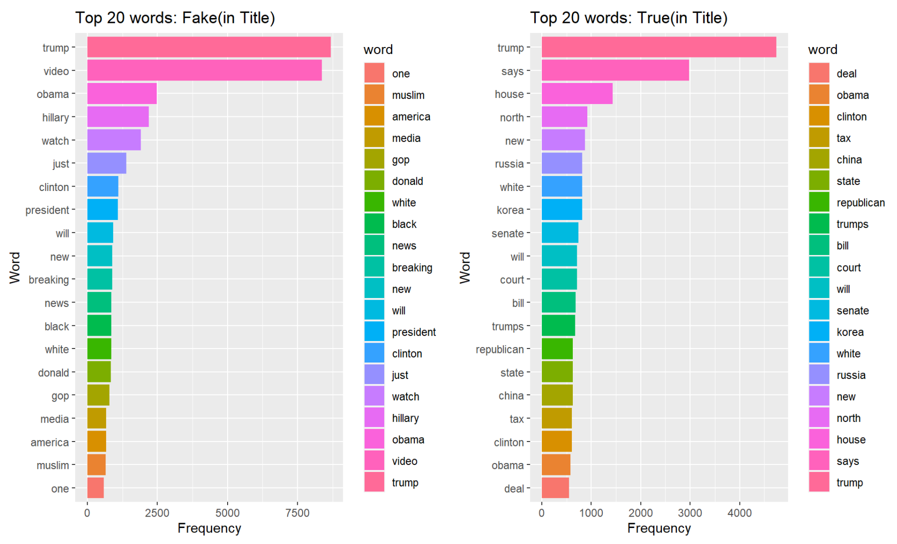


We found it necessary to use natural language processing techniques and perform feature extraction since the data that we are using did not come with features that we could work with. To create a corpus of our data, we removed numbers, punctuation, special characters, whitespace, stopwords and changed all text to lowercase. Finally, we stemmed each word. In the English language, a single word can take different forms. For instance, the word "change" could appear in the text in forms like "changing," "changed," and "changer." However, in our feature extraction all these forms should be counted as the same as they have the root. Therefore, we would stem all these variations in the form of "chang." The same process would be repeated over all words in the text.

3. Exploratory Data Analysis

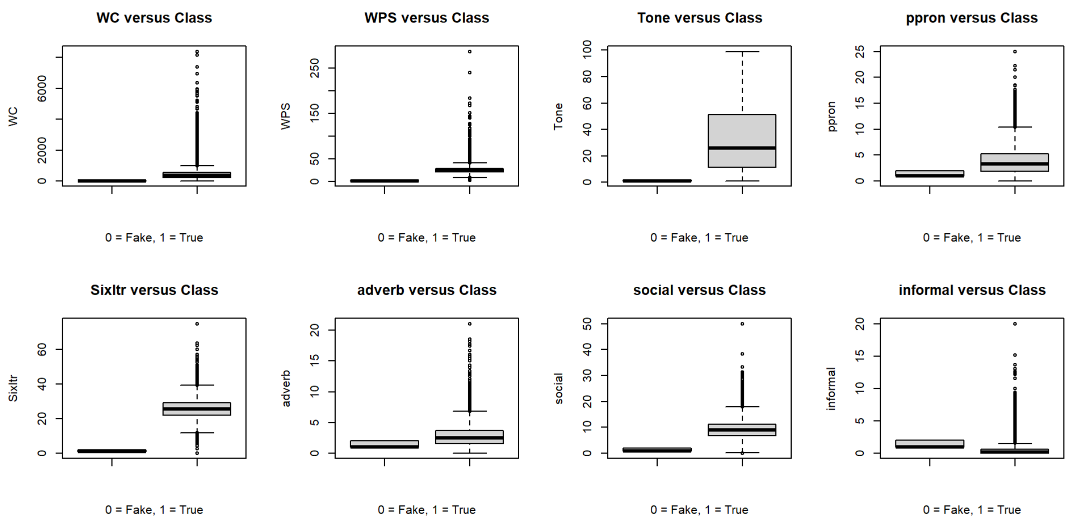
To look at the difference between fake and true news, we first look at the word count distribution of fake and true news. Looking at Figure 1, we find that the spread of fake news is wider compared to the true news. As for the word count distribution for the main text, we find that fake news and real news share similar distributions for word count. Next, we looked at the most frequently used words of fake and true news in both the title and the main text. Looking at Figure 2 we can compare the fake and the real news titles. We find that there is a difference in the words that are frequently used in fake and real news.

Figure 2. Histogram of Most Frequently Used Words in Title for Fake and Real News



Finally, we used a lexicon based method with the Linguistic Inquiry and Word Count (LIWC). Using the LIWC we were able to extract 95 variables. Out of these variables, we selected a few that had a significant difference between true and fake news. The boxplot in Figure 3 visually confirms the difference between textual variables.

Figure 3. Boxplot of Select LIWC Variables



4. Feature Extraction

When using text data, the data itself does not include variables with which models can be directly built. Therefore, we need to perform a step of feature extraction. We used two methods: bag of n-grams and sentiment analysis.

4.1 Bag of n-Gram

The reason why we use n-gram is that with language the order words appear is a crucial part that needs to be considered. For example, looking at two very simple sentences "I like you" and "I am like you", if a simple word count is used the only difference with these two sentences would be with one count of the word "as." However, these two sentences have a different meaning. Hence, the bag of n-grams, rather than the simple count of each word, is the count of n words together. Taking the second sentence as an example, when the n is set to 2 the terms would be "I am," "am like," and "like you." The count for each term would be one. We transform this into the Term Frequency (TF), which is calculated as TF = (term count in the doc)/(total number of terms in the doc). In the example above, the total number of terms in the document would be 3 and we would divide each frequency by that.

It is also important to normalize the bag of n-grams. This is due to the fact that some terms appear more frequently in documents, but may not have an increased added value in the classification model.

Therefore, we would want to control for more frequently used terms. The controlled term is called "Term Frequency-Inverse Document Frequency (TF-IDF)." The calculation for TF-ID is TF multiplied by Inverse Document Frequency (IDF). The IDF is the log of the number of documents divided by the number of documents with the term.

For our data to calculate the TF and the TF-IDF, we used the "bind_tf_idf" function in the "janeaustenr" library (Silge, 2017). We also set the n=2, based on trial and error and as it is widely accepted that bigrams would perform adequately without the need to increase the complexity by increasing the n above 2 (Jurafsky & Martin, 2021) (Wang & Manning, 2012). We selected the 20 most frequently used terms for fake and true news titles respectively and calculated the TF and TF-IDF. This returned a total of 40 TFs and 40 TF-IDFs. The same process is repeated for the text, but we discarded the overlapping terms which returned 33 TFs and 33 TF-IDFs.

4.2 Sentiment Analysis

We use the Syuzhet Package to construct features using sentiment analysis techniques (Jockers, 2020). We first use the get_sentences() function which implements the openNLP sentence tokenizer to parse a text into a vector of sentences. Then two different methods are applied to the generated vector. The first method is get_sentiment(). This method takes two arguments which are a character vector of sentences and a method. The default method is "Syuzhet". And more method options include "bing," "afinn," "nrc," and "stanford." The values are the model's assessment of the sentiment in each sentence. Different methods will return slightly different results. Part of the reason for this is that each method uses a slightly different scale. We chose to use the default method. The other method is called get_nrc_sentiment(). These methods implement Saif Mohammad's NRC Emotion lexicon. It also takes in a vector of sentences and returns a data frame with eight emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive).

A total of 23 features from applying the sentiment analysis package for title and main text are derived. Since the title is just one sentence, get_sentiment will only return one score. While for the main text, we will have Minimum, 1st Quartile, Median, Mean, 3rd Quartile, Maximum for the whole paragraph coming from each sentence's score. The reason we pass in raw text and title as the input for the get_sentiment is that this method returns a score for each sentence. As we used corpus stemmed data as the input for get_nrc_sentiment. We chose the scores for eight emotions for main and text from the data frame. This method gives us 16 features in total.

5. Classification Model

To build a model, we first split the data into two datasets. We used 75% of the total data for training and 25% for the test data. Then, we developed a total of 10 classification methods as listed below. Since some models have hyper-parameters to tune, we did a lot of trial and error like cross-validation to find the optimal model. The written values written with the model are the parameters that are finally chosen. For the neural network, we used 3 dense layers and 2 dropout layers with a learning rate of 0.01.

Table 2. Result of Classification Model

Method	Accuracy	Sensitivity	Specificity	False Positive	Test Error
LDA	0.864	0.856	0.871	0.129	0.136
QDA	0.8	0.874	0.734	0.266	0.2
KNN (K=1)	0.891	0.869	0.911	0.089	0.109
KNN (K=184)	0.868	0.905	0.835	0.165	0.132
SVC	0.858	0.82	0.892	0.108	0.142
SVM	0.732	0.76	0.707	0.293	0.268
Decision Tree	0.788	0.616	0.943	0.057	0.212
Bagging	0.923	0.902	0.941	0.059	0.077
Random Forest	**0.931**	**0.914**	**0.947**	**0.053**	**0.069**
Logistic	0.888	0.869	0.905	0.095	0.112
Neural Network	0.887	0.89	0.884	0.116	0.113

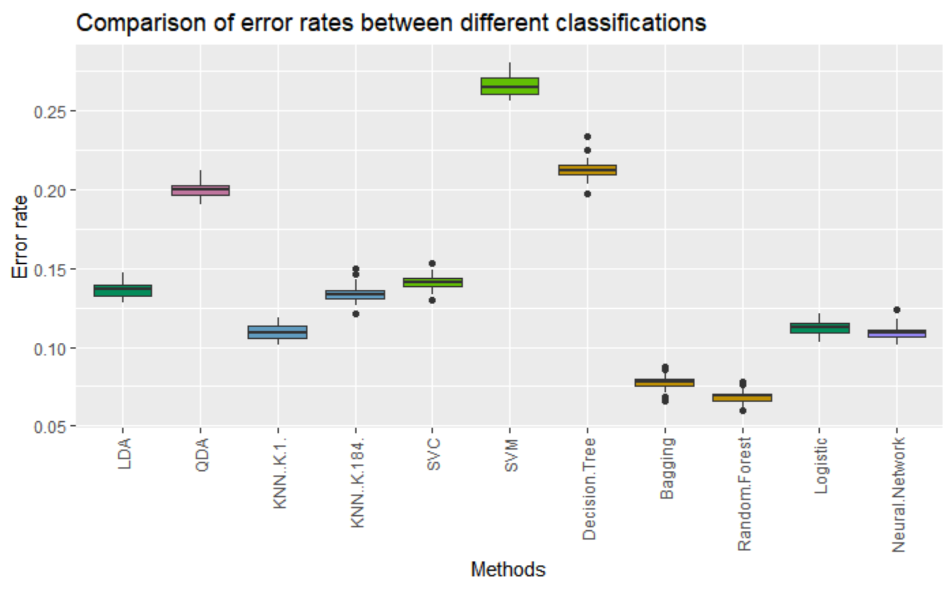
1) LDA
2) QDA
3) KNN: k = 1, k = 184
4) SVC(linear): cost = 5
5) SVM(polynomial): cost = 1, degree = 1
6) Decision Trees: 11 variables
7) Bagging: m = 169
8) Random Forest: m = 13
9) Logistic

10) Neural Networks: learning rate = 0.01

5.1 Result of Classification Model

Table 2 is the summary of the accuracy table for each method. All methods showed at least 73% accuracy. In particular, random forest showed the highest accuracy at about 93% with sufficiently high sensitivity and specificity. We also did a test error simulation. We randomly selected 30% of the data from the test set and calculated the test error. We repeated these steps 50 times and implemented a boxplot. As seen in Figure 4, the random forest has the lowest error rate, and other tests, such as neural networks and logistics, also have low enough error rate.

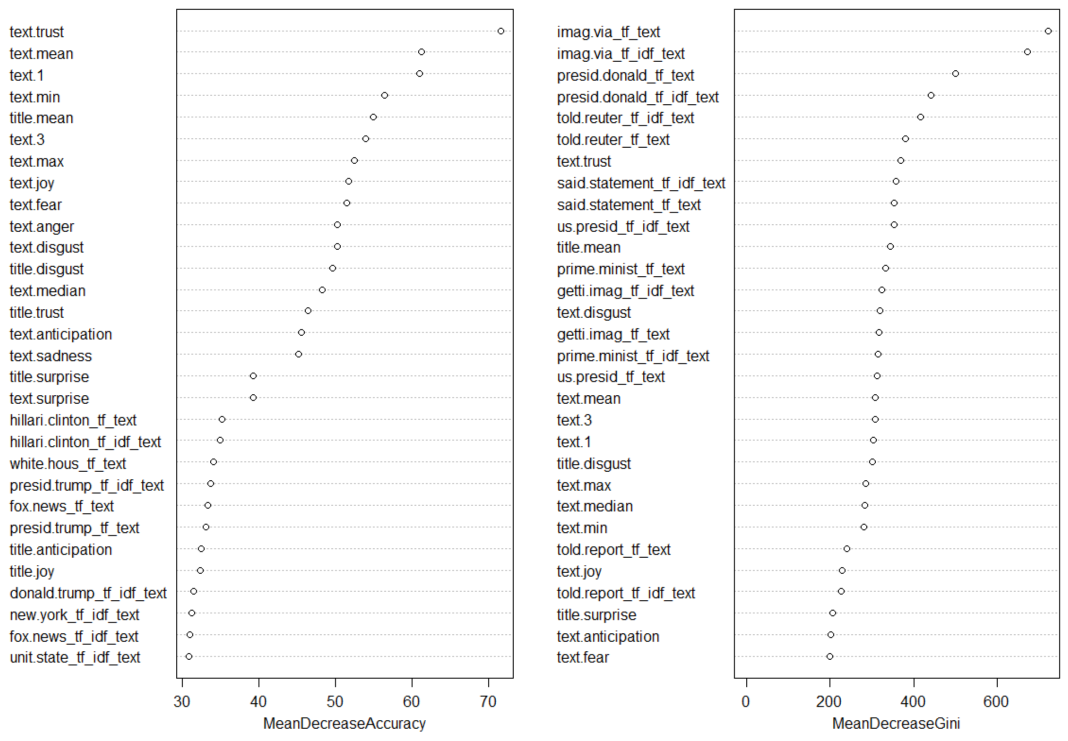
Figure 4. Boxplot of Classification Model Test Error



5.2 Interpretation of Best Model

To see which variables are important in our model, Figure 5 shows a ‘MeanDecreaseAccuracy’ plot drawn by our best model, Random Forest. ‘MeanDecreaseAccuracy’ predicts how much the model accuracy decreases when we drop a particular variable so it intuitively shows how important each variable is. Among the 170 variables, we can see that both our tf_idf and sentiment scores are important. Especially, individuals should be more alert to words related to trust as the sentiment of trust is a strong indicator of fake news. When reading news articles, a lower frequency of words like agree, accord, associate, assure, authority, certify, reassure, verified, and unquestionable may point to a possibility that the article is fake. Also, the average, and the lower quartile of the text sentiment scores are the next important variables. This informs us that fake news articles tend to have more negative sentiment compared to true news. In addition to the overall sentiment of the article, if the article contains sentences that are extremely negative, this may indicate that the article is fake. However, it is important to note that such textual characteristics are not a determinant of fake news, but that individuals should be more skeptical when reading articles with such characteristics.

Figure 5. Variable Importance of Random Forest Model



6. Conclusion

Our best performing model was the Random Forest model with m = 13. With this model, we can draw conclusions about which text characteristics are the most important in the identification of fake news articles. Unsurprisingly, one of the most important features that contributed to the accuracy of our model was the trust sentiment score. With this we found that articles that contained more words that were associated with “trust” were more likely to be real. In addition, the emotional valence of the text was another feature that our Random Forest model had indicated was integral in the classification process. Overall, fake news articles had a lower sentiment score than that of true news articles, which led us to the interpretation that fake news is written in a more negative voice compared to real news. Through our work, we have discovered that we can identify characteristics of fake news articles using various machine learning techniques. By applying this, we hope that it will help people stay vigilant of fake news given the characteristics that we deemed to be indicators of fake news. As for future work, we think that it will be interesting to utilize other feature extraction methods and test its performance against the two feature extraction methods that we explored. We would also like to see if our model accuracy would be improved with other machine learning methods, such as RNN. In addition, we would like to see the performance of our model applied on different datasets.

References

Bisaillon, C. (2020). *Fake and real news dataset* [Data set]. Kaggle. https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

Budgar, L. (2022, April 28). *Misinformation vs. disinformation: How to tell the difference.* Reader's Digest. https://www.rd.com/article/misinformation-vs-disinformation/

Jockers, M. (2020, November 24). *Introduction to the syuzhet package*. The R Project for Statistical Computing. https://cran.r-project.org/web/packages/syuzhet/vignettes/syuzhet-vignette.html

Juliasilge. (2017, June 10). *Juliasilge/janeaustenr: An R package for Jane Austen's complete novels*. GitHub. https://github.com/juliasilge/janeaustenr

Jurafsky, D., & Martin, J. H. (2021, December 29). *Speech and Language Processing (3rd ed. draft)*. https://web.stanford.edu/~jurafsky/slp3/

Spring, M. (2020, May 27). *Coronavirus: The human cost of virus misinformation*. BBC News. https://www.bbc.com/news/stories-52731624

Wang, S., & Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *50th Annual Meeting of the Association for Computational Linguistics, ACL 2012 - Proceedings of the Conference* (pp. 90-94)

Appendix

Detailed codes used can be found in the link below.

https://github.com/KyungJin-JinnySohn/STOR565_Machine_Learning