



# **SPAM COMMENTS CLASSIFICATION PROJECT**

Submitted by:  
**TAMALI SAHA**

FlipRobo SME:  
**GULSHANA CHAUDHARY**

## ACKNOWLEDGMENT

I would like to express my special gratitude to Flip Robo Technologies team, who has given me this opportunity to deal with this dataset during my internship. It helped me to improve my analyzation skills. I want to express my gratitude to Ms. Gulshana Chaudhary (SME, Flip Robo) as she has helped me to get out of all the difficulties I faced while doing the project. I also want to give huge thanks to entire DataTrained team.

### **Bibliography:**

Reference used in this project:

1. Hands on Machine learning with scikit learn and tensor flow by Aurelien Geron.
2. Andrew Ng Notes on Machine Learning (GitHub).
3. Different projects on Github and Kaggle.
4. Different conference papers on Recharchgate.
5. "A Review of Machine Learning Approaches to Spam Filtering", Thiago S. Guzella et al. in 2009.
6. S. Ananthi (2009) ,"Spam Filtering using K-NN"
7. "A Survey on Spam Detection Techniques", Anjali Sharma et al. in 2014
8. Simon Tong and Daphne Koller (2001), "Support Vector Machine Active Learning with Applications to Text Classification".

## Table of Contents

1. Introduction.....	4
1.1 Business Problem Framing.....	4
1.2 Conceptual Background of the Domain Problem .....	4
1.3 Review of Literature.....	4
1.4 Motivation for the Problem Undertaken .....	5
2. Analytical Problem Framing.....	6
2.1 Mathematical/ Analytical Modelling of the Problem.....	6
2.2 Data Sources and their formats .....	6
2.3 Data Pre-processing Done:.....	6
2.3.1 Feature Engineering:.....	6
2.3.2 Drop unnecessary columns: .....	7
2.3.3 Calculate length before cleaning of 'message_body' column: .....	7
2.3.4 Encoding type dataset: .....	7
2.3.5 Length of message_body: .....	8
2.3.6 Natural Language Processing: .....	8
2.4 State the set of assumptions (if any) related to the problem under consideration.....	9
2.5 Hardware and Software Requirements and Tools Used .....	9
3. Model/s Development and Evaluation.....	10
3.1 Identification of possible problem-solving approaches (methods): .....	10
3.2 Testing of Identified Approaches (Algorithms) .....	10
3.3 Key Metrics for success in solving problem under consideration: .....	10
3.4 Run and Evaluate selected models .....	11
3.5 AUC- ROC Curve: .....	13
3.6 Hyper Parameter Tuning: .....	13
3.7 Final Model: .....	14
3.8 Confusion Matrix: .....	14
3.9 Load the model:.....	14
3.10 Visualizations: .....	15
3.11 Word Cloud for Spam sms: .....	16
3.12 Interpretation of the Results .....	17
4. CONCLUSION.....	18
4.1 Key Findings and Conclusions of the Study .....	18
4.2 Learning Outcomes of the Study in respect of Data Science .....	18
4.3 Limitations of this work and Scope for Future Work .....	18

## **1. Introduction**

### **1.1 Business Problem Framing**

The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according being ham (legitimate) or Spam.

Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the nonspam texts. It uses a binary type of classification containing the labels such as 'ham' (nonspam) and spam. Application of this can be seen in Google Mail (GMAIL) where it segregates the spam emails in order to prevent them from getting into the user's inbox. The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text.

### **1.2 Conceptual Background of the Domain Problem**

A collection of 5573 rows SMS spam messages was manually extracted from the Grumble text 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.

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.

### **1.3 Review of Literature**

Text classification is used to evaluate whether an incoming message or piece of mail should go directly to the inbox or the spam folder. It is the process of categorising texts based on their contents. It is employed to arrange, organise, and classify text. Either manually or automatically can be done. Text is automatically classified by machine learning significantly more quickly than manually. Pre-labeled text is used by machine learning to learn the many connections between text and output. Each text was converted to a numerical representation in the form of a vector that represents the frequency of words in a predetermined lexicon using feature extraction.

Most research has been conducted into detecting and filtering spam email using a variety of techniques.

The study "A Review of Machine Learning Approaches to Spam Filtering" was conducted by Thiago S. Guzella et al. in 2009. In their article, they discovered that Bayesian filters, which are used to filter spam, need a lengthy learning period before they can work flawlessly.

S. Ananthi (2009) has conducted a research on "Spam Filtering using K-NN". In this paper, she used KNN as it is one of the simplest algorithm. An object is classified by a majority vote of its neighbours where the class is typically small.

"A Survey on Spam Detection Techniques" was done by Anjali Sharma et al. in 2014. In this paper, it was discovered that in order to classify emails as spam or not starting from the specific data sets, an Artificial Neural Network (ANN) must first be trained.

Simon Tong and Daphne Koller (2001) has conducted a research on "Support Vector Machine Active Learning with Applications to Text Classification". They introduced a new active learning technique in this research that uses SVM induction and transduction. At each query, it is used to minimise version space as much as possible. They discovered that the extant dataset and the original tagged data set only varied by one instance.

#### **1.4 Motivation for the Problem Undertaken**

The project is provided to me by Flip Robo Technologies as a part of the internship programme (Internship Batch No-31). This problem is a real world dataset. The exposure of this data gives me the opportunity to locate my skills in solving a real time problem. It is the primary motivation to solve this problem.

We have a lot of possibilities here, but not as many specific solutions. The primary aim is to create a prototype for an online hate and abuse comment classifier that can be used to categorise hateful and offensive comments in order to limit their ability to spread intolerance and cyberbullying.

This study aims to analyse and predicting malignant comment when using **Classification Model** like Logistic Regression, Random Forest, Decision Tree, Gradient Boosting, Extra Tree, Ada Boost Classification algorithms. Thus, the purpose of this study is to grow the knowledge of **Classification methods** in machine learning fields. Those are the different factors to undertaken the problem for study purpose.

## 2. Analytical Problem Framing

### 2.1 Mathematical/ Analytical Modelling of the Problem

The goal of this project is to predict spam message from SMS in social media. We would use a classification method, which is a sort of supervised learning. We will only do classification in this case. Filtering the words is necessary to avoid overfitting because the dataset only contains one feature.

This project have one big set of data one for both training and testing. Throughout the project's classification phase, we would first eliminate all numbers, words, spaces and other terms with stops, in order to calculate the regularisation parameter. We also used Count Vectorizer to transform the tokens into vectors so that the machine could carry out additional processing in order to further enhance our models.

Here 4 different algorithm are used and final model is chosen by best AUC-ROC score and accuracy score.

### 2.2 Data Sources and their formats

There are one set of data. The dataset has 5572 rows and 2 columns. Primarily it has 5 columns where 3 columns are unnecessary. After deleting those 3 columns, finally this dataset has 2 columns named v1 and v2. Later rename the columns as type and message\_body.

Later the dataset is divided into two parts, training and testing. After determine the proper model, the model is applied to predict the target variable for the test dataset.

```
data = data.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
data = data.rename(columns={"v1": "type", "v2": "message_body"})
```

```
print('No. of Rows of test dataset :', data.shape[0])
print('No. of Columns of test dataset :', data.shape[1])
```

```
No. of Rows of test dataset : 5572
No. of Columns of test dataset : 2
```

### 2.3 Data Pre-processing Done:

#### 2.3.1 Feature Engineering:

'--', 'null', 'NA', ' ' are not present in the dataset.

```
data.isin(['--','null','NA','']).sum().any()
```

```
False
```

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

```
type          0
message_body  0
dtype: int64
```

### 2.3.2 Drop unnecessary columns:

Drop the unnecessary column Unnamed: 2, Unnamed: 3, Unnamed: 4 from the dataset.

```
data = data.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
data = data.rename(columns={"v1": "type", "v2": "message_body"})
```

### 2.3.3 Calculate length before cleaning of 'message\_body' column:

Let's calculate the comment length before cleaning.

```
data['length_before_cleaning'] = data['message_body'].str.len()
data.head()
```

	type	message_body	length_before_cleaning
0	0	Go until jurong point, crazy.. Available only ...	111
1	0	Ok lar... Joking wif u oni...	29
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	0	U dun say so early hor... U c already then say...	49
4	0	Nah I don't think he goes to usf, he lives aro...	61

### 2.3.4 Encoding type dataset:

Here 'type' dataset is object datatype. Let's encoded the type columns as ham=0 and spam=1 for further progress.

```
data.loc[:, 'type'] = data.type.map({'ham':0, 'spam':1})
data.head()
```

	type	message_body
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

### 2.3.5 Length of message\_body:

Here the maximum length of the message is 910 where minimum length is 2. The median of the comment length is 61.

```
data['length_before_cleaning'].min()
```

2

```
data['length_before_cleaning'].max()
```

910

```
data['length_before_cleaning'].median()
```

61.0

### 2.3.6 Natural Language Processing:

Import all necessary libraries for NLP. Now let's remove all alphabates, numbers from the text body. Then apply Stemmer, Stop words and such necessary NLP steps on message body for cleaning the dataset.

```
allmessage=[] # initialize empty list for adding all text messages
for i in range(0, 5572):
    ps=PorterStemmer()
    message=data['message_body'][i]
    message=re.sub('[^a-zA-Z]', ' ', message)
    message=re.sub(r'\d+', '', message)
    message=message.lower()
    message=message.split()
    message=[ps.stem(word) for word in message if not word in set(stopwords.words('english')+ ['u','ur','4','2','im',
    'dont','doin','I','numbr','YOU','You','A','If'])]
    message=' '.join(message)
    allmessage.append(message)
```

We also use Count Vectorizer method on the features and label encoding on target variable.



## **2.4 State the set of assumptions (if any) related to the problem under consideration**

No such assumptions are taken for this case.

## **2.5 Hardware and Software Requirements and Tools Used**

Processor: Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 2.00 GHz

RAM: 4.00 GB

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

Window: Windows 10 Pro

Anaconda – Jupyter Notebook

Libraries Used –

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
```

For Word-Cloud the following libraries are used.

```
#Importing Required Libraries
import nltk
import re
import string
from nltk.corpus import stopwords
from wordcloud import WordCloud
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

Except this, different libraries are used for machine learning model building from sklearn.

### 3. Model/s Development and Evaluation

#### 3.1 Identification of possible problem-solving approaches (methods):

In this problem Classification-based machine learning algorithm like logistic regression can be used. Removed any excess spaces, changed the email addresses subject line to a phone number that is probably wise, etc. For building an appropriate ML model before implementing classification algorithms, data is split in training & test data using `train_test_split`. Then different statistical parameter like accuracy score, confusion matrix, classification report, precision, recall etc. are determined for every algorithm. Hyper parameter tuning is performed to get the accuracy score much higher and accurate than earlier.

Then the best model is chosen from 4 different algorithm.

#### 3.2 Testing of Identified Approaches (Algorithms)

Total 7 algorithms used for the training and testing are:

1. Logistic Regression
2. Decision Tree Classifier
3. Gradient Boosting Classifier
4. Multinomial Naive Bayes

#### 3.3 Key Metrics for success in solving problem under consideration:

From metrics module of sklearn library import `classification_report`, `accuracy_score`, `confusion_matrix`, `classification_report` and `f1_score`. From model\_selection also, we use `cross_val_score`. Those are the matrices use to validate the model's quality. Let's discuss every metrics shortly.

- Classification report: It is a performance evaluation metric in machine learning which is used to show the precision, recall, F1 Score, and support score of your trained classification model
- Accuracy score: It is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.
- Confusion Matrix: It is a table that is used in classification problems to assess where errors in the model were made. The rows represent the actual classes the outcomes should have been. While the columns represent the predictions we have made. Using this table it is easy to see which predictions are wrong.
- Precision: It can be seen as a measure of quality. If the precision is high, an algorithm returns more relevant results than irrelevant ones.
- Recall: The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples.
- F1 Score:  $F1 = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$

### 3.4 Run and Evaluate selected models

First find the best random state of train\_test\_split to get best accuracy. Here the random state is 8. Then after splitting the data into 4 different part and check the shape of the data.

```
print('Training feature shape:',x_train.shape)
print('Training target shape:',y_train.shape)
print('Test feature shape:',x_test.shape)
print('Test target shape:',y_test.shape)
```

```
Training feature shape: (3900, 6221)
Training target shape: (3900,)
Test feature shape: (1672, 6221)
Test target shape: (1672,)
```

#### A Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3, random_state= 8)
log = LogisticRegression()
log.fit(x_train, y_train)
y_pred = log.predict(x_test)

print('accu score : ', accuracy_score(y_test, y_pred))
print('cof_mat:\n ', confusion_matrix(y_test, y_pred))
print('classification report:\n ', classification_report(y_test, y_pred))
print("-----")
print("-----")
print('training score : ', log.score(x_train, y_train))
print('testing score : ', log.score(x_test, y_test))
```

```
accu score : 0.9856459330143541
cof_mat:
[[1447   3]
 [  21 201]]
classification report:
              precision    recall  f1-score   support

      0       0.99      1.00      0.99      1450
      1       0.99      0.91      0.94       222

   accuracy          0.99
  macro avg       0.99      0.95      0.97
weighted avg       0.99      0.99      0.99

-----
-----
training score : 0.9953846153846154
testing score : 0.9856459330143541
```

In this way accuracy score is determined for each 4 different classification model.

### **B Decision Tree Classifier:**

The accuracy score, confusion matrix and classification report after using Decision tree Classifier is as follows.

```
accu score : 0.9700956937799043
cof_mat: [[1425  25]
 [ 25 197]]
classification report:
              precision    recall  f1-score   support

     0           0.98       0.98       0.98       1450
     1           0.89       0.89       0.89        222

 accuracy          0.97       0.97       0.97       1672
 macro avg          0.94       0.94       0.94       1672
weighted avg          0.97       0.97       0.97       1672

-----
training score : 1.0
testing score : 0.9700956937799043
```

### **C Gradient Boosting Classifier:**

The accuracy score, confusion matrix and classification report after using Gradient Boosting Classifier is as follows.

```
accu score : 0.9688995215311005
cof_mat: [[1447   3]
 [ 49 173]]
classification report:
              precision    recall  f1-score   support

     0           0.97       1.00       0.98       1450
     1           0.98       0.78       0.87        222

 accuracy          0.97       0.97       0.97       1672
 macro avg          0.98       0.89       0.93       1672
weighted avg          0.97       0.97       0.97       1672

-----
training score : 0.9784615384615385
testing score : 0.9688995215311005
```

### **D Multinomial Naïve Bayes Classifier:**

The accuracy score, confusion matrix and classification report after using Multinomial Naïve Bayes Classifier is as follows.

```

accu score : 0.9850478468899522
cof_mat: [[1433  17]
          [  8 214]]

classification report:
              precision    recall  f1-score   support

     0           0.99       0.99       0.99       1450
     1           0.93       0.96       0.94        222

 accuracy          0.99          1672
 macro avg         0.96          1672
 weighted avg      0.99          1672
-----
training score : 0.9915384615384616
testing score  : 0.9850478468899522

```

As per 4 different model, Naïve Bayes is the best model.

But for Logistic Regression the difference between training and testing is very small as well as it also gives good accuracy.

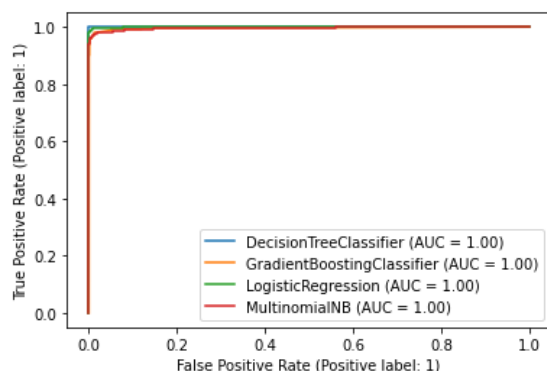
### 3.5 AUC- ROC Curve:

```

from sklearn.metrics import plot_roc_curve

disp = plot_roc_curve(clf, x_train, y_train)
plot_roc_curve(gbdt, x_train, y_train, ax=disp.ax_)
plot_roc_curve(log, x_train, y_train, ax=disp.ax_)
plot_roc_curve(naive, x_train, y_train, ax=disp.ax_)
plt.show()

```

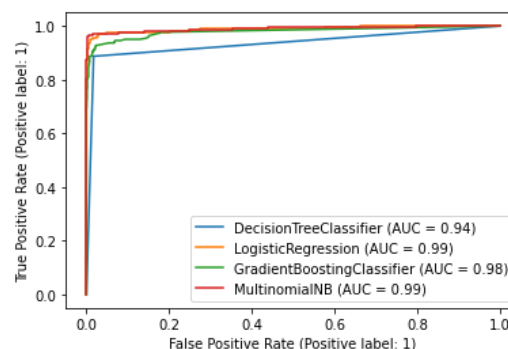


```

from sklearn.metrics import plot_roc_curve

disp = plot_roc_curve(clf, x_test, y_test)
plot_roc_curve(log, x_test, y_test, ax=disp.ax_)
plot_roc_curve(gbdt, x_test, y_test, ax=disp.ax_)
plot_roc_curve(naive, x_test, y_test, ax=disp.ax_)
plt.show()

```



### 3.6 Hyper Parameter Tuning:

Here for Naive bayes, Logistic Regression the AUC score is same. Here we take Naive bayes for final model. So it is the final model for this dataset.

```
from sklearn.model_selection import GridSearchCV
grid = dict(alpha=[1, 0.5,0.0001,0.001],fit_prior=[True, False])

grid_naive = GridSearchCV(estimator=naive, param_grid= grid,refit = True)

grid_naive.fit(x_train, y_train)
print('best params : ', grid_naive.best_params_)

best params :  {'alpha': 1, 'fit_prior': True}
```

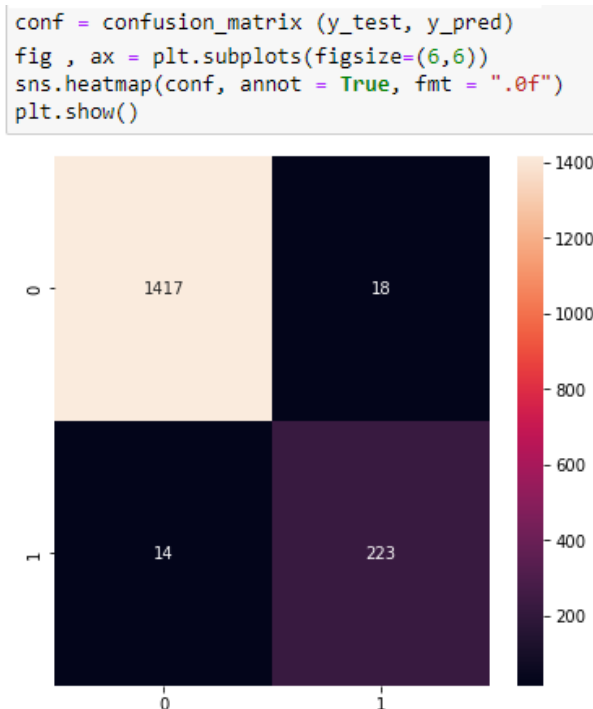
Best params: {'alpha': 1, 'fit\_prior': 'True'}

Here accuracy score is not improved after using hyper parameter tuning.

### 3.7 Final Model:

For final model the target variable is as follows after using Naïve Bayes.

### 3.8 Confusion Matrix:



### 3.9 Load the model:

Let's save the model using pickle for future use. Then see the actual and predicted value of 6 random sample.

```
import pickle
pickle.dump(grid_naive_best, open("Spam_SMS_Classification_model", "wb"))
load_Spam_SMS_Classification_model= pickle.load(open("Spam_SMS_Classification_model", "rb"))

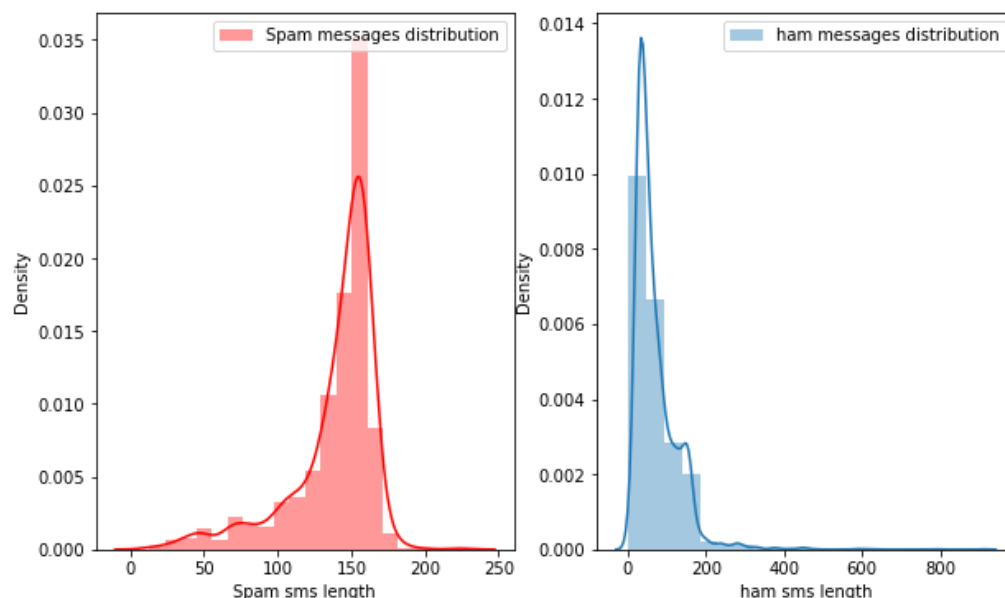
y_pred = load_Spam_SMS_Classification_model.predict(x_test)

y_test = np.array(y_test)
data_prediction_by_model = pd.DataFrame()
data_prediction_by_model["Predicted Values"] = y_pred
data_prediction_by_model["Actual Values"] = y_test
data_prediction_by_model.sample(n=6)
```

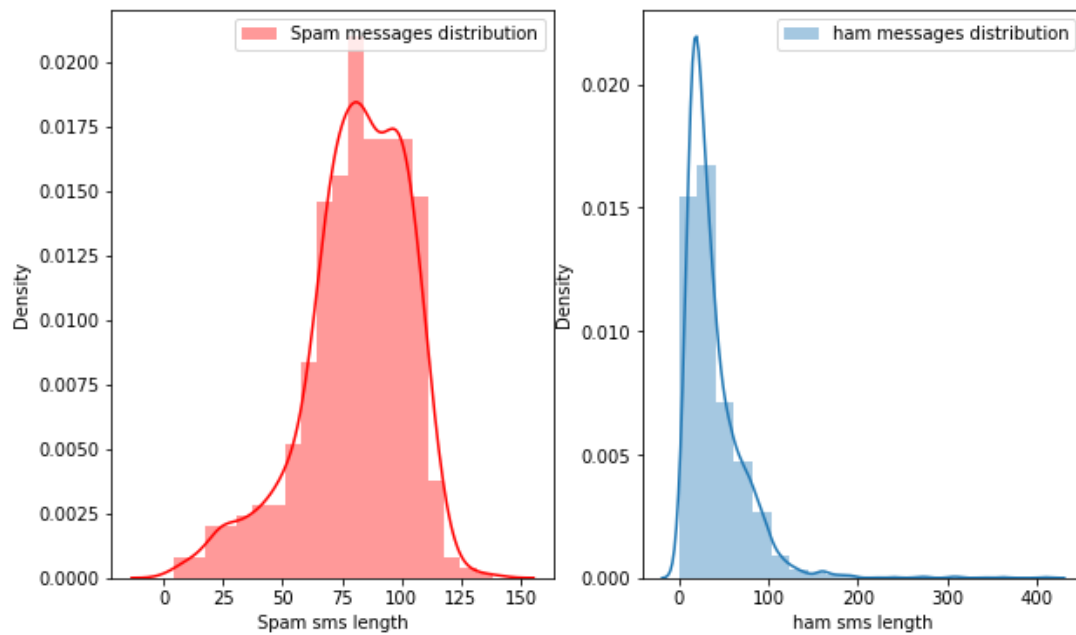
	Predicted Values	Actual Values
1070	0	1
27	0	0
522	1	1
193	0	0
1621	0	0
474	1	1

### 3.10 Visualizations:

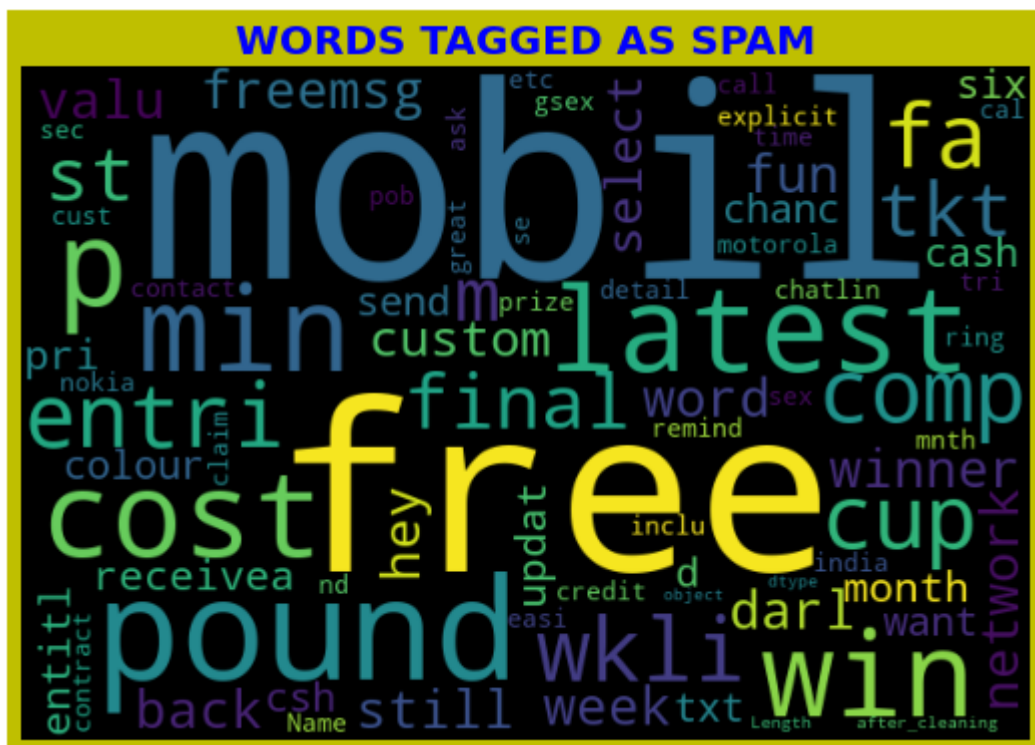
Let's start the observation exploration of feature analysis. Visualize the dataset before cleaning for spam and not spam messages.



Let's visualize it after cleaning the dataset.



### 3.11 Word Cloud for Spam sms:







### 3.12 Interpretation of the Results

After all the pre-processing steps, the dataset is ready to train machine learning models. All unnecessary words from comment text are deleted as they might give overfitting problem as well as it also could increase the time complexity. Now apply this dataset on different ML Classification Model (as discussed on part 3.4 - ‘Run and Evaluate selected models’) and check the best model for this particular dataset.

## **4. CONCLUSION**

### **4.1 Key Findings and Conclusions of the Study**

Here, we observed the various spam SMS. We have also seen how easily accessible algorithms can be used in this way to handle this difficulty. It was shown in our particular investigation that a logistic regression solution offers a significant improvement in classification compared to any other approach.

### **4.2 Learning Outcomes of the Study in respect of Data Science**

Data cleansing is one of the most crucial phases; I attempted to make comments shorter and included all the relevant keywords in it. The power of visualisation is beneficial for converting data into a graphical representation; it helps me to comprehend what the data is trying to communicate.

In this dataset, I utilised a variety of methods to find the best result and preserve that model. The use of NLP method is very helpful to determine the Spam message.

### **4.3 Limitations of this work and Scope for Future Work**

Additionally, the following studies are examples that might be taken into account for future work in this field:

- We offer the following strategy to enhance NLP classifiers: Convolutional neural networks (CNN) and Support vector clustering (SVC) are two additional algorithms that can be used to enhance the performance of existing classifiers. In the present study, the issue was reduced to two classes, although it is worthwhile to pursue the primary objective of six classes of remarks.
- For text processing and classification, we also advocate the use of better NLP methods.