FLIP ROBO

Spam Classification Project

Submitted by:

Vedant Singh Shankar

ACKNOWLEDGMENT

I would like to thank my SME, Mr. Harsh Ayush, for providing me the opportunity to work on this project. The sample data is provided to me from the Flip Robo Technologies, which was shared to me for this project.

The following are the website links as a reference that helped me in the project.

Website Link:

- https://www.mygreatlearning.com/blog/multinomial-naive-bayes-explained/
- https://scikit-learn.org/stable/modules/svm.html
- https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes
- https://en.wikipedia.org/wiki/Tf%E2%80%93idf
- https://towardsdatascience.com/logic-and-implementation-of-a-spam-filter-machine-learning-algorithm-a508fb9547bd
- https://towardsdatascience.com/applied-text-classification-on-email-spam-filtering-part-1-1861e1a83246

Contents

1.INTRODUCTION

1.1 Background

Email is an important method of business communication that is fast, cheap, accessible and easily replicated. Using email can greatly benefit businesses as it provides efficient and effective ways to transmit all kinds of electronic data. Spam email is unsolicited and unwanted junk email sent out in bulk to an indiscriminate recipient list. Typically, spam is sent for commercial purposes. It can be sent in massive volume by botnets, networks of infected computers. Therefore, to have an effective communication, spam filtering is one of the important features to incorporate.

1.2 Problem Statement

To build a spam detection system to classify spam and ham emails.

1.3 Summary of the project

- In this project we have applied various text processing techniques to process the raw text into vectors before feeding to the Machine Learning algorithms.
- We have analysed the most frequent words for spam and ham emails in context of the given data.

- Due to imbalance in data-set as spam emails proportions being significantly smaller than the ham emails, we have used synthetic oversampling technique.
- We have extracted the features out from the processed text using popular feature extraction technique like **TF-IDF.**
- Build two different machine learning models and tuned their hyper-parameters and evaluated the models on various metrics.

2. Analytical Problem Framing

2.1 Modelling of the Problem

In this project we will be using two different kinds of Machine Learning algorithms to model our data. Here, we will be requiring sufficient amount of data consisting of emails having associated labels indicating whether email being a spam or ham. This email may contain just the body of the email or both subject and body. Hence, we will be modelling our spam detection system based on this data in a Supervised Learning fashion.

The high-level overview of each algorithm that is used to model the data is as follows:

Multinomial Naive Bayes (NB)

Multinomial Naive Bayes is one of the algorithms in a family of probabilistic algorithms which is based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of a feature.

It is used for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification. This distribution of the data is parametrized by vectors $\boldsymbol{\theta}_{y} = (\boldsymbol{\theta}_{y1},\ldots\ldots, \boldsymbol{\theta}_{yn})$.

Where, each class is denoted by **y** and **n** is the number of features or in text classification, the size of the vocabulary.

The parameter for each of the vocabulary is given by the Bayes' theorem on conditional probability as $\boldsymbol{\theta}_{yi} = \mathbf{P}\ (\mathbf{x}_i \mid \mathbf{y})$

Here, $\boldsymbol{\theta}_{yi} = \dfrac{\mathbf{A} + \boldsymbol{\alpha}}{\boldsymbol{B} + \boldsymbol{n\alpha}}$

Where, **A =** The number of times feature **i** appears in a sample of class **y** in the training set **T**.

B = The total count of all features for class **y**.

$\boldsymbol{\alpha}$ = Smoothing parameter, also a hyperparameter of this algorithm. This accounts for features not present in the learning samples and prevents zero probabilities in further computations, here, $\boldsymbol{\alpha} \geq \mathbf{0}$**.**

Advantages:

- Low computation cost.
- It can effectively work with large datasets.
- For small sample sizes, Naive Bayes can outperform the most powerful alternatives.
- Easy to implement, fast and accurate method of prediction.
- Can work with multiclass prediction problems.

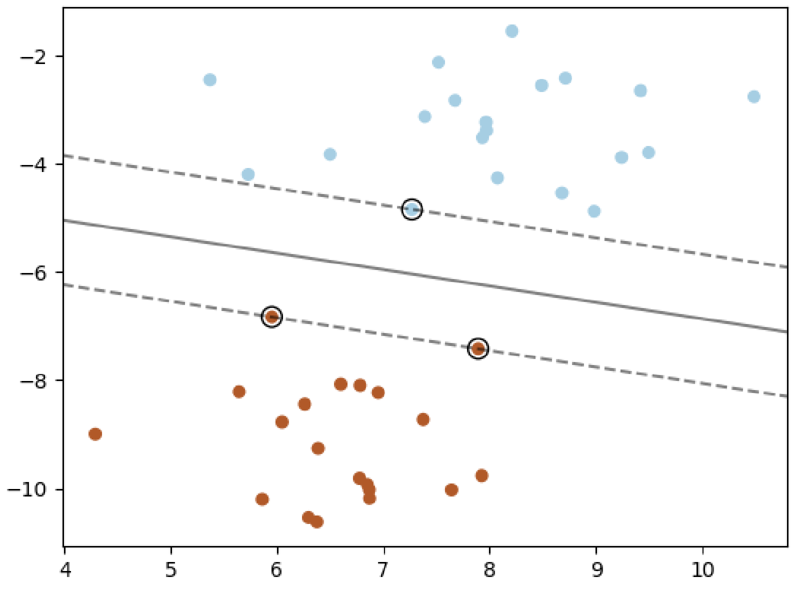
- It performs well in text classification problems

Disadvantages:

It is very difficult to get the set of independent predictors for developing a model using Naive Bayes

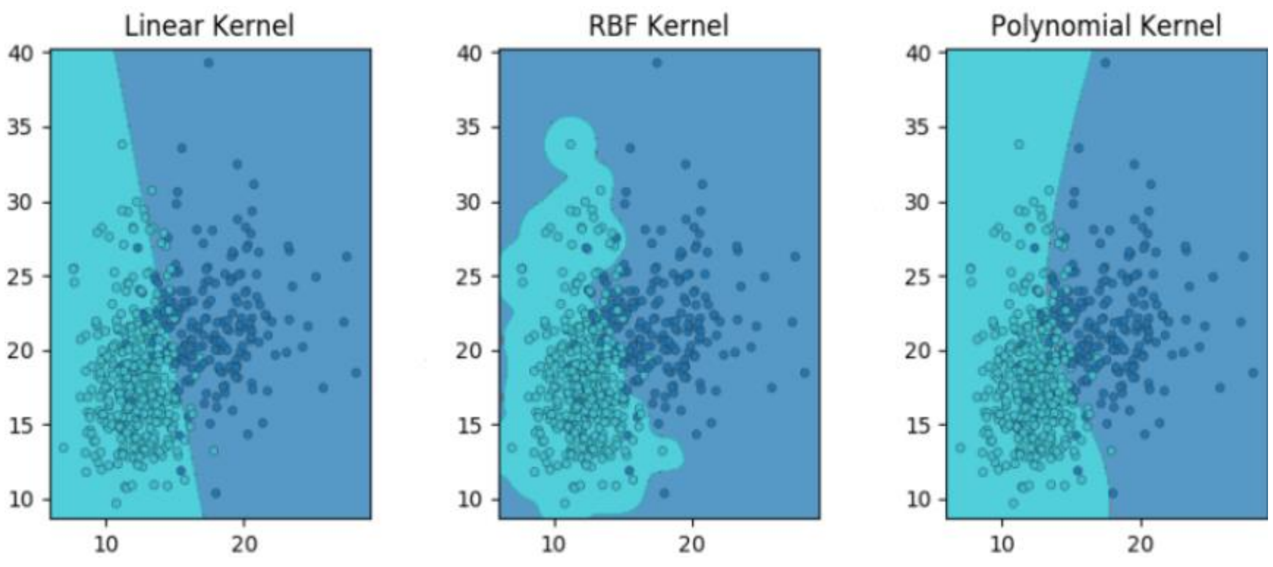
Support Vector Machines (SVM)

A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. The figure below shows the decision function for a linearly separable problem, with three samples on the margin boundaries, called “support vectors”. However, in general, when the problem isn’t linearly separable, the support vectors are the samples within the margin boundaries.



SVM decision boundary

Kernel Function: SVM kernels are mathematical functions that are the part of the algorithm where the input data is transformed into higher dimensions to possible form a hyper-plane that separate the classes. There are many SVM kernels but the most popularly used are **linear**, **polynomial** and **radial basis function (RBF)**. The performance of the algorithm largely depends on the type of kernel and the amount of data used. Linear kernels train very quickly compared to polynomial and RBF kernels which can take considerably longer on the same volume of data. The figure below gives the intuition of how SVM with these different kernels may classify a given data set.



SVM decision boundary for Linear, RBF and Polynomial Kernel

Advantages:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

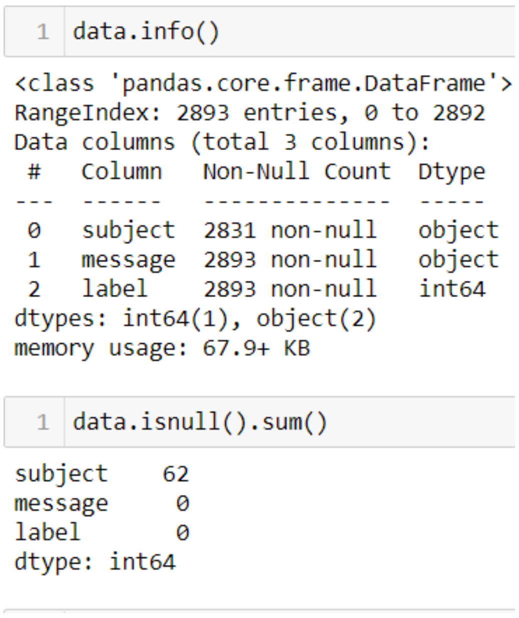
Disadvantages:

- If the number of features is much greater than the number of samples, to avoid over-fitting, choosing Kernel functions and regularization term is crucial.

- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

2.2 Data Sources and Description

- The sample data has been provided to us from the Flip Robo Technologies which is labelled as spam and ham as 1 and 0 respectively for the supervised learning task.
- It contains 3 columns and 2,893 rows and also contains 62 null values for ‘subject’ column as shown in the below snapshot.



```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2893 entries, 0 to 2892
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   subject  2831 non-null   object
 1   message  2893 non-null   object
 2   label    2893 non-null   int64
dtypes: int64(1), object(2)
memory usage: 67.9+ KB
```

```
data.isnull().sum()
```

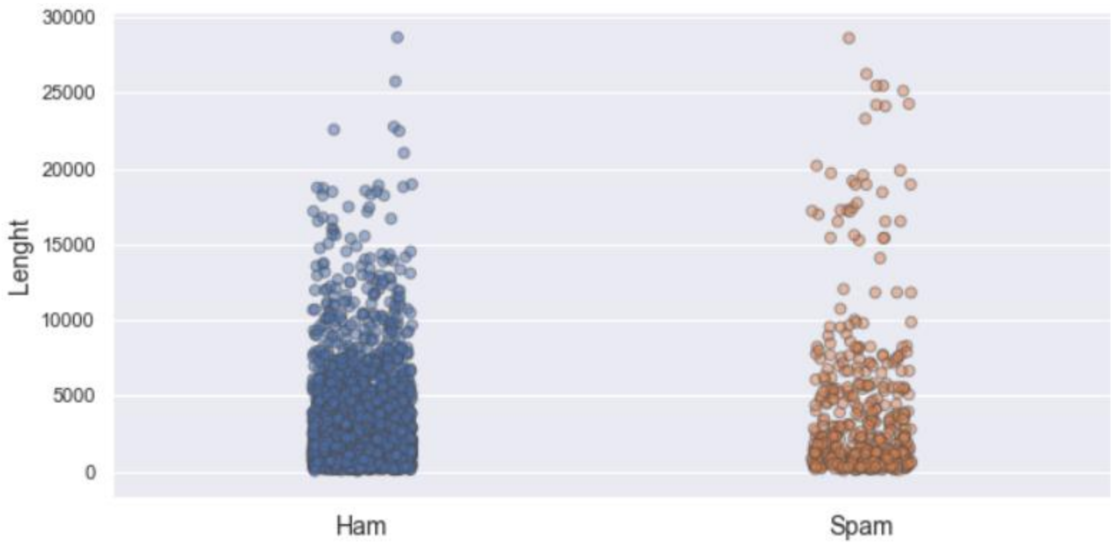
```
subject    62
message     0
label       0
dtype: int64
```

2.3 Data Preparation

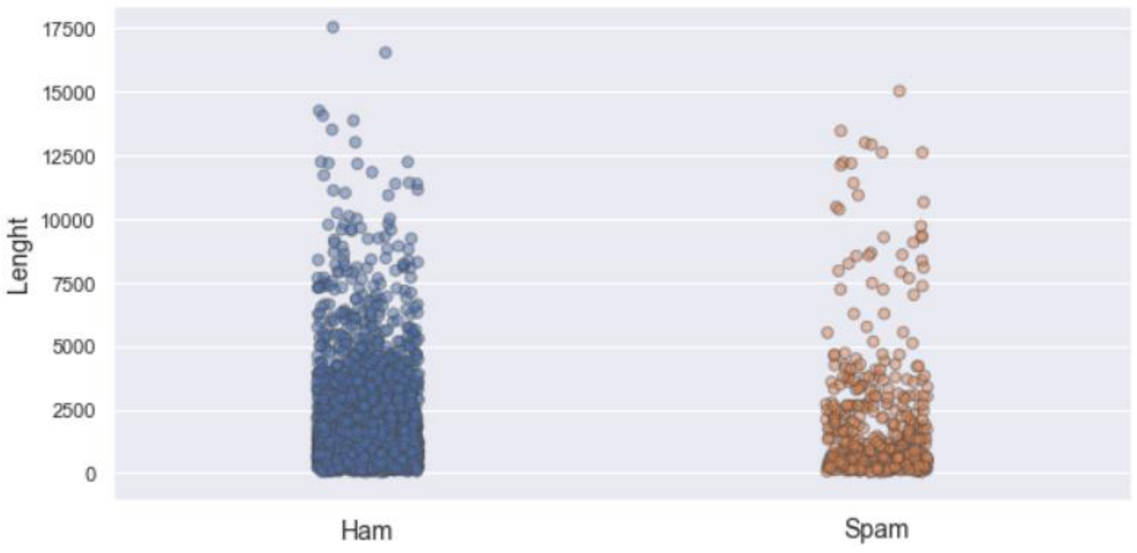
- The data contained many irregularities in the form of punctuations and whitespaces.
- The missing values from the ‘subject’ column was replaced by empty string and the columns ‘subject’ and ‘message’ was joined together to form a single column ‘message’.
- The length of the text from each row was extracted to form column ‘length’ for analysis of whether there is any difference in the length of messages between spam and ham mails.
- A pre-processing function **‘pre_process’** is defined to identify special characters form the email text like ‘**$**’, ‘**http’**, any digit of specified length (pertaining to phone or fax numbers) and ‘**@**’ (containing email address) to replace them with more generic words like ‘**money’**, ‘**link’**, ‘**call’** and ‘**email’** respectively. Also, any digit that are not in the specified length is discarded in this step.
- Within the above pre-defined function, before reassembling the words we took only those words that contribute to the meaning fo the document and discarded the words in ‘**Stop words’** (words that do not add any meaning to the document) and whose lengths that are less than 4. And then, we **lemmatized** each word for preserving the base word to analyse text more efficiently. Hence, by done so, we had cleaned and pre-processed the data for further analysis.

2.4 Visualization of the data

2.4.1 Visualization of the text length



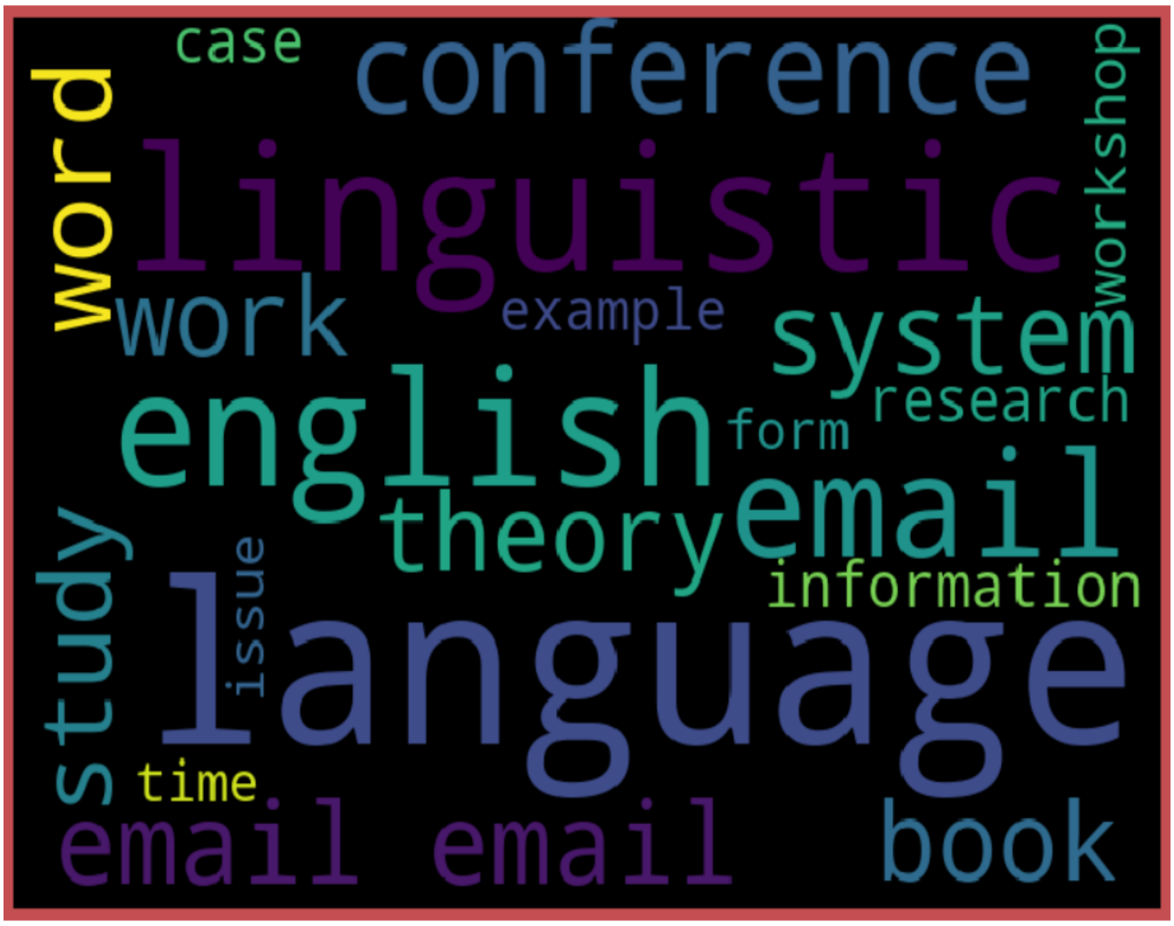
Scatterplot of un-cleaned text length



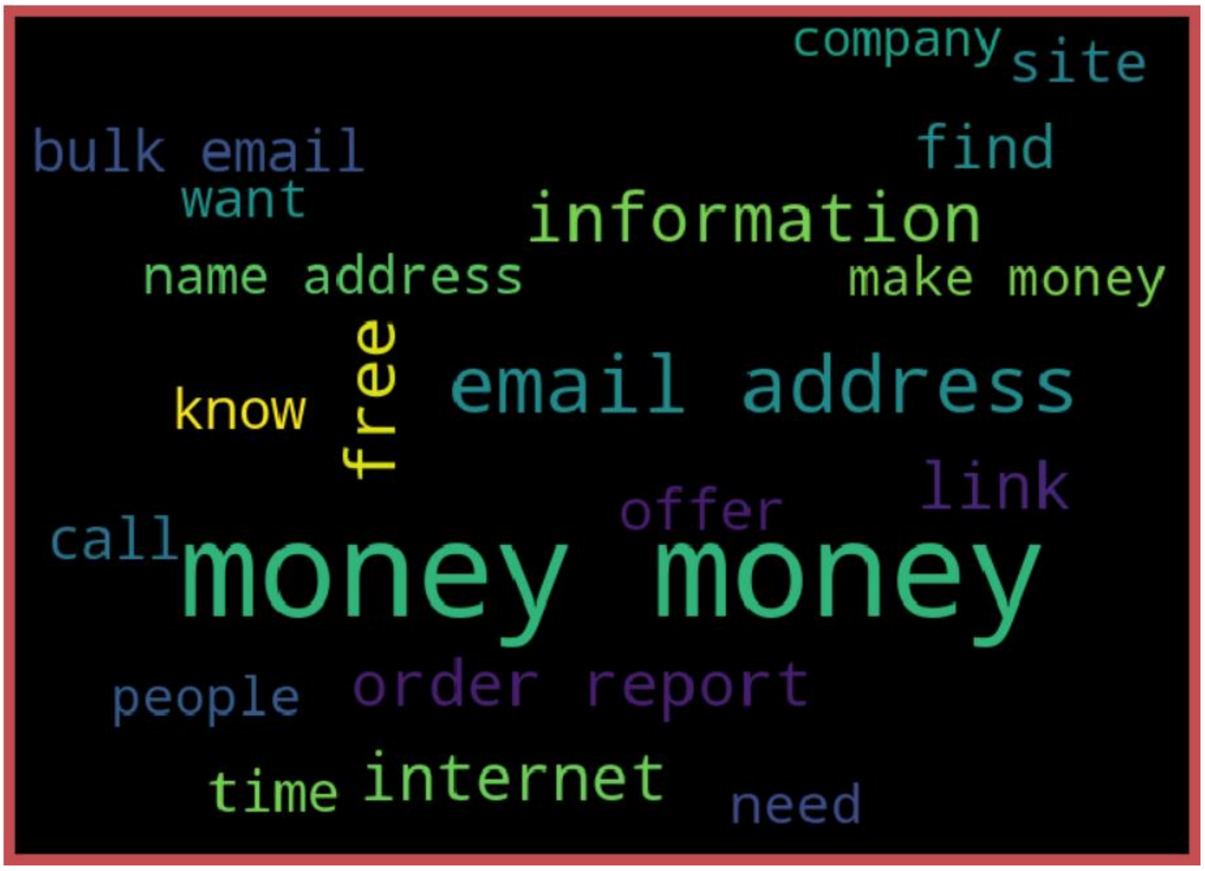
Scatterplot of cleaned text length

By analysing the above plots, we see that after cleaning the text data, the length of the text has reduced considerably. Also, for the cleaned text, the length of the spam mails is more densely populated on the lower half.

2.4.2 Visualization of most frequent words



Most frequent words of ham emails



Most frequent words of spam emails

From above tag clouds for ham and spam mails, we see clear distinction between most frequent words and the meaning that they are implying on the overall intention of the mail sent.

Words such as '**money'**, **'free'**, **'call'**, **'link'**, etc intuitively gave us the indication of the typical spam mails, whereas, for ham mails words such as '**work'**, '**conference'**, **'linguistics'**, '**theory',** etc may be more relevant for a typical ham mail.

It is to be noted that the most frequent words appearing on the above tag cloud may behave as a Stop words, that may appear on multiple documents and may not necessarily be useful in classifying whether the email is spam or not.

Therefore, to properly extract the useful feature form the given textual data, here we have used vectorization technique like **TF-IDF (Term Frequency – Inverse Document Frequency).**

2.5 Feature Extraction

TF-IDF (Term Frequency – Inverse Document Frequency).

TF-IDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modelling. The TF-IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general TF-IDF is one of the most popular term-weighting schemes today.

Term Frequency (TF)

It is simply the frequency in which a word appears in a document in comparison to the total number words in the document. Mathematically given as:

Term frequency = (Number of times a word appears in the document) / (Total number of words in the document).

Inverse Document Frequency (IDF)

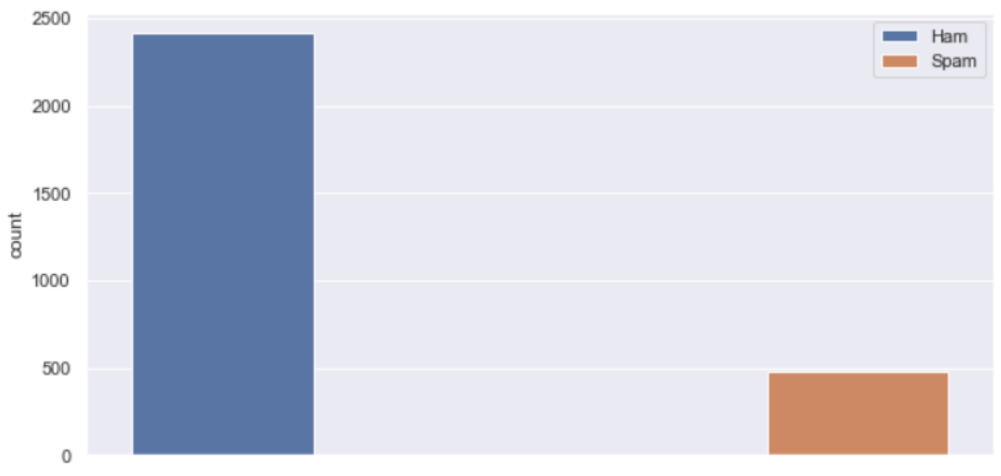
Term frequency has a disadvantage that it tends to give higher weights to words with higher frequency. In such cases words like ‘a’, ‘the’, ‘in’, ’of’ etc. or any stop words in general may appear more in the documents than other regular words. Thus, more important words are wrongly given lower weights as their frequency is less. To tackle this problem IDF was introduced. IDF decreases the weights of such high frequency terms by penalizing them and increases the weight of terms with rare occurrence. Mathematically it is given as:

Inverse Document Frequency = log [(Number of documents)/ (Number of documents the word appears in)]

Therefore, TF-IDF score for a given word is calculated as **TF*IDF.**

The code implementation of the above feature extraction process was done using scikit-learn module **tfidfvectorizer.** This technique was applied after the train-test split of the data set into training and test set, where the data was first fitted from the training set and then transformed both training and test set to prevent any data leakage.

Application of Synthetic Minority Over-sampling Technique (SMOTE) for imbalanced dataset



Target variable distribution

From the above plot, we see that our dataset is imbalanced and imbalanced classification involves developing predictive models on classification datasets that have a severe class imbalance. The challenge of working with imbalanced datasets is that most machine learning techniques will ignore, and in turn have poor learning performance on the minority class, and in our case, since the minority class is 'Defaulter', therefore, it is very crucial to properly capture its importance.

And this can be done by simply increasing the examples of minor class, which can be done by the simplest approach involving duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short.

In python, under **imbalanced-learn** API, we have used over-sampling methods like **SMOTE** for this task.

2.6 Hardware and Software Tools Used

Hardware used:

Processor: intel i7-9750 H

System type: 64-bit operating system, x64-based processor

Installed RAM: 16 GB

GPU: GeForce GTX 1650 (supports CUDA libraries)

Software and tools used:

Operating Software: Windows 10

Programming Language: Python

IDEs/Framework: Jupyter notebook using Anaconda Framework

Libraries and Packages used:

NumPy: Contains a multi-dimensional array and matrix data structures. It is utilised to perform a number of mathematical operations on arrays.

Pandas: It contains inbuilt Data-structures like Data-Frames and Series, which is used for data manipulation and analysis.

Matplotlib and Seaborn: Used as a data visualization tool.

Scikit-learn: It is used in implementation of various machine learning algorithms and it also contains various classes and functions for data pre-processing metrics for model evaluation.

Imbalanced-learn: This python library is used for implementing sampling algorithms to imbalanced dataset.

NLTK: It is a library used for statistical natural language processing for English. We have used stop words (in English) and class **WordNetLemmatizer** from this library.

WordCloud: It a python package used as a data visualization tool for representing text data in which the size of each word indicates its frequency or importance.

3. Model Development and Evaluation

3.1 Problem-solving approaches

- Th first step is to identify the if there are any missing values in the given data and replace them with appropriate values or in the case of textual data, we can just replace them with empty string.
- As our original data set contained both subject and message columns, and since it is better to combine the text of subject and message to analyse them together rather individually, therefore, we combined these two columns to form a single document in each row.

- Try to extract features like length of the text form each document to form a column ‘length’ to analyse whether there is any relation between it and the target variable with appropriate plots.
- Create a user defined function that is used to pre-process the text for cleaning and extracting useful information or words before feature extraction process. This is one of the most important steps as it may influence the overall performance of a ML models used.
- Visualize the cleaned text in the form of tag cloud to understand the most frequently appearing words in the document or corpus with respect to spam and ham emails.
- We can then split the dataset into training and testing set. Also, in order to prevent data leakage, it is very important to apply any feature extraction technique (in our case, TF-IDF) after the train-test split using training set and then to transform both training and testing set using statistics of training set.
- Finally, before modelling the data, we can apply oversampling techniques like SMOTE to balance the class in our target variable.

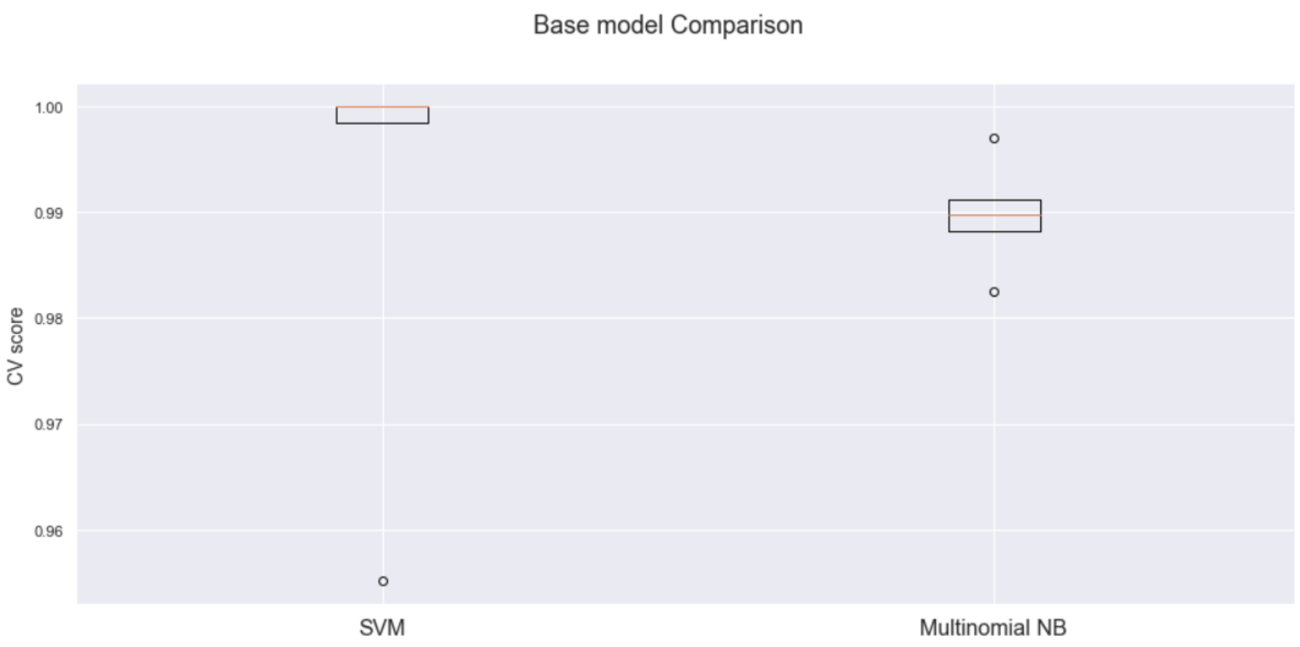
Algorithm used for Data Modelling

As mentioned earlier, we have used supervised learning algorithms that consists that are efficient in analysing large textual data like:

- Multinomial Naive Bayes
- SVM (Support Vector Machine)

3.2 Base Model Evaluation

Base model evaluation was done by Cross Validation (CV) on training set using k-fold = 5, that implies, 80% of data was used in training and 20% on testing, which was iteratively done on each of kth fold, and the final score being the average test score of all these folds. Also, we have used scoring function as f1 score, which is better metrics than accuracy, particularly in case of data imbalance.



By analysing the above boxplots, we see can see that SVM performs better than the Multinomial NB, although both the models score have outliers, but the score of SVM is less stable (and has high standard deviation) compared to the Multinomial NB due to the single extreme outlier in SVM score, this may imply the need of further regularization or hyper-parameter tuning of these models to reduce the any overfitting.

3.3 Train and Test set results

Metric	**SVM**	**Multinomial NB**
Accuracy %	100	99.53
F1 %	100	99.53

Train Results

Metric	**SVM**	**Multinomial NB**
Accuracy %	98.27	99.19
F1 %	94.51	97.61
ROC AUC %	99.96	99.96

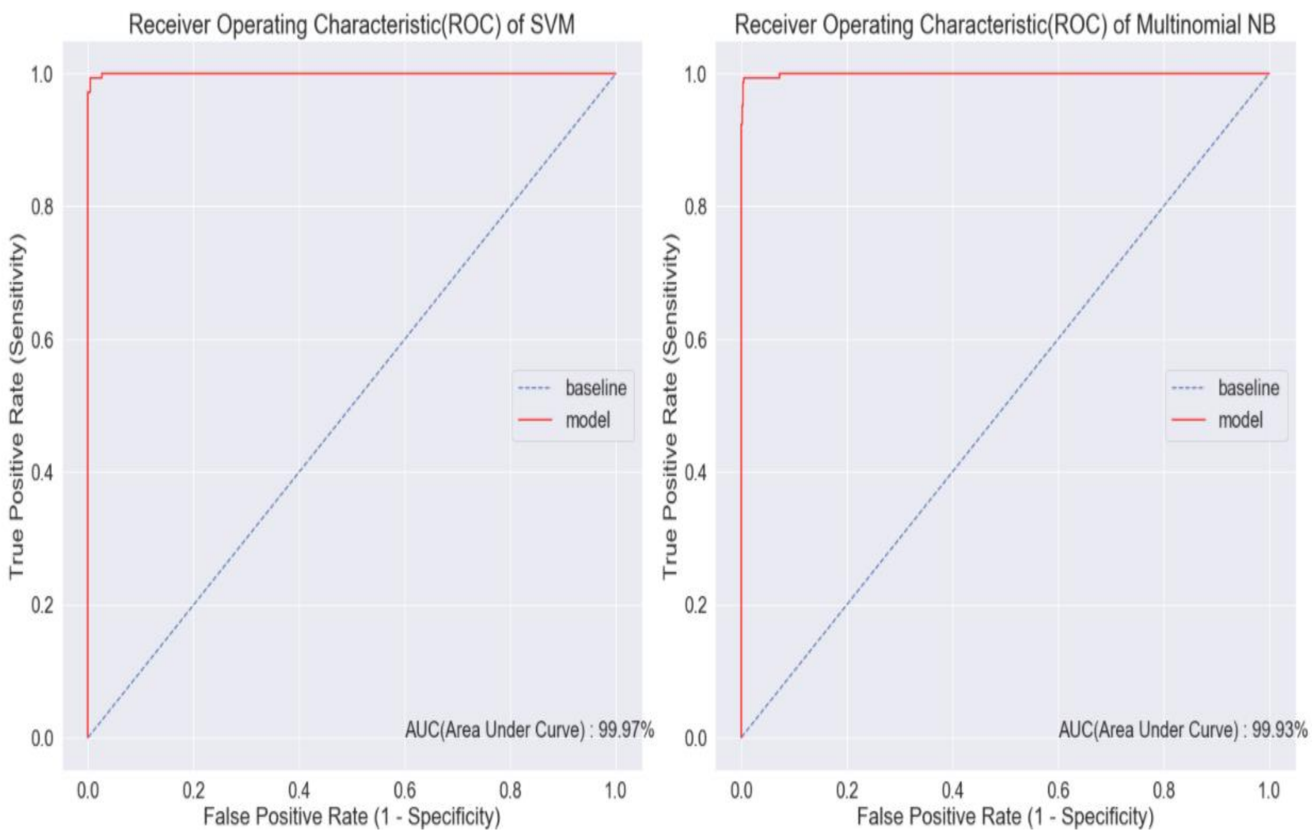
Test Results

By comparing the above train and test results, we see that as SVM overfitted a bit compared to the Multinomial NB, it scored less than the Multinomial NB on the test set.

3.4 Test results after Hyper-Parameter Tuning

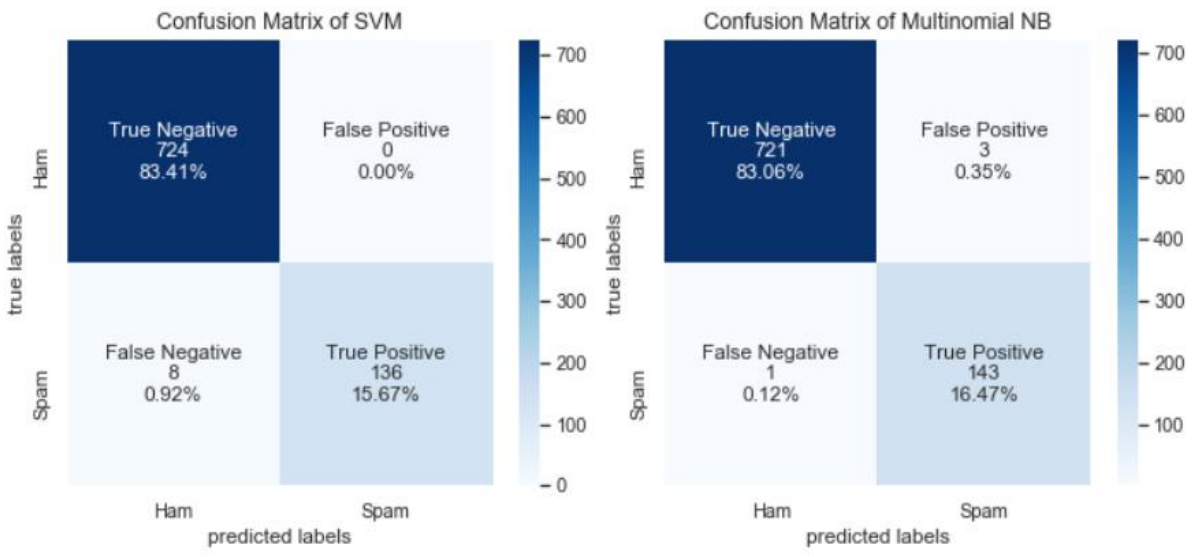
Metric	**SVM**	**Multinomial NB**
Accuracy %	99.08	99.54
F1 %	97.14	98.62
ROC AUC %	99.97	99.93

Test Results after Hyper-parameter Tuning



ROC curve after hyper-parameter tuning

Visualizing Confusion Matrix of Final Test Results with the help of Heatmaps.



3.5 Understanding Key metrics for success and the interpretation of the final test results

In a spam detection system, a classifier should be able to correctly classify both spam and ham mails. For example, **False positives and False negatives** become vitally important only in relation to spam filtering. A false negative (spam that ends up in your inbox) is un-acceptable, while false positive may be an important message that ends up in your spam folder or, much worse, gets deleted. Therefore, both **type -I** and **type-II** errors should be minimized.

Hence, **F1** score gives us the harmonic between recall and precision given by:

F1 = $\frac{Recall * Precision}{Recall + Precision}$

Here,

Recall = $\frac{True\ Positives}{True\ Positives + False\ Nagatives}$

Precision = $\frac{True\ Positives}{True\ Positives + False\ Positive}$

From above formula, we see that, in order to minimize the **False positives** and **False negatives,** both **precision** and **recall** should increase, which can be achieved by increasing the harmonic mean between the two given by **F1** score.

Hence, form above Final test result, we see that although both SVM and Multinomial NB almost performed the same in all metrics, but Multinomial NB performed bit better on the **F1** score. Also, by analysing the confusion matrix although we see that SVM has zero **type-I** error, but the overall sum of **type-I** and **type-II** errors in Multinomial NB is lesser.

Therefore, on a final note, based on the final test results, the SVM does better job at classifying the ham emails whereas, the Multinomial NB does better in classifying the spam emails.

4. CONCLUSION

- It was observed that, through proper cleaning and pre-processing of textual data, we visualized the most frequently appearing words. Also, these words were typically appearing in spam or ham mails in general.

- On an average, the spam mails do not tend to be longer than the ham mails. Having said that, there was not significant dependencies in the length of the mails with respect to the target variable.

- The most important and time-consuming part was the data analysis, cleaning, pre-processing and feature extraction with appropriate assumptions and methods, which requires the both, respective domain knowledge and efficiency in data manipulation. This is one of the most key steps before modelling the data.
- The main criteria for selecting the algorithms used for this project is to use algorithms that are effective in analysing large dimensional data, where number of features are larger than the rows or samples.

- Understanding the success metrics highly depends on the domain knowledge and interpretation of these metrics in the real-world scenarios. Like in our case, the success metrics was decided based on of importance of both spam and ham emails.

- For more robust and generalized model, it is recommended to extend the current work and collect recent email data with labels, and to work on the large volume of data to probably extract more useful vocabulary, which could further help to extract better feature for the model.