# Singtel: SMS Spam Filter Model

By Phong Pham

## 1. Introduction

### a) Problem statement

What challenges does Singtel face if they build an in-house spam filter system in order to reduce the number of spam messages received by customers by 75% at the end of 2021?

### b) Background

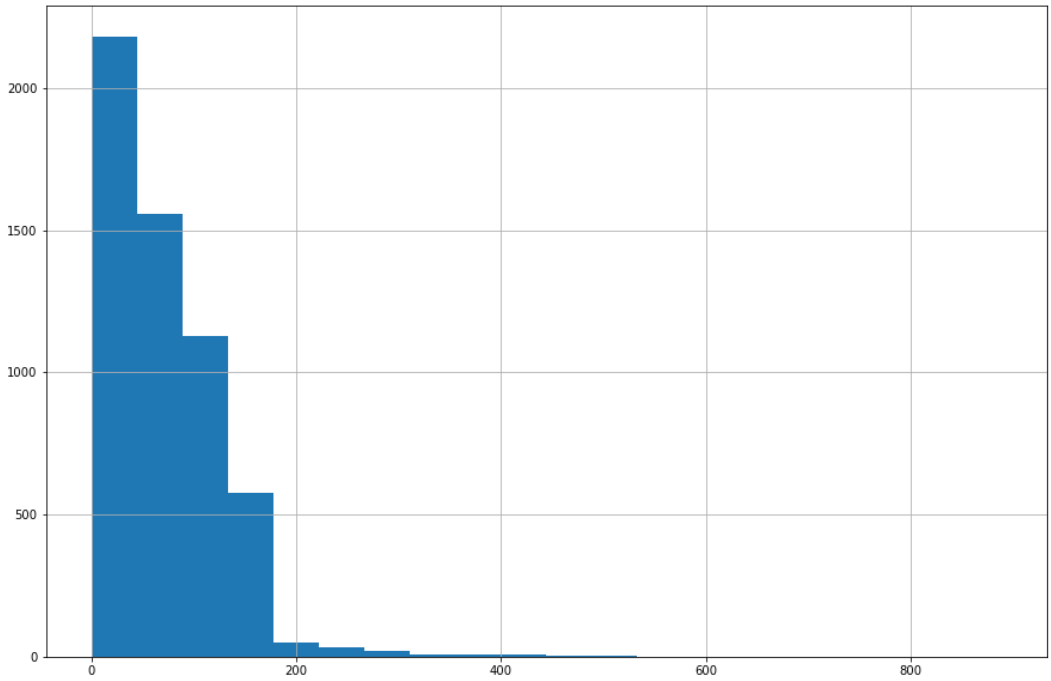
The management at Singtel, a leading telecommunication company in Singapore, is concerned with more and more customers complaining about the growing number of spam SMSes they receive on the mobile network and potentially leaving for another carrier. They would like the data scientist team to come up with an in-house SMS filter system that could detect spam messages and prevent them from being delivered to customers. Based on 5,574 SMSes collected from various sources, the company management wants to find data-driven solutions to reduce spam messages on the network and encourage customers to continue using their mobile plans.

### c) Goal

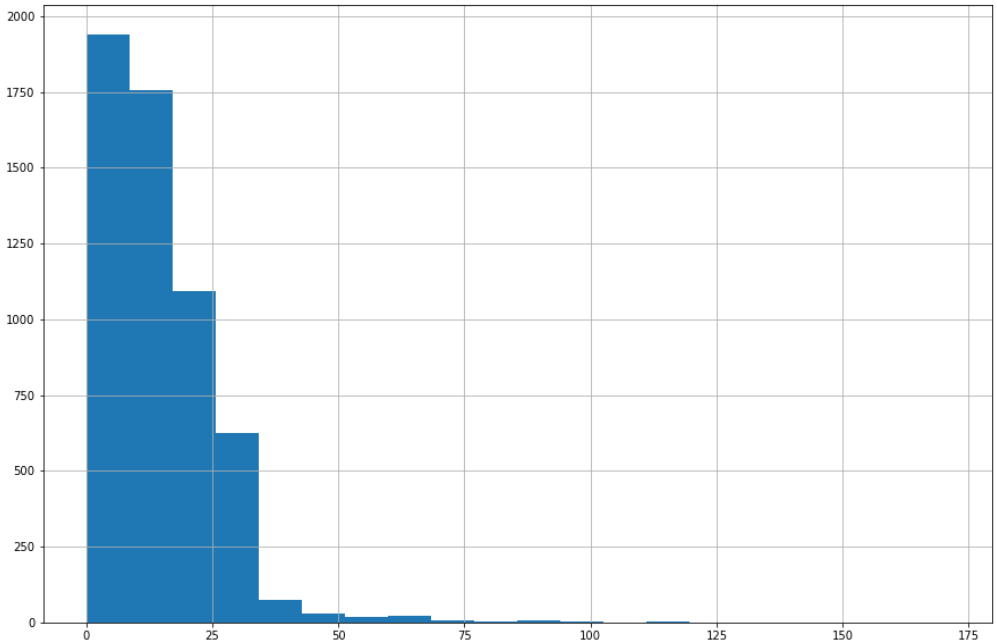
This project aims to provide a detailed plan to create a spam filter system that can reduce the number of spam SMSes by 75% by the end of 2021. We use a plain text file containing 5,574 SMS messages collected from free for research sources on the Internet, with the messages having been correctly categorized as either spam or ham (not spam). Based on our prediction model, Singtel can come up with an in-house spam filter system for their existing customers to keep their subscriptions with the company.

## 2. Data Wrangling

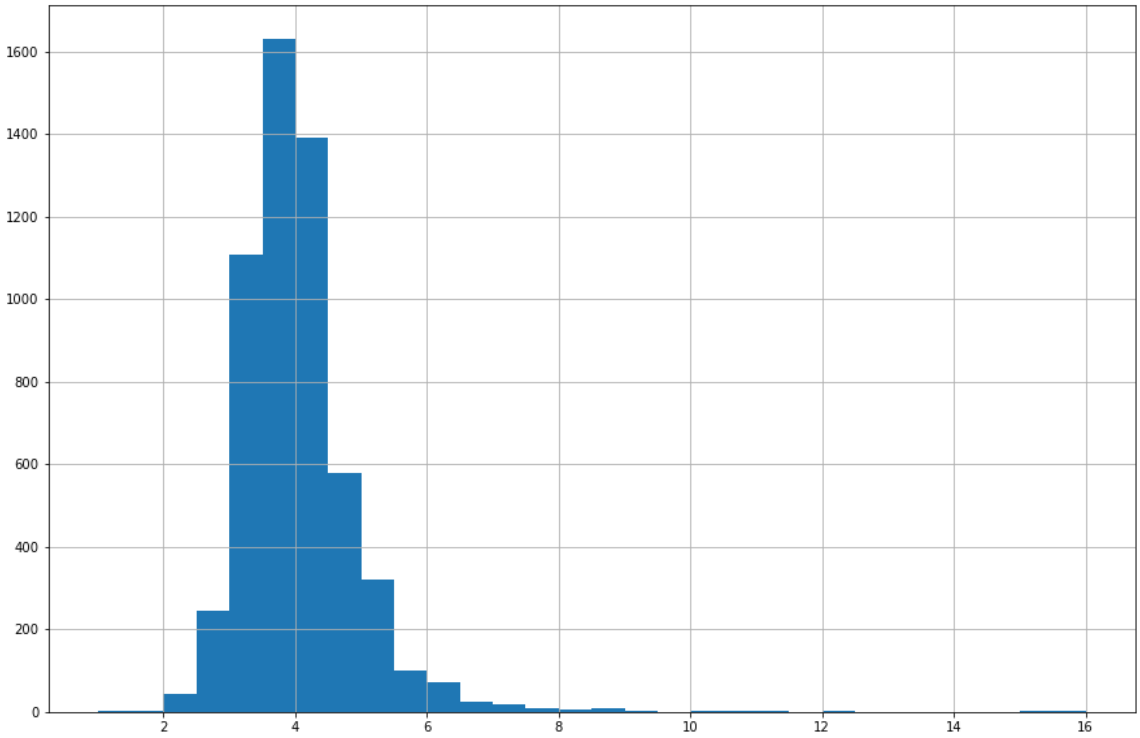
The original dataset contained 5,574 rows with the message and the spam flag columns. As these are text messages, a lot of emoticons, emojis, and non-alphanumeric characters were used. These often do not contribute to the prediction process, so we removed them from the messages. Having numbers in the messages are not useful either, especially texts usually contain stopword abbreviations like "to" => '2', "for" => 4. Therefore, we removing numbers from the SMS's too.




We can see that most SMS messages consist of fewer than 200 characters.



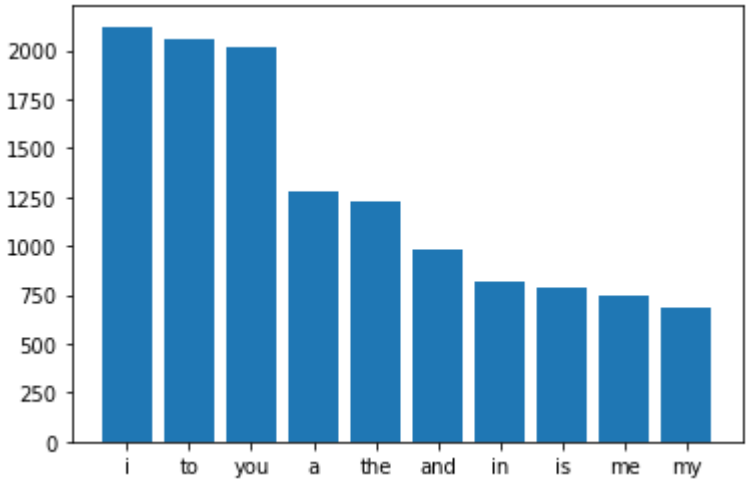

Most of the messages have 50 words or fewer, with a few outliers. The longest message has 171 words. As we went through the average word length, we found a message with an average word length of 37. This turned out to be a URL, so we removed it as it doesn't really contain any words.




Looking at the above histogram of the messages' average word length, we can see that some messages still have an average word length of 10 or greater. Upon investigation, these

messages are missing spaces after a punctuation or are also URLs. As there were only 14 records, we could also remove them.
We also found more than 400 duplicate messages in the dataset, so we removed them as well. After cleaning up the data, we were left with 5,112 messages, of which 614 are spam messages (12%) and 4,498 messages are not spam (88%).

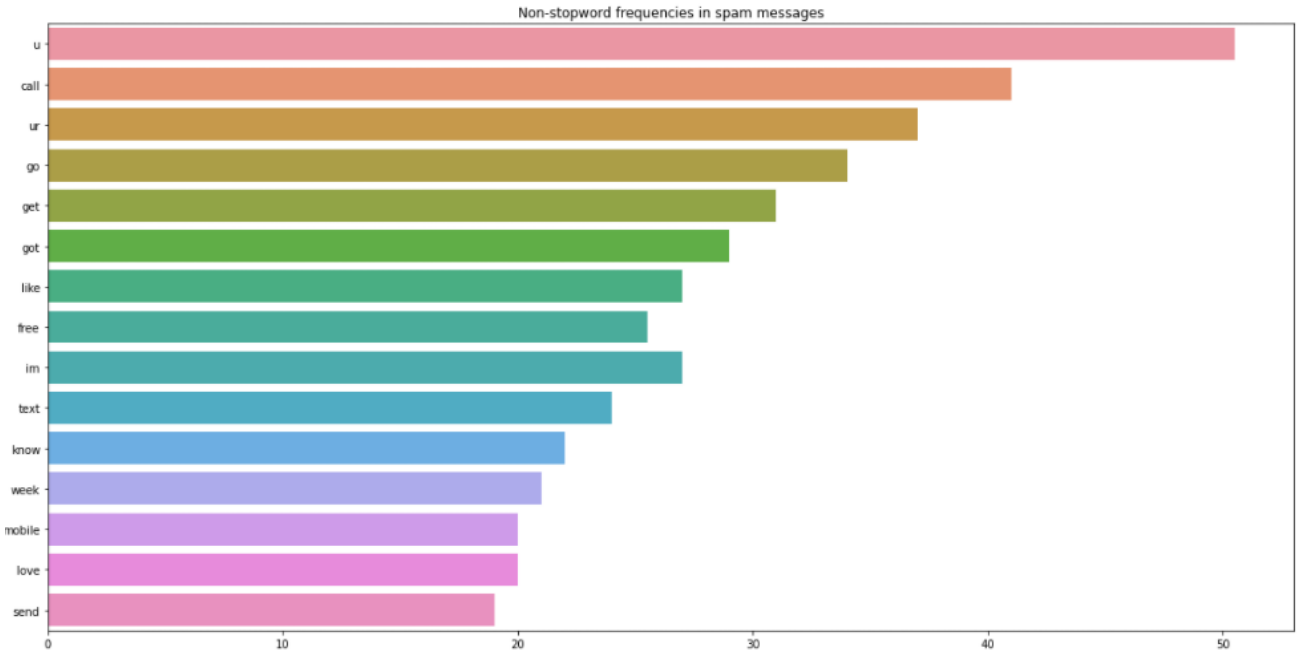
Messages might contain stopwords, which are words that are most commonly used in any language such as "the", "a", "an", etc. These words might have resulted in the above graph being left-skewed, i.e. shorter words appear more frequently.



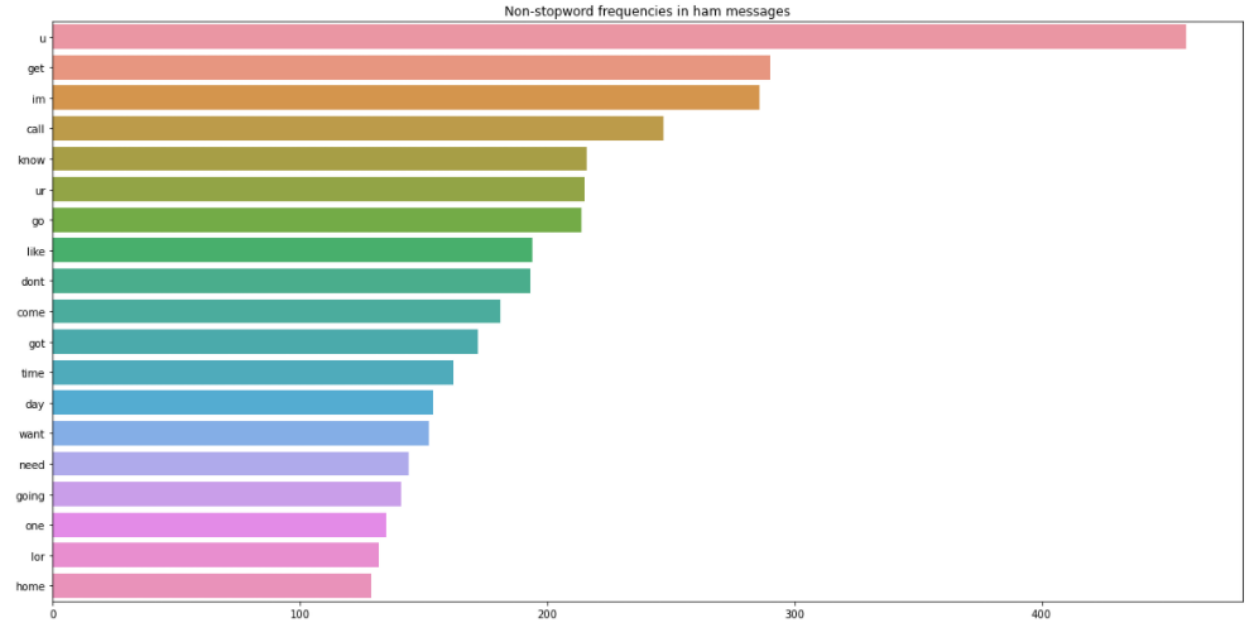

So the top 3 stopwords used were 'i', 'to' and 'you'. We removed these words from the messages as they carry minimal to no importance.

## 3. Exploratory Data Analysis

In this section, we explore the word usage frequencies in the messages. We first look at which single words are used the most after removing all stopwords.

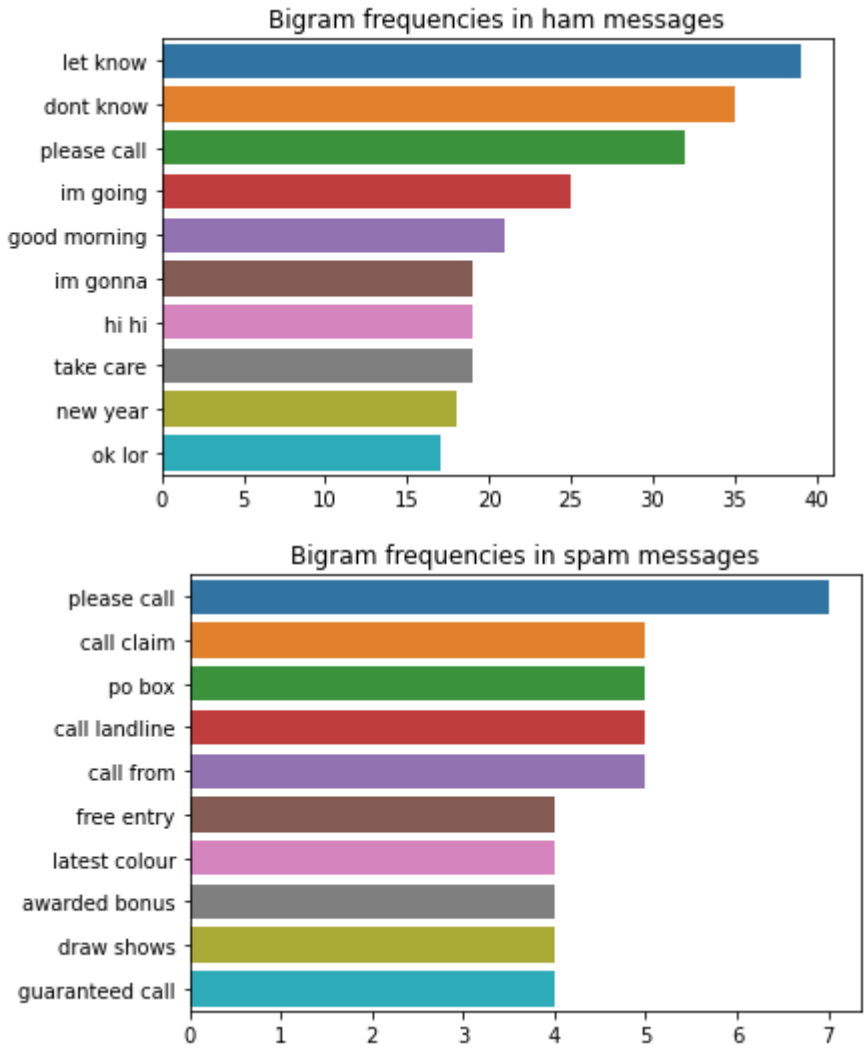



We can see that the top 3 words used in spam messages are “u”, “call” and “ur”. “U” and “ur” are short forms of “you” and “your”. These words might have been used frequently as spammers tend to use calls-to-action to encourage recipients (“u” and “ur”) to contact them (“call”). We can see other action verbs that encourage contacts like “text” or “send”. “Free” was also used quite frequently as a way to entice recipients to take action.



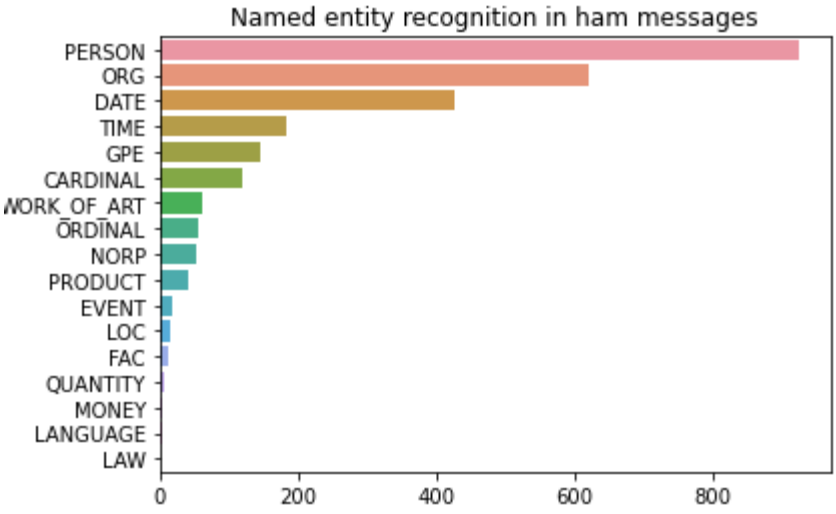
For non-spam messages, the top 3 most common words are “u”, “get”, “im” (I’m). These are used quite frequently in conversational language, so it is not surprising that they end up here. “Call” is the 4th most frequently used, probably because texters tend to ask others to call instead when something needs to be discussed at length.

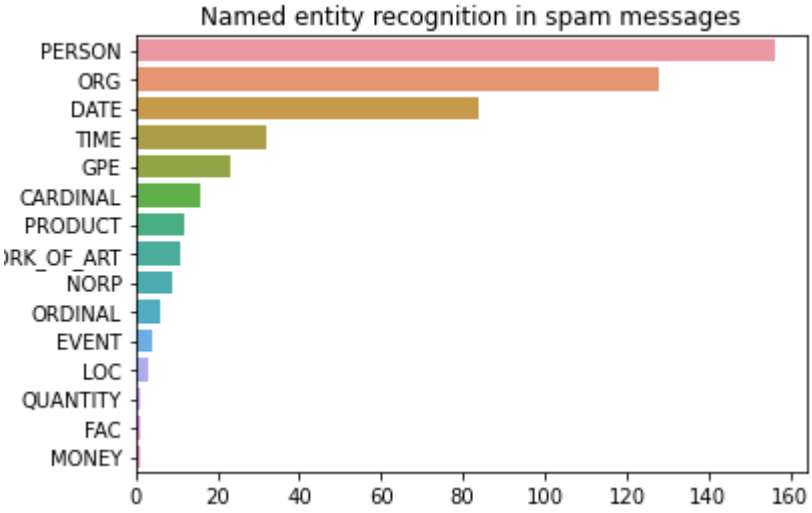
We now take a look at bigram (combination of two words) frequencies in both non-spam and spam messages.




One thing we can notice is that for spam messages, the top bigrams are some word combinations indicating the recipient has either won something or missed an important call. These usually are used to urge the recipient to take an action/call a certain number and are likely to have scams involved.


Looking at the wordclouds, we can see that spam messages usually mention something "Free" or related to money ("cash", "win" or a number most likely to represent an amount of money), and call-to-action words like "call", "text", "txt". On the other hand, ham messages do not have clear patterns.




We can see that people names have the highest frequency in non-spam messages (appeared in more than 800 messages). This is expected as the messages are similar to real-life conversations between people.

Organization names, people names and dates are in the top 3 of most used named entity types for both spam and non-spam SMSes.

## 4. Preprocessing and modeling

In order to further clean up the dataset, we performed lemmatization on the messages. Lemmatization is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form.

Lemmatization depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence.

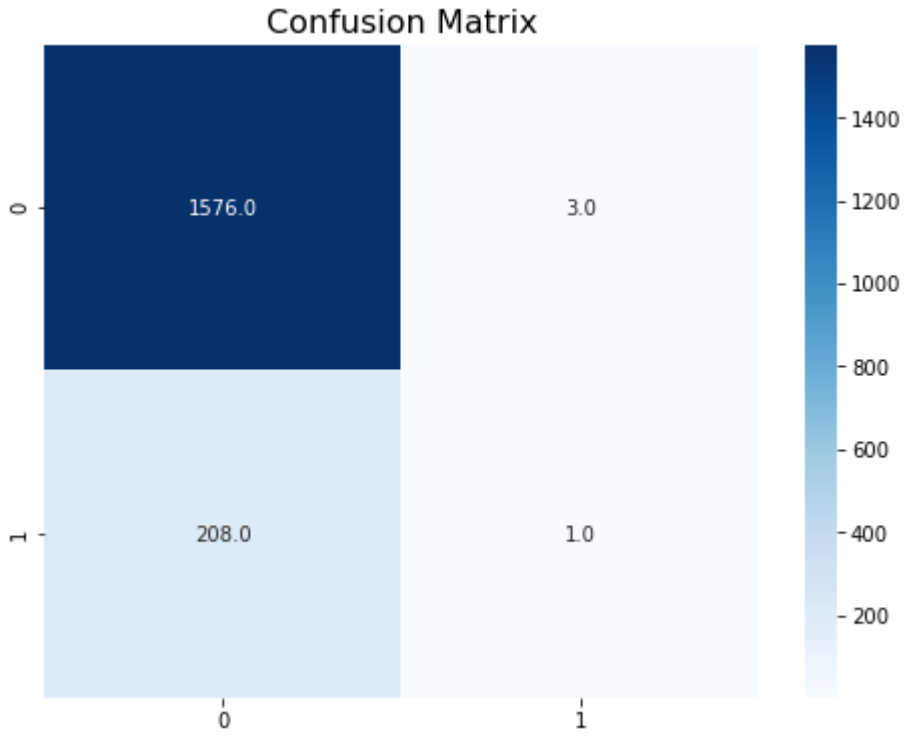
There is only 1 categorical variable in the dataset, the “flag” column, which contains two values “ham” and “spam”. In order to use the column in our prediction models, we converted them into dummy/indicator variables using Pandas one-hot encoding method get_dummies. After this step, the original column was transformed into a new column “flag_spam” with values 1 and 0, respectively indicating whether a message is a spam or not.

We then converted the messages into vectors using CountVectorizer, with each vector containing an array of token counts. After applying CountVectorizer, we used TF-IDF to find out how relevant a word is to each message in the dataset.

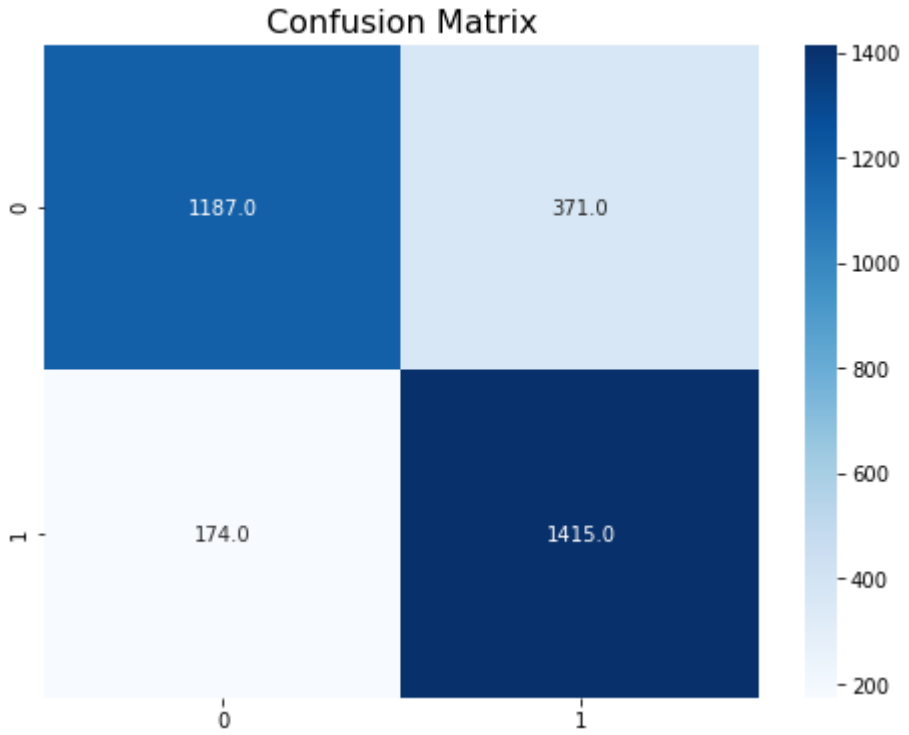
The value of TF-IDF increases proportionally to the number of times a word appears in the message, and is offset by the frequency of the word in the collection.

As spam messages account for only about 12% of total messages, there is a class imbalance in this dataset. We needed to handle this imbalance using SMOTE to oversample the minority class.

For comparison, we created a Support Vector Machine model on the dataset and ignored the class imbalance. This resulted in an accuracy score of 88% and a precision score of only 25%.



As we can see, it can correctly classify a high number of non-spam messages. However, 3 out of 4 spam messages were incorrectly classified (these 3 messages are in fact non-spam). The imbalance classes might have contributed to this issue.



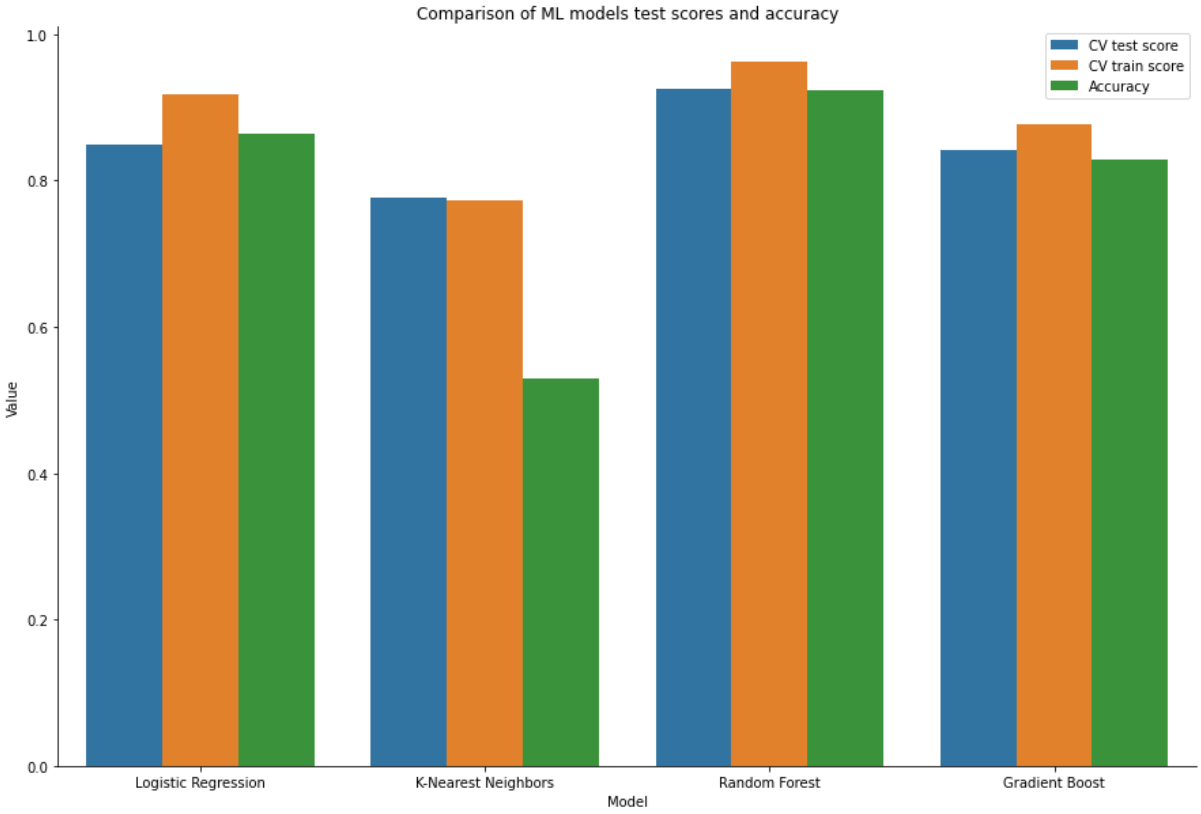

Using SMOTE, we oversampled the number of spam. The original dataset with 5,108 rows became a balanced one with 8,990 rows. When applying an SVM model to this new dataset, the accuracy dropped to 83%, but the precision score dramatically shot up to 79%. This means that the number of messages incorrectly marked as spam has decreased a lot when we used a balanced dataset.

As this is a classification problem aiming to predict which messages are spam, we will use the following machine learning models:

- Logistic Regression
- K-Nearest Neighbor (KNN)
- Random Forest
- Gradient Boost

We split the dataset into a training and a testing set with a 65:35 ratio, then applied different ML models and evaluated them using ROC-AUC scores. Here are the cross validation scores and accuracy of all models:

| | Model | CV test score | CV train score | Accuracy |
|---|---|---|---|---|
| 0 | Logistic Regression | 0.84942484 | 0.91834724 | 0.86336195 |
| 1 | K-Nearest Neighbors | 0.7757731 | 0.7722718 | 0.5300286 |
| 2 | Random Forest | 0.92601615 | 0.96237785 | 0.9237369 |
| 3 | Gradient Boost | 0.8414976 | 0.87610507 | 0.8290435 |



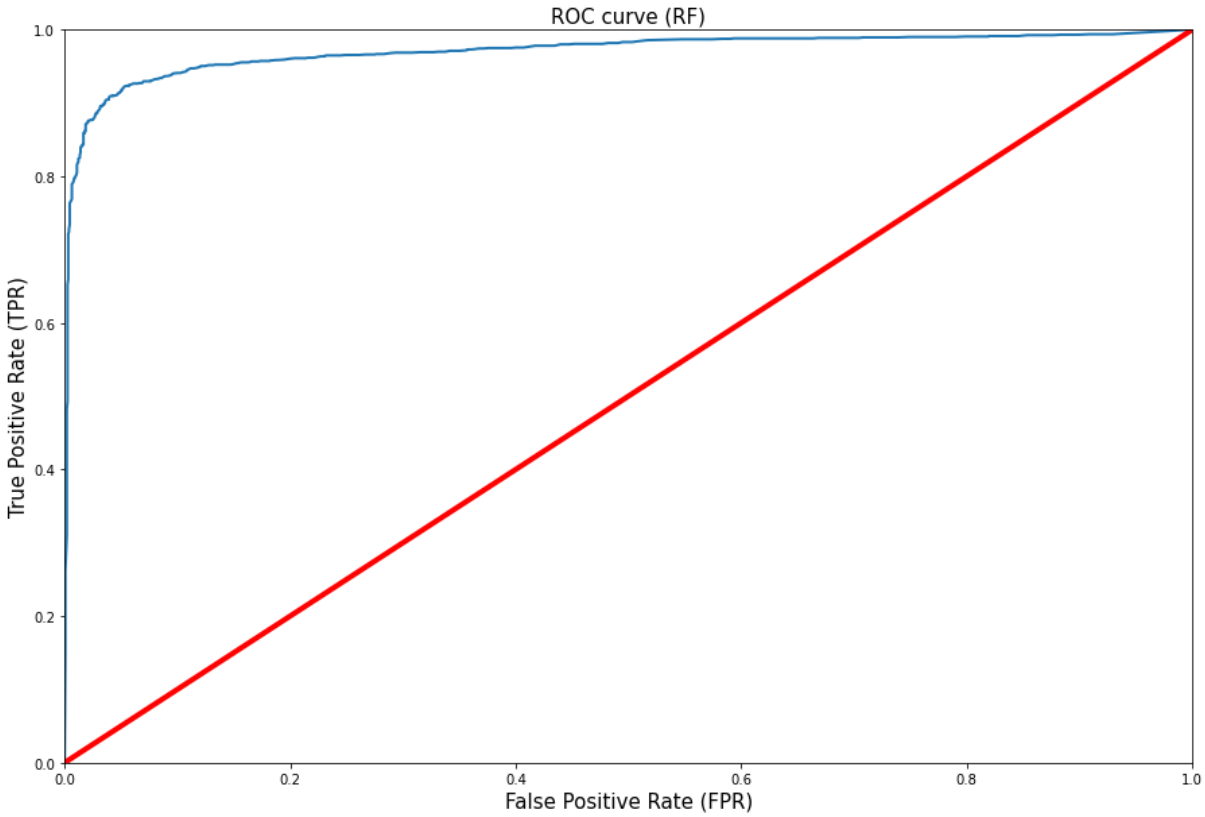

From the barplot, we can see that Random Forest and Logistic Regression are the best models to use for our spam filter. Therefore, we used these 2 models with optimal hyperparameters to predict the attritted flag value.

For the Random Forest model, we used the following hyperparameters:

RandomForestClassifier(bootstrap=False, n_estimators=400, random_state=42)

These optimal parameters resulted in an accuracy improvement of 0.98% when compared to the Random Forest model with default hyperparameters (93.35% vs. 92.37%). The precision score also improved from 93.6% to 94.8%.

This model also has very high ROC-AUC and F1 scores of 0.97 and 0.933, respectively. Below is the ROC curve of the model.

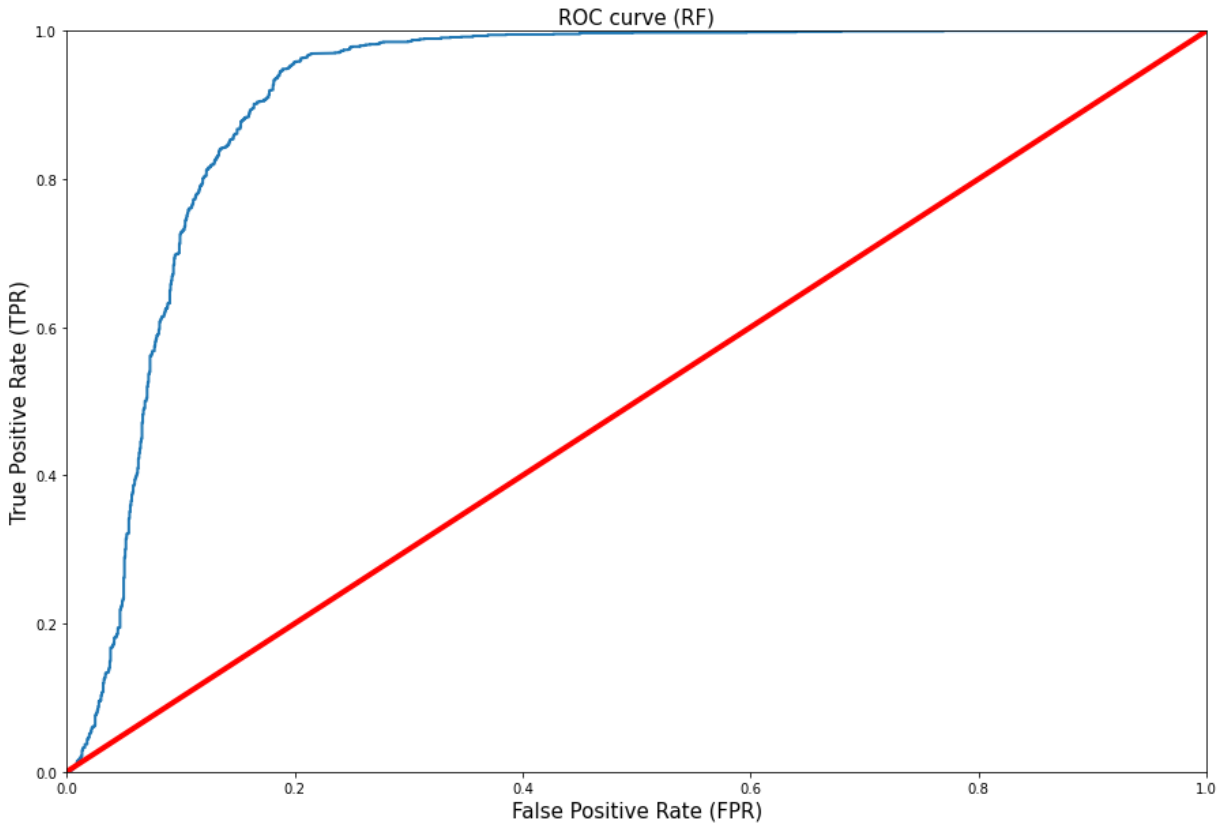



For the Logistic Regression model, we used the following hyperparameters:

LogisticRegression(C=10, fit_intercept=False, n_jobs=2, penalty='l1',

random_state=42, solver='liblinear', verbose=10)

These optimal parameters resulted in an accuracy improvement of 8.55% when compared to the Random Forest model with default hyperparameters (79.28% vs. 87.83%). The recall score also improved from 79.49% to 82.18%.




The ROC-AUC and F1 scores are also quite high, though not as high as those of the Random Forest model: 0.9128 and 0.8894, respectively.

## 5. Conclusion

In order to predict spam messages, here we have considered word vectors engineered from the original dataset containing more than 5,000 messages.

This is a classification problem. Here we have used the following classification models:

- Logistic Regression
- K-Nearest Neighbor (KNN)
- Random Forest
- Gradient Boost

We have evaluated each models in terms of model accuracy score, and 'ROC-AUC' score for both the training and test data, and plotted them. The two best performing models are Random Forest and Logistic Regression.

Using optimal parameters for our Random Forest and Logistic Regression models, we can see that Random Forest yields both higher accuracy and precision scores than Logistic Regression. Since we need to focus on keeping the number of false positives (i.e. the number of messages predicted as spam but are in fact not spam) as low as possible, we

want a model with higher precision. Therefore, the Random Forest model should be selected in this case for future spam predictions.