**Swathi Priya. N** **Mentor: Jatin Dixit**

# Capstone Project -2 SMS SPAM CLASSIFICATION

# FINAL REPORT

## Problem

In this project the main goal is to predict legitimate or spam SMS messages. The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. The files contain one message per line. Each line is composed of two columns: v1 contains the label (ham or spam) and v2 contains the raw text.

## Client

Public(Mobile User)

## Data Set

The dataset is taken from kaggle. This corpus has been collected from free or free for research sources at the Internet:

-> A collection of 425 SMS spam messages was manually extracted from the Grumbletext Web site. This is a UK forum in which cell phone users make public claims about SMS spam messages, most of them without reporting the very spam message received. The identification of the text of spam messages in the claims is a very hard and time-consuming task, and it involved carefully scanning hundreds of web pages. The Grumbletext Web site is: [Web Link].

-> A subset of 3,375 SMS randomly chosen ham messages of the NUS SMS Corpus (NSC), which is a dataset of about 10,000 legitimate messages collected for research at the Department of Computer Science at the National University of Singapore. The messages largely originate from Singaporeans and mostly from students attending the University. These messages were collected from volunteers who were made aware that their contributions were going to be made publicly available. The NUS SMS Corpus is avalaible at: [Web Link].

-> A list of 450 SMS ham messages collected from Caroline Tag's PhD Thesis available at [Web Link].

-> Finally, we have incorporated the SMS Spam Corpus v.0.1 Big. It has 1,002 SMS ham messages and 322 spam messages and it is publicly available at: [Web Link].

https://www.kaggle.com/uciml/sms-spam-collection-dataset

## Approach

- Loading Data
- Input and Output Data
- Applying Regular Expression
- Each word to lower case
- Splitting words to Tokenize
- Stemming with PorterStemmer handling Stop Words
- Preparing Messages with Remaining Tokens

- Preparing WordVector Corpus
- Applying Classification

## Classification Steps

- Data Preparation
- Exploratory Data Analysis(EDA)
- Text Pre-processing and TF-IDF
- Model Building with Classification Algorithm

## Deliverables

Trained classification model to be deployed in production to predict whether the SMS will Spam or Legitimate.

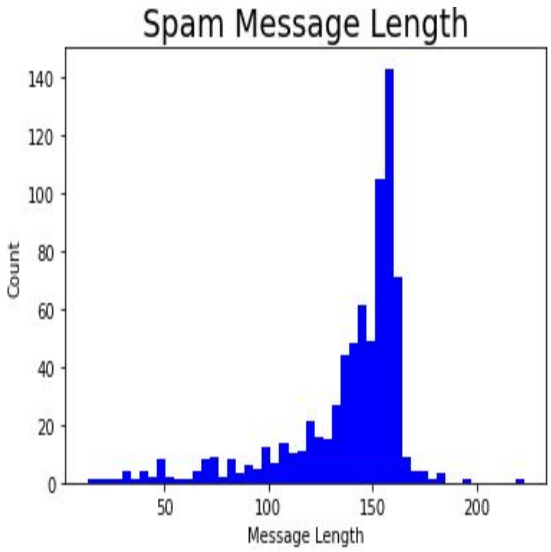
## 1. Data Preparation

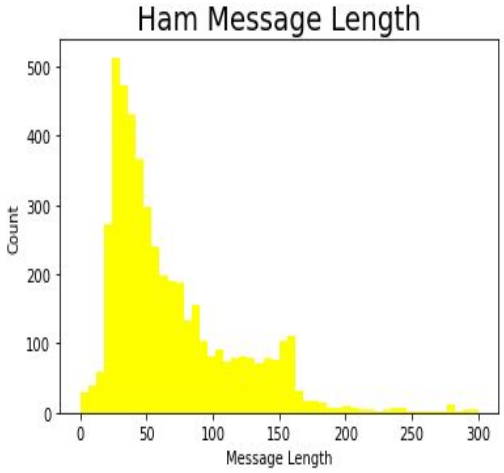
- Dropped unnecessary columns like unnamed 1, unnamed 2 and unnamed 3.
- Inserted new column called **count** to find the count the number of ham and spam messages
- Using info(), identify the null values if any and also the memory usage.
- Created corpus empty list for collective message and applied stemmer algorithm called PorterStemmer().

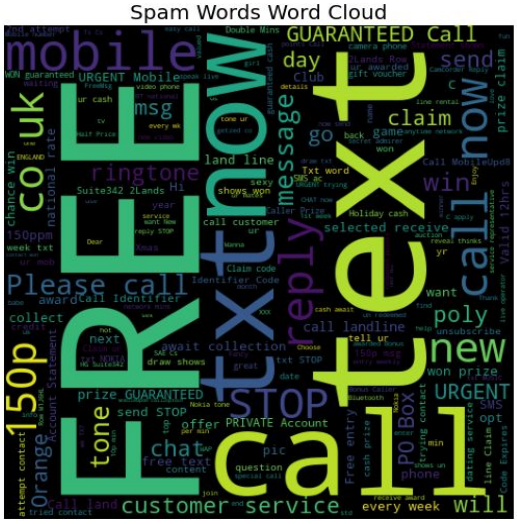
## 2. Exploratory Data Analysis

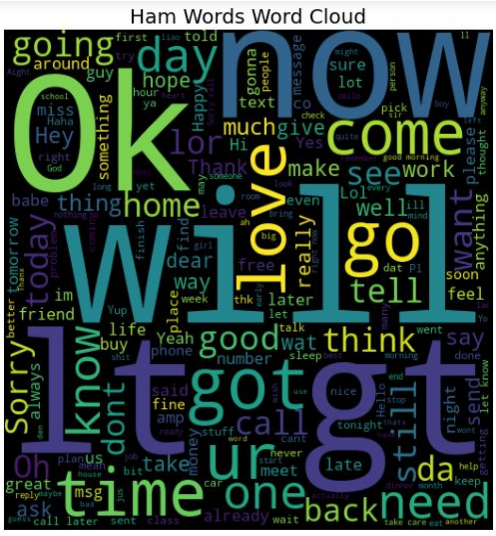
- Applied **groupby** to group the column information to show the ham and spam messages count and then used **describe** function to explore the statistical distribution of messages using mean , standard deviation ,etc.
- Replaced the column values and named it as label and sms instead of v1 and v2.
- Then applied count for both column label and sms which will give you the number of times a message is repeated.
- Using groupby function on column and describe : We can see the top msgs in ham and spam. Please call our customer service rep seems to be the most common spam message.
- Adding new column as sms length to find the length of the message
- **Visualize** the length of the ham and spam message by applying histogram.
- Inference of visualization is that we can see that sms with longer text tend to be spam.
- Then using word cloud we can find the repeated ham and spam word
- Inference of the word cloud code for spam words will be: we can see that sms containing words FREE,Please Call, Now , Win,Text,Call tend to be very common spam words
- Inference of the word cloud code for ham words will be: we can see the most common ham sms contain words will, know, gt (got), OK, know, Love,now.

**Visualization:**




## 3. Text PreProcessing

- PreProcessing Text - removing stopwords, punctuation and apply stemming
- Convert label to a numerical variable: Ham as 0 and Spam as 1

  sms = sms.translate(str.maketrans('','',string.punctuation)) This function removes punctuations and returns the sms as a list of words.

- Convert label to a numerical variable: df['label'] = df.label.map({'ham':0, 'spam':1})
- After dropping the unnecessary column we can see label, sms, sms length, then ham and spam message as 0 and 1 respectively.

**Using TF-IDF:** Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining.TF means Term Frequency. It

measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more often in long documents than shorter ones. Thus, the term frequency is often divided by the document length as a way of normalization.

TF = (Number of times term w appears in a document) / (Total number of terms in the document)

- Second part idf stands for Inverse Document Frequency. It measures how important a term is. While computing TF, all terms are equally important.
- IDF = log_e(Total number of documents / Number of documents with term w in it)
- TfidfVectorizer() is used by passing parameters as encoding, stopwords, analyzer,lower case, smooth_idf.
- **Steps for Vectorization**
  1. Import
  2. Instantiate
  3. Fit
  4. Transform
- Then splitting into train and test set by setting text_size as 0.30 and random state as 7.
- pd.DataFrame(features_train.todense(),columns=tfidf.get_feature_names())

**Inference:** The TFID scores : the more common the word across documents, the lower its score and the more unique a word is to our first document (e.g. ‘had’ and ‘tiny’) the higher the score. So it’s working as expected except for the mysterious a that was chopped off.

## 4. Building and evaluating a model

Chosen 3 types of model to predict the accuracy likelihood of spam detection.

1. Multinomial Naive Bayes
2. K-Neighbours Classifier
3. Decision Tree

### 1. Multinomial Naive Bayes classifier:

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. Instantiate a Multinomial Naive Bayes model

The parameters passed to the Multinomial Naive Bayes are:

- alphafloat, default=1.0
  Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing).

- fit_priorbool, default=True
  Whether to learn class prior probabilities or not. If false, a uniform prior will be used.

- class_priorarray-like of shape (n_classes,), default=None
- Prior probabilities of the classes. If specified the priors are not adjusted according to the data.
- Calculated the accuracy of class predictions

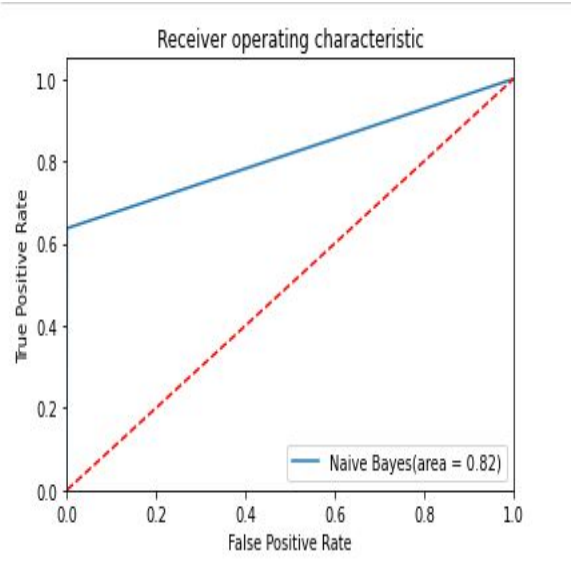
```
0.9509862522414824
              precision    recall  f1-score   support

           0       0.95      1.00      0.97      1450
           1       0.99      0.64      0.78       223

    accuracy                           0.95      1673
   macro avg       0.97      0.82      0.87      1673
weighted avg       0.95      0.95      0.95      1673
```

**Calculated AUC**

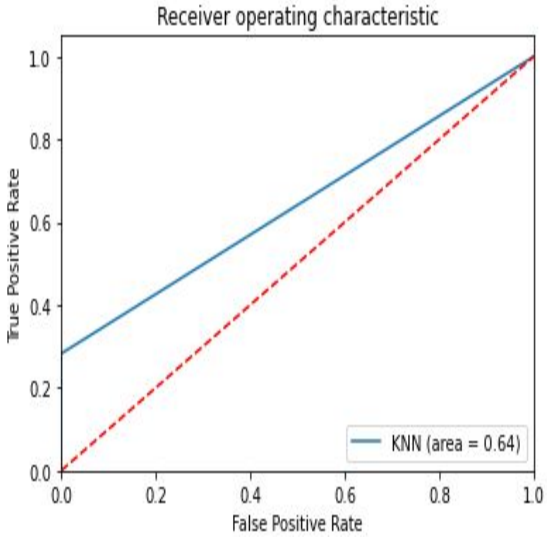
- AUC is useful as a single number summary of classifier performance.Higher value = better classifier.
- If you randomly chose one positive and one negative observation, AUC represents the likelihood that your classifier will assign a higher predicted probability to the positive observation.
- AUC is useful even when there is high class imbalance (unlike classification accuracy) Spam case Null accuracy almost 82%.

  Bottomline to how it thinks

  - Learns spamminess of each token
    - If have a lot of ham then class = ham
    - If have a lot of spam then class = spam




1. *Multinomial Naive Bayes classifier*




2. *K-Neighbours Classifier*

**2. K-Neighbours Classifier:**

Neighbors-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data.

- KNeighborsClassifier(n_neighbors=5)
  - Number of neighbors to use by default for kneighbors queries.
- Accuracy score 90% will be followed by fit and predict the model.

```
0.9043634190077705
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      1450
           1       1.00      0.28      0.44       223

    accuracy                           0.90      1673
   macro avg       0.95      0.64      0.69      1673
weighted avg       0.91      0.90      0.88      1673
```

**Calculated AUC:**

- AUC is useful even when there is high class imbalance (unlike classification accuracy) Spam case Null accuracy almost 64%.

**3. Decision Tree:**

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

1.Predicting

- DecisionTreeClassifier(random_state=50). Random_state controls the randomness of the estimator.
- To obtain a deterministic behaviour during fitting, random_state has to be fixed to an integer. Here I have taken it as 50.
- The accuracy will be predicted as 95%.

Evaluating using confusion matrix

The confusion matrix gives an overview of the classification results:

- The diagonal elements represent the number of points for which the predicted label is equal to the true label,
- while off-diagonal elements are those that are mislabeled by the classifier.
- The higher the diagonal values of the confusion matrix the better, indicating many correct predictions.

- The rows of a confusion matrix correspond to the true (actual) classes and the columns correspond to the predicted classes.

So, all together the confusion matrix for a binary classifier consists of 4 values:

```
[[1436   14]
 [  63  160]]
```

```
Accuracy : 0.95397

              precision    recall  f1-score   support

           0       0.96      0.99      0.97      1450
           1       0.92      0.72      0.81       223

    accuracy                           0.95      1673
   macro avg       0.94      0.85      0.89      1673
weighted avg       0.95      0.95      0.95      1673
```

**Final Result based on Accuracy:**

The final result based on the accuracy is that decision tree hold the high accuracy over KNN and Multinomial NAive Bayes

- Decision Tree : 95.39%
- KNN classifier : 90.43%
- Multinomial Naive Bayes:95.09%