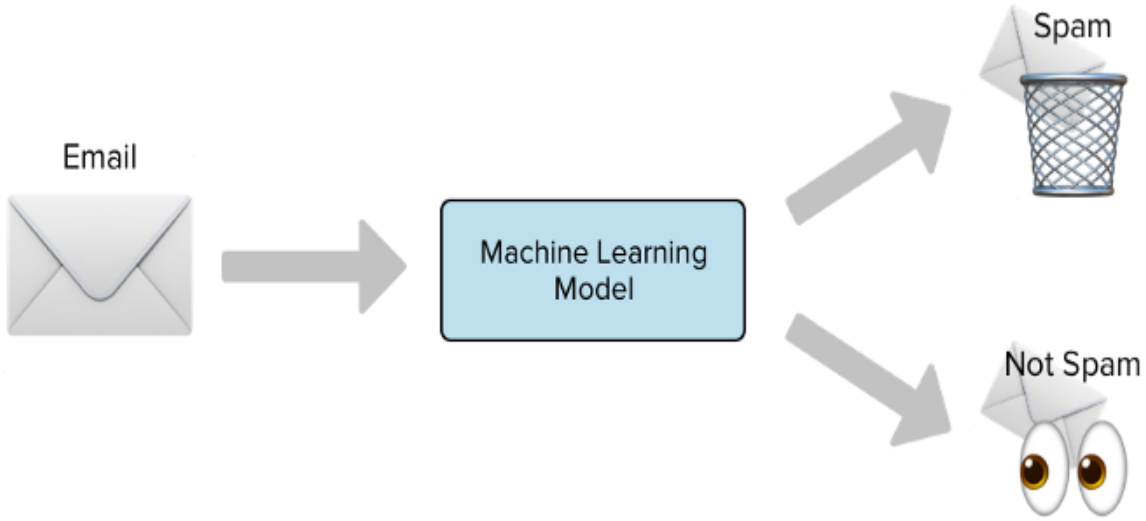
FLIP ROBO

# SPAM DETECTION CLASSIFIER PROJECT




**By:**

Arpan Pattanayak

Internship-29

# ACKNOWLEDGMENT

I would like to convey my heartfelt gratitude to Flip Robo Technologies for providing me with this wonderful opportunity to work on a Machine Learning project using NLP "Spam Detection Classifier Project" and also want to thank my SME "Shwetank Mishra" for providing the dataset and directions to complete this project. This project would not have been accomplished without their help and insights.

I would also like to thank my academic "Data Trained Education" and their team who has helped me to learn Machine Learning and NLP.

Working on this project was an incredible experience as I learnt more from this Project during completion.

# INTRODUCTION

## 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.

A collection of 5573 rows 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.

## 2. Conceptual Background of the Domain Problem

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.

## 3. Review of Literature

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.

## 4. Motivation for the Problem Undertaken

To build an application which can detect the spam by seeing the review.

# Analytical Problem Framing

## 1. Mathematical/ Analytical Modelling of the Problem

1) Cleaned Data by removing irrelevant features
2) Pre-processing of text using NLP processing
3) Used Word Counts
4) Used Character Counts
5) Used Count Vectorizer
6) Splitted data into train and test
7) Built Model
8) Hyper parameter tunning

## 2. Data Sources and their formats

The data-set is in csv format: **spam.csv**. Features of this dataset are:

- v1- target column
- v2- containing messages
- Unnamed: 2- Containing Null Values
- Unnamed: 3- Containing Null Values
- Unnamed: 4- Containing Null Values

## 3. Data Pre-processing:

- Checked 5 Rows of Dataset

```
df.sample(5)
```

| | v1 | v2 | Unnamed: 2 | Unnamed: 3 | Unnamed: 4 |
|---|---|---|---|---|---|
| **2546** | ham | Company is very good.environment is terrific a... | NaN | NaN | NaN |
| **3557** | ham | No da. . Vijay going to talk in jaya tv | NaN | NaN | NaN |
| **4482** | ham | True lov n care wil nevr go unrecognized. thou... | NaN | NaN | NaN |
| **5309** | ham | Jolly good! By the way, will give u tickets f... | NaN | NaN | NaN |
| **3553** | ham | am up to my eyes in philosophy | NaN | NaN | NaN |

- Checked Total Numbers of Rows and Column

```
df.shape
```

```
(5572, 5)
```

- Checked Data Type of All Data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

# Data cleaning

- Dropped Column " Unnamed: 2, Unnamed: 3, Unnamed: 4 " as this column contains Null Values.

```
# drop last 3 cols
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

```
df.sample(5)
```

| | v1 | v2 |
|---|---|---|
| **254** | ham | I'm back, lemme know when you're ready |
| **2526** | ham | Do u noe how 2 send files between 2 computers? |
| **3463** | ham | My phone |
| **2218** | ham | Nice talking to you! please dont forget my pix... |
| **3471** | ham | I think I□Û÷m waiting for the same bus! Inform... |

- Checked and Dropped Duplicates Values.

```
# check for duplicate values
df.duplicated().sum()
```

```
403
```

```
#describing data for spam messages
spam_data[spam_data['target'] == 1][['length','num_words','num_sent']].describe()
```

| | length | num_words | num_sent |
|---|---|---|---|
| count | 653.000000 | 653.000000 | 653.000000 |
| mean | 137.891271 | 27.667688 | 2.969372 |
| std | 30.137753 | 7.008418 | 1.488910 |
| min | 13.000000 | 2.000000 | 1.000000 |
| 25% | 132.000000 | 25.000000 | 2.000000 |
| 50% | 149.000000 | 29.000000 | 3.000000 |
| 75% | 157.000000 | 32.000000 | 4.000000 |
| max | 224.000000 | 46.000000 | 9.000000 |

# 4. Data Inputs- Logic- Output Relationships

## I. Text Pre-Processing

```
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        if i not in stopwords.words('english'):
            y.append(i)

    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))


    return " ".join(y)
```

```
transform_text("I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.")
```

```
'gon na home soon want talk stuff anymor tonight k cri enough today'
```

```
df['text'][10]
```

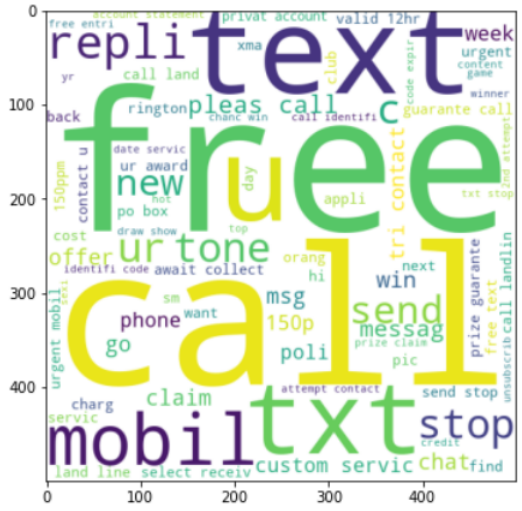
```
"I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today."
```

```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
```

```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

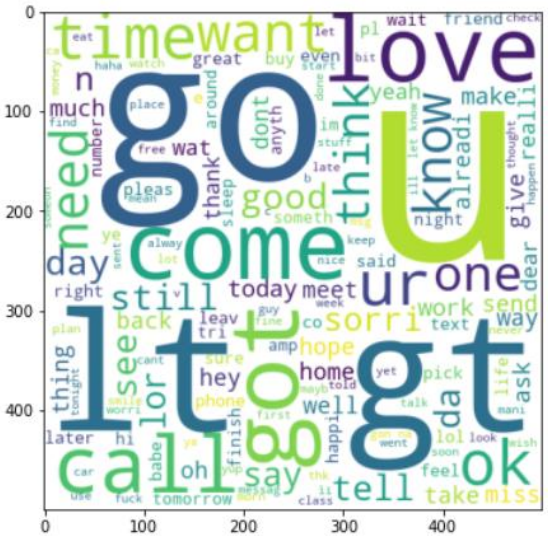
```
<matplotlib.image.AxesImage at 0x11281f4dfd0>
```



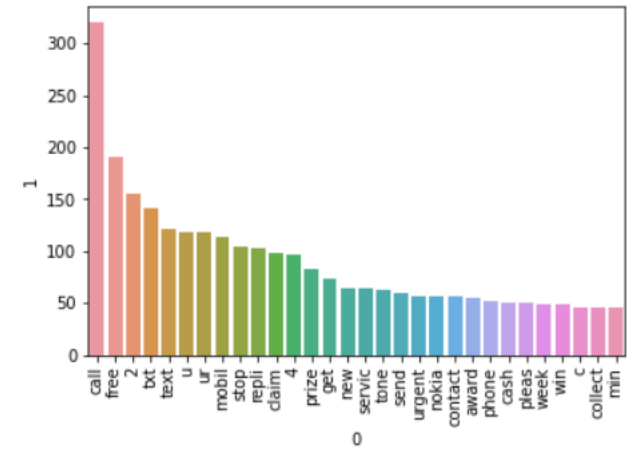

```
ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))
```

```
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

```
<matplotlib.image.AxesImage at 0x112820ddd60>
```



```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(30))[0],pd.DataFrame(Counter(spam_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
```




---

## 5. State the set of assumptions (if any) related to the problem under consideration

- It was observed that there are two types of messages: ham and spam. So, have to detect which message is spam and this column is target column. And also have to rename column names.

---

- First column contains the type.
- Second column contains text which means these are messages and have detect these messages.
- Rest three columns contains Null Values, so, it is not relevant and have to be dropped.
- It was observed that in message column there are irrelevant values. So, we need to replace or pre-process those values.
- Also have to convert text (reviews) into vectors using counter-vectorize.
- By looking into the Target Variable, it is assumed that it is a classification problem.

## 6. **Hardware and Software Requirements and Tools Used**

- **Hardware used:**
  - **Processor**: Processor: core i5 RAM: 12 GB ROM/SSD: 512 GB
  - **System Type**: 64-bit OS

- **Software used:**
  - **Anaconda** for 64-bit OS
  - **Jupyter** notebook

- **Tools, Libraries and Packages used:**

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

```
from sklearn.model_selection import train_test_split, GridSearchCV,cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
import lightgbm
from sklearn.svm import LinearSVC
from sklearn.linear_model import SGDClassifier
from xgboost import XGBClassifier
import scikitplot as skplt

import nltk
import re
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
import gensim
from gensim.models import Word2Vec
from sklearn.feature_extraction.text import TfidfVectorizer
from wordcloud import WordCloud
from sklearn.feature_extraction.text import CountVectorizer

import joblib
```

# Model/s Development and Evaluation

## 1. Identification of possible problem-solving approaches (methods)

In this project, we want to differentiate between comments and its categories and for this we have used these approaches:

- Checked Total Numbers of Rows and Column
- Checked All Column Name
- Checked Data Type of All Data
- Checked for Null Values
- Checked total number of unique values
- Description of Data
- Dropped irrelevant Columns
- Replaced special characters and irrelevant data
- Checked all features through visualization.
- Removed unwanted punctuations and special characters
- Converted all messages to lower case
- Removed punctuations
- Removed StopWords
- Used Counter-Vectorization
- Used Word Counts

- Used Character Counts
- Checked loud word using WordCloud
- Converted text into vectors using Counter-Vectorize

## 2. **Run and evaluate selected models**

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

```
X = tfidf.fit_transform(df['transformed_text']).toarray()
```

```
X.shape
```

```
(5169, 3000)
```

```
y = df['target'].values
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

```
gnb.fit(X_train,y_train)
y_pred1 = gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

```
0.8694390715667312
[[788 108]
 [ 27 111]]
0.5068493150684932
```

```
mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))
```

```
0.9709864603481625
[[896   0]
 [ 30 108]]
1.0
```

```
bnb.fit(X_train,y_train)
y_pred3 = bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))
```

```
0.9835589941972921
[[895   1]
 [ 16 122]]
0.991869918699187
```

```
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)
xgb = XGBClassifier(n_estimators=50,random_state=2)
```

```
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
```

```
def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)

    return accuracy,precision
```

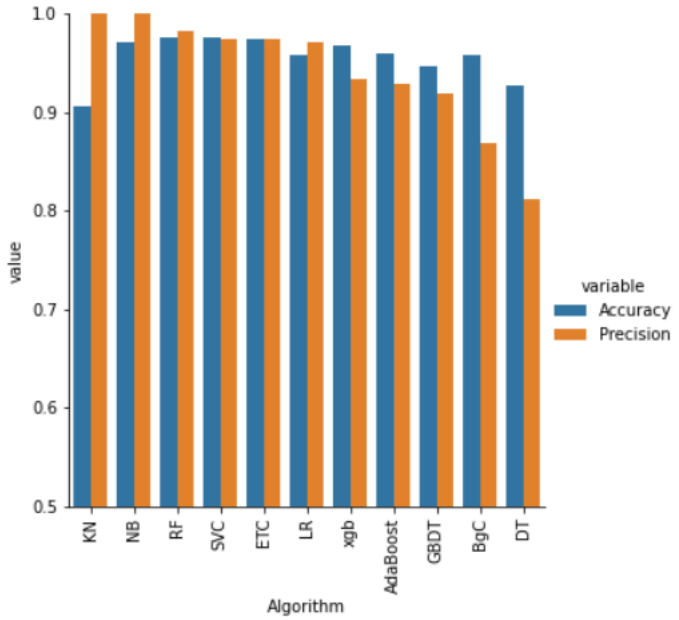
```
train_classifier(svc,X_train,y_train,X_test,y_test)
```

```
(0.9758220502901354, 0.9747899159663865)
```

```
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf, X_train,y_train,X_test,y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

```
For  SVC
Accuracy -  0.9758220502901354
Precision -  0.9747899159663865
For  KN
Accuracy -  0.9052224371373307
Precision -  1.0
For  NB
Accuracy -  0.9709864603481625
Precision -  1.0
For  DT
Accuracy -  0.9274661508704062
Precision -  0.8118811881188119
For  LR
Accuracy -  0.9584139264990329
Precision -  0.9702970297029703
For  RF
Accuracy -  0.9758220502901354
Precision -  0.9829059829059829
For  AdaBoost
Accuracy -  0.960348162475822
Precision -  0.9292035398230089
For  BgC
Accuracy -  0.9584139264990329
Precision -  0.8682170542635659
For  ETC
Accuracy -  0.9748549323017408
Precision -  0.9745762711864406
For  GBDT
Accuracy -  0.9468085106382979
Precision -  0.9191919191919192
For  xgb
Accuracy -  0.9671179883945842
Precision -  0.9333333333333333
```

```
sns.catplot(x = 'Algorithm', y='value',
            hue = 'variable',data=performance_df1, kind='bar',height=5)
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
```




```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':accuracy_scores,'Precision_num_chars':precision_scores}).sor
```

```
new_df_scaled.merge(temp_df,on='Algorithm')
```

| | Algorithm | Accuracy | Precision | Accuracy_scaling_x | Precision_scaling_x | Accuracy_scaling_y | Precision_scaling_y | Accuracy_num_chars | Precision_num_chars |
|---|---|---|---|---|---|---|---|---|---|
| 0 | KN | 0.905222 | 1.000000 | 0.905222 | 1.000000 | 0.905222 | 1.000000 | 0.905222 | 1.000000 |
| 1 | NB | 0.970986 | 1.000000 | 0.970986 | 1.000000 | 0.970986 | 1.000000 | 0.970986 | 1.000000 |
| 2 | RF | 0.975822 | 0.982906 | 0.975822 | 0.982906 | 0.975822 | 0.982906 | 0.975822 | 0.982906 |
| 3 | SVC | 0.975822 | 0.974790 | 0.975822 | 0.974790 | 0.975822 | 0.974790 | 0.975822 | 0.974790 |
| 4 | ETC | 0.974855 | 0.974576 | 0.974855 | 0.974576 | 0.974855 | 0.974576 | 0.974855 | 0.974576 |
| 5 | LR | 0.958414 | 0.970297 | 0.958414 | 0.970297 | 0.958414 | 0.970297 | 0.958414 | 0.970297 |
| 6 | xgb | 0.967118 | 0.933333 | 0.967118 | 0.933333 | 0.967118 | 0.933333 | 0.967118 | 0.933333 |
| 7 | AdaBoost | 0.960348 | 0.929204 | 0.960348 | 0.929204 | 0.960348 | 0.929204 | 0.960348 | 0.929204 |
| 8 | GBDT | 0.946809 | 0.919192 | 0.946809 | 0.919192 | 0.946809 | 0.919192 | 0.946809 | 0.919192 |
| 9 | BgC | 0.958414 | 0.868217 | 0.958414 | 0.868217 | 0.958414 | 0.868217 | 0.958414 | 0.868217 |
| 10 | DT | 0.927466 | 0.811881 | 0.927466 | 0.811881 | 0.927466 | 0.811881 | 0.927466 | 0.811881 |

```
# Voting Classifier
svc = SVC(kernel='sigmoid', gamma=1.0,probability=True)
mnb = MultinomialNB()
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)

from sklearn.ensemble import VotingClassifier
```

```
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)],voting='soft')
```

```
voting.fit(X_train,y_train)
```

```
VotingClassifier(estimators=[('svm',
                              SVC(gamma=1.0, kernel='sigmoid',
                                  probability=True)),
                             ('nb', MultinomialNB()),
                             ('et',
                              ExtraTreesClassifier(n_estimators=50,
                                                   random_state=2))],
                 voting='soft')
```

```
y_pred = voting.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.9816247582205029
Precision 0.9917355371900827
```

```
# Applying stacking
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator=RandomForestClassifier()
```

```
from sklearn.ensemble import StackingClassifier
```

```
clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
```

```
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
Accuracy 0.9816247582205029
Precision 0.9612403100775194
```

# Saving The Model

```
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model.pkl','wb'))
```

## 3. Key Metrics for success in solving problem under consideration

- Accuracy Score, Precision Score, Recall Score, F1-Score and CV score are used for success. Also, confusion matrix is used for success.

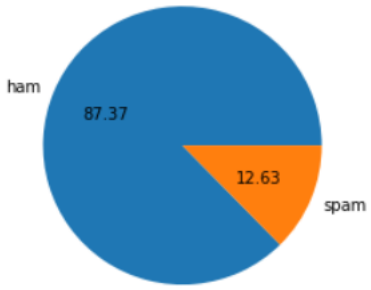
## 4. Visualization

Pie-Plot

```
df['target'].value_counts()
```

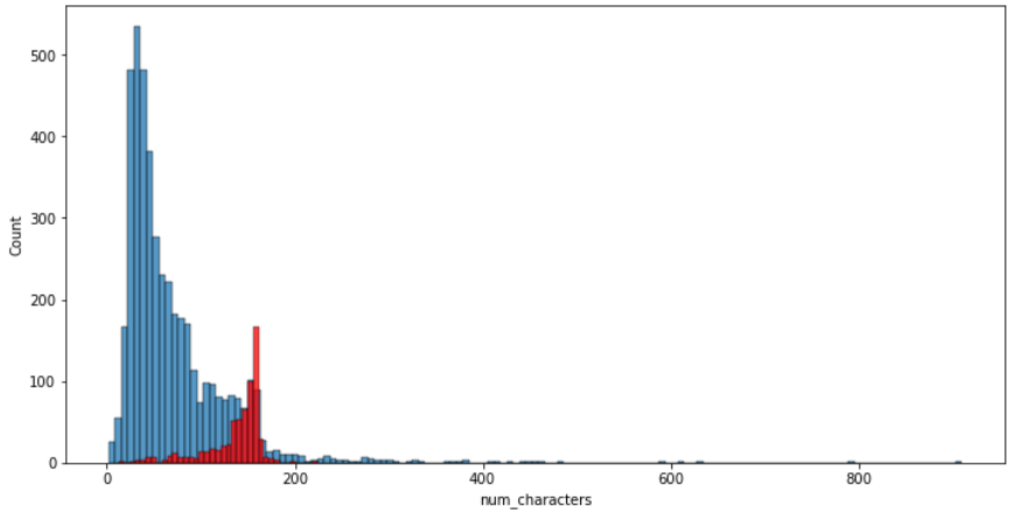
```
0    4516
1     653
Name: target, dtype: int64
```

```
import matplotlib.pyplot as plt
plt.pie(df['target'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
plt.show()
```



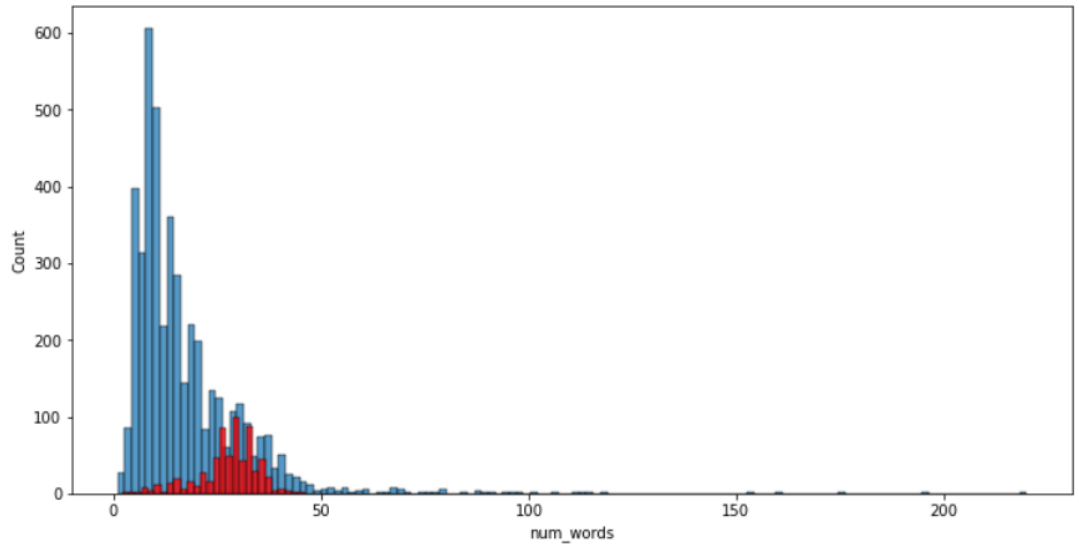
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

```
<AxesSubplot:xlabel='num_characters', ylabel='Count'>
```




```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

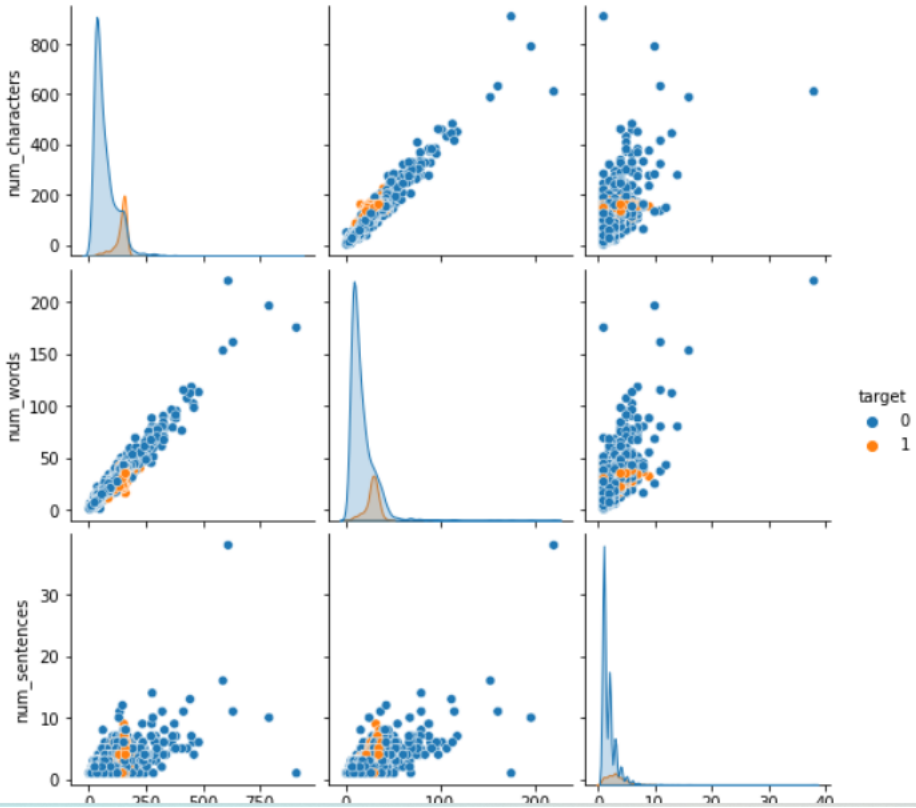
```
<AxesSubplot:xlabel='num_words', ylabel='Count'>
```




These histogram are for the number of characters and words in ham and spam messages and we can clearly see that spam messages have large number of charcters and words than ham messages.

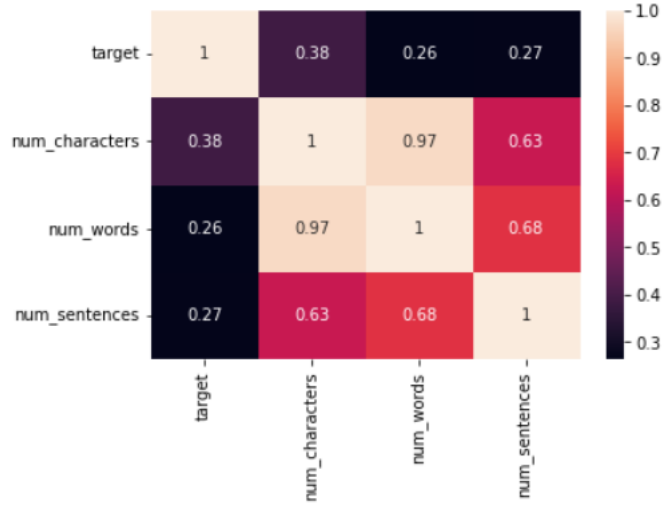
```
sns.pairplot(df,hue='target')
```

```
<seaborn.axisgrid.PairGrid at 0x11280033f70>
```




```
sns.heatmap(df.corr(),annot=True)
```

```
<AxesSubplot:>
```



## 5. Interpretation of the Results

- Through Pre-processing it is interpretated that all texts are converted to lower case, removed Punctation, replaced extra space, removed stop-words, Calculated length of sentence, words, and characters, converted text using Counter-Vectorize.
- Natural Language Processing and Machine Learning is used in this project.

# CONCLUSION

## Key Findings and Conclusions of the Study

In this project we have detected spam and ham messages that have been collected for SMS Spam research. Then we have done different text process to eliminate problem of imbalance. By doing different EDA steps we have analyzed the text.

We have checked frequently occurring words in our data as well as rarely occurring words. After all these steps we have built function to train and test different algorithms and using various evaluation metrics we have selected Linear-SVC for our final model.

Finally, by doing hyperparameter tuning we got optimum parameters for our final model. And finally, we got improved accuracy score for our final model.

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

- This project has demonstrated the importance of NLP.
- Through different powerful tools of visualization, we were able to analyse and interpret the huge data and with the help of pie plot,

count plot & word cloud, I can see the distribution of spam and ham messages.

- Through data cleaning we were able to remove unnecessary columns, values, stop-words and punctuation from our dataset due to which our model would have suffered from overfitting or underfitting.

**The few challenges while working on this project were: -**

- Using NLP to find punctuations & stop words, it took time in giving the result.
- The data set took time to run some algorithms & to check the cross-validation score.

## Limitations of this work and Scope for Future Work

As we know there are two types of messages to. So, it is difficult to detect with higher accuracies. Still, we can improve our accuracy by fetching more data and by doing extensive hyperparameter tuning.