SPAM DETECTION CLASSIFIER PROJECT

Prepared by:

Vikash Kumar Singh

SME Name:

Gulshana Chaudhary

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 “Spam Detection Classifier Project” and also want to Thank my SME, **Gulshana Chaudhary** for providing the dataset and guiding me 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.

I also references from some websites which are- https://www.youtube.com https://www.kaggle.com ,
https://www.github.com , https://stackoverflow.com

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 no spam texts. It uses a binary type of classification containing the labels such as 'ham' (no spam) 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

As we all know, in today's digital world spamming is becoming very

common and unavoidable. Basically, Email spams are the junk file attached infected malware. It is used by Samper to get any one's personal information and bank details to harm him/her mentally and financially. It's is very irritation. In this project I might get any information to How can we stop it.

Analytical Problem Framing

1. Mathematical/ Analytical Modeling of the Problem

1) Used Panda's Library to save data into csv file
2) Cleaned Data by removing irrelevant features
3) Descriptive Statistics
4) Analyzed correlation
5) Pre-processing of text using NLP processing
6) Used Word Counts Removed Punctuation
7) Replaced extra space
8) Used Character CountUsed Character
9) Added and removed stop words
10) Calculated length of sentence
11) Checked the word which are spam message
12) Checked the word which are ham message
13) Converted text into vectors using TF-IDF

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:

a) Checked Top 5 Rows of both Dataset and Checked Total Numbers of Rows and Column

top 5 rows data

```
df.head()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

EDA

Total Numbers of Rows and Column

```
df.shape
```

```
(5572, 5)
```

b) Checking the relevant info about the dataset

```
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
```

c) Columns name of the dataset

Column Name

```
df.columns
```

```
Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')
```

d) Checked for Null Values

Now, checking for NULL values

```
df.isnull().sum()
```

```
v1              0
v2              0
Unnamed: 2    5522
Unnamed: 3    5560
Unnamed: 4    5566
dtype: int64
```

We will remove the null values data and Unnamed :2, Unnamed :3, Unnamed :4 columns are almost all the data having null values.

droping irrelevant columns

```
df1=df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'])
```

e) Changing the columns name.

changing the columns name as target and message

```
df2=df1.rename(columns={'v1' : 'target','v2' : 'message'})
df2
```

38]:

	target	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...

f) Checking unique values and value counts.

```
df2['target'].nunique()
```

9]: 2

```
df2['target'].unique()
```

9]: array(['ham', 'spam'], dtype=object)

```
df2['target'].value_counts()
```

1]:
```
ham     4825
spam     747
Name: target, dtype: int64
```

g) Information about Data

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5169 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   target   5169 non-null   object
 1   message  5169 non-null   object
dtypes: object(2)
memory usage: 121.1+ KB
```

h) Describing the data

```
df2.describe()
```

	target	message
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

i) Checking for duplicate data

```
df2.duplicated().sum()
```

```
403
```

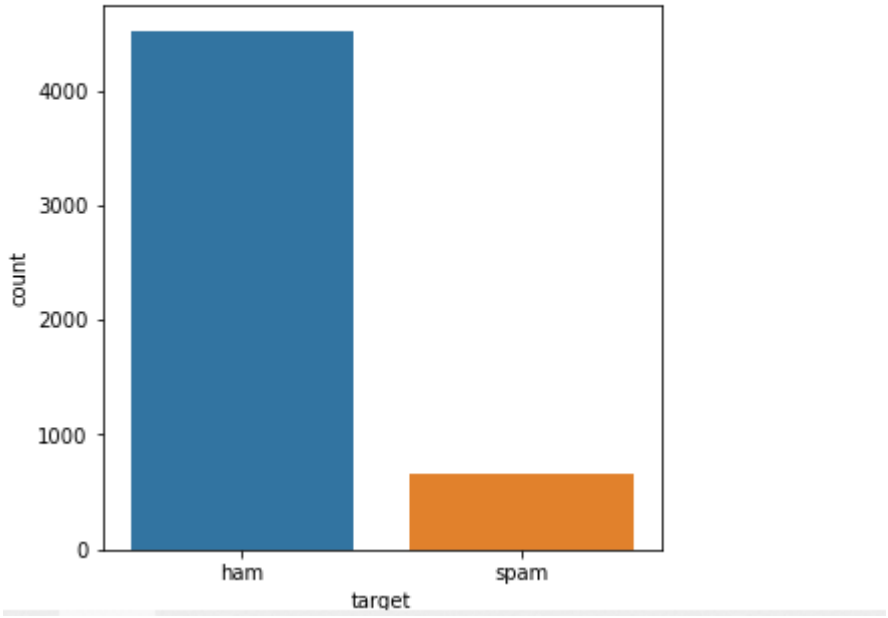
Dropping off the duplicate data

```
df2.drop_duplicates(inplace = True)
```

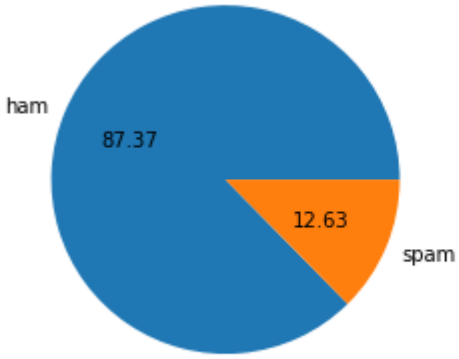
j) Data Visualization

- Count plot

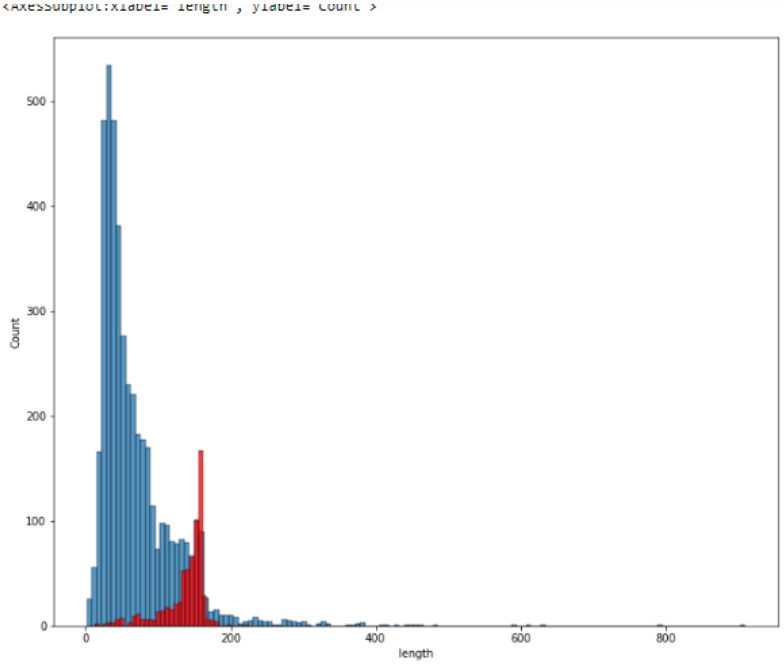
```
<AxesSubplot:xlabel='target', ylabel='count'>
```



- Pie plot

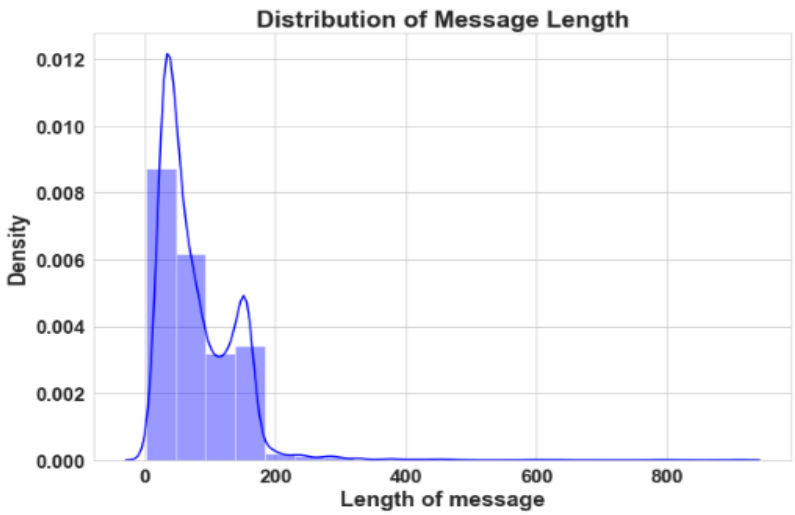


Histplot

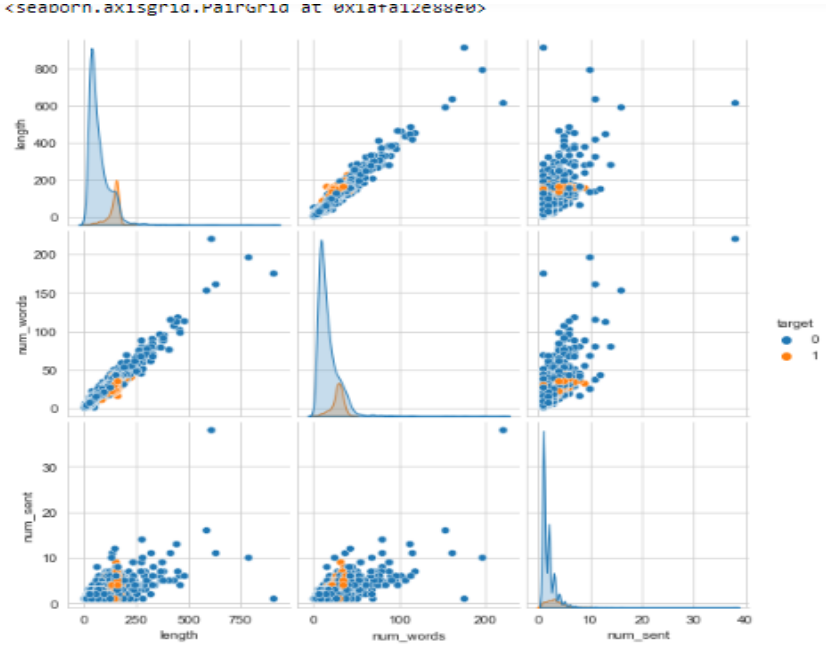


Distplot

```
plt.yticks(fontsize=16,fontweight ='bold')
plt.show()
```



Pair plot



4. Data Inputs- Logic- Output Relationships

i Descriptive Statistics

```
df2.describe()
```

42]:

	target	message
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

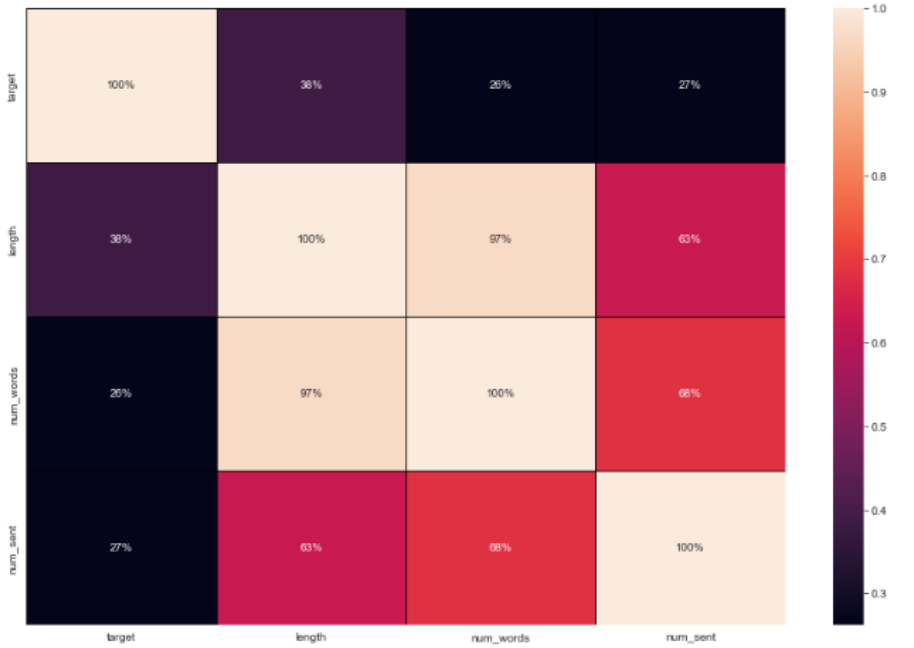
ii Checking Correlation

```
#Checking correlation of the dataset
corr=df3.corr()
corr
```

	target	length	num_words	num_sent
target	1.000000	0.384717	0.262969	0.267602
length	0.384717	1.000000	0.965784	0.626118
num_words	0.262969	0.965784	1.000000	0.680882
num_sent	0.267602	0.626118	0.680882	1.000000

iii Correlation with Heatmap

```
sns.heatmap(corr, linewidth=0.5, linecolor='black', fmt='.0%', annot=True)
plt.show()
```



iv Checking Skewness

```
## checking skewness
df3.skew()
```

```
target       2.250180
length       2.610100
num_words    3.309687
num_sent     4.945423
dtype: float64
```

v creating a coloumn which will contain the numbers of characters

```
df2['length'] = df2['message'].str.len()
df2.head()
```

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

vi creating a coloumn which will fetch numbers of words

```
df2['num_words'] = df2['message'].apply(lambda x: len(nltk.word_tokenize(x)))
df2.head()
```

:

	target	message	length	num_words
0	ham	Go until jurong point, crazy.. Available only ...	111	24
1	ham	Ok lar... Joking wif u oni...	29	8
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155	37
3	ham	U dun say so early hor... U c already then say...	49	13
4	ham	Nah I don't think he goes to usf, he lives aro...	61	15

---

vii creating a coloumn which will fetch numbers of sentences

```
df2['num_sent'] = df2['message'].apply(lambda x: len(nltk.sent_tokenize(x)))
df2.head()
```

]:

	target	message	length	num_words	num_sent
0	ham	Go until jurong point, crazy.. Available only ...	111	24	2
1	ham	Ok lar... Joking wif u oni...	29	8	2
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2
3	ham	U dun say so early hor... U c already then say...	49	13	1
4	ham	Nah I don't think he goes to usf, he lives aro...	61	15	1

---

viii now converting "target" data into binary data i.e., ham for 0 and spam for 1

```
df2['target'] = df2.target.map({'ham':0, 'spam':1})
```

```
df2.head()
```

]:

	target	message	length	num_words	num_sent
0	0	Go until jurong point, crazy.. Available only ...	111	24	2
1	0	Ok lar... Joking wif u oni...	29	8	2
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2
3	0	U dun say so early hor... U c already then say...	49	13	1
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1

---

ix Data description

```
df3[['length','num_words','num_sent']].describe(
```

]:

	length	num_words	num_sent
count	5169.000000	5169.000000	5169.000000
mean	78.977945	18.455407	1.961308
std	58.236293	13.322448	1.432583
min	2.000000	1.000000	1.000000
25%	36.000000	9.000000	1.000000
50%	60.000000	15.000000	1.000000
75%	117.000000	26.000000	2.000000
max	910.000000	220.000000	38.000000

x Average length of the messages

```
comment_len = df3.message.str.len()
df3.message.str.len().median()
```

0]: 60.0

xi Dataset before Pre-processing

```
df4=df3.rename(columns={'length' : 'length_before_cleaning'})
df4
```

]:

	target	message	length_before_cleaning	num_words	num_sent
0	0	Go until jurong point, crazy.. Available only ...	111	24	2
1	0	Ok lar... Joking wif u oni...	29	8	2
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2
3	0	U dun say so early hor... U c already then say...	49	13	1
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1
...	...	...	...	...	...
5567	1	This is the 2nd time we have tried 2 contact u...	161	35	4
5568	0	Will ì_ b going to esplanade fr home?	37	9	1
5569	0	Pity, * was in mood for that. So...any other s...	57	15	2
5570	0	The guy did some bitching but I acted like i'd...	125	27	1
5571	0	Rofl. Its true to its name	26	7	2

5169 rows × 5 columns

xii Data Pre-processing

```
#Defining the stop words
stop_words = stopwords.words('english')

#Defining the lemmatizer
lemmatizer = WordNetLemmatizer()
```

```
#Replacing '\n' in message
df4['message'] = df4['message'].replace('\n',' ')
```

```python
#Function Definition for using regex operations and other text preprocessing for getting cleaned texts
def clean_comments(text):

    #convert to lower case
    lowered_text = text.lower()

    #Replacing email addresses with 'emailaddress'
    text = re.sub(r'^.+@[^\.].*\.[a-z]{2,}$', 'emailaddress', lowered_text)

    #Replace URLs with 'webaddress'
    text = re.sub(r'http\S+', 'webaddress', text)

    #Removing numbers
    text = re.sub(r'[0-9]', " ", text)

    #Removing the HTML tags
    text = re.sub(r"<.*?>", " ", text)

    #Removing Punctuations
    text = re.sub(r'[^\w\s]', ' ', text)
    text = re.sub(r'\_',' ',text)

    #Removing all the non-ascii characters
    clean_words = re.sub(r'[^\x00-\x7f]',r'', text)

    #Removing the unwanted white spaces
    text = " ".join(text.split())

    #Splitting data into words
    tokenized_text = word_tokenize(text)

    #Removing remaining tokens that are not alphabetic, Removing stop words and Lemmatizing the text
    removed_stop_text = [lemmatizer.lemmatize(word) for word in tokenized_text if word not in stop_words if word.isalpha()]

    return " ".join(removed_stop_text)
```

```python
df4["clean_comments"] = df4['message'].apply(clean_comments)
```

```python
df4
```

	target	message	length_before_cleaning	num_words	num_sent	clean_comments
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazy available bugis n great ...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joking wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entry wkly comp win fa cup final tkts st ...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say early hor u c already say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think go usf life around though
...	...	...	...	...	...	...
5567	1	This is the 2nd time we have tried 2 contact u...	161	35	4	nd time tried contact u u å pound prize claim ...
5568	0	Will ì_ b going to esplanade fr home?	37	9	1	ì b going esplanade fr home
5569	0	Pity, * was in mood for that. So...any other s...	57	15	2	pity mood suggestion
5570	0	The guy did some bitching but I acted like i'd...	125	27	1	guy bitching acted like interested buying some...
5571	0	Rofl. Its true to its name	26	7	2	rofl true name

5169 rows × 6 columns

```python
df4['length_after_cleaning'] = df4['clean_comments'].map(lambda clean_comments: len(clean_comments))
df4.head()
```

	target	message	length_before_cleaning	num_words	num_sent	clean_comments	length_after_cleaning
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazy available bugis n great ...	82
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joking wif u oni	23
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entry wkly comp win fa cup final tkts st ...	101
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say early hor u c already say	35
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think go usf life around though	35

```python
print("Original Length:", df4.length_before_cleaning.sum())
print("Cleaned Length:", df4.length_after_cleaning.sum())
print("Total Words Removed:", (df4.length_before_cleaning.sum()) - (df4.length_after_cleaning.sum()))
```

```
Original Length: 408237
Cleaned Length: 249465
Total Words Removed: 158772
```

```python
df4.shape
```

```
(5169, 7)
```

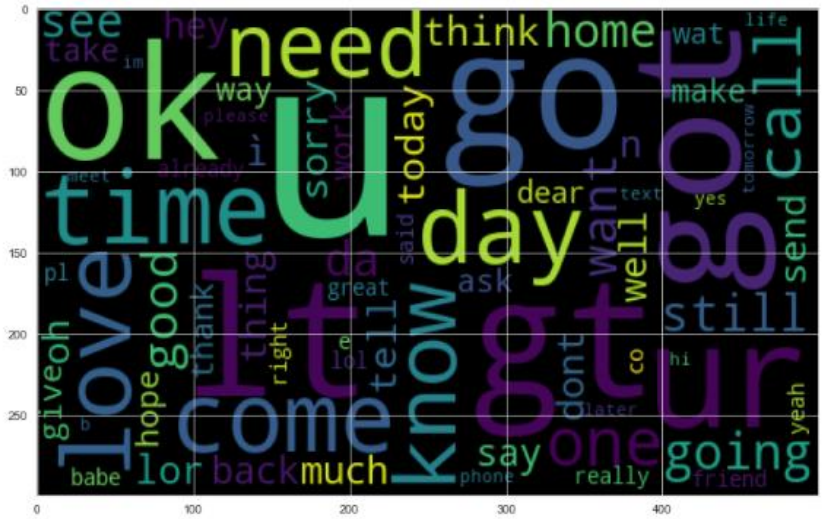
xiii World cloud

```
wc = WordCloud(width = 500, height = 300, min_font_size= 10, background_color= 'black')
```

```
#Generating Word Cloud for Spam Messages
spam_wordcloud = wc.generate(df4[df4['target']==1]['clean_comments'].str.cat(sep = " "))
plt.figure(figsize=(12,8))
plt.imshow(spam_wordcloud)
plt.show()
```

In the above Word Cloud, we can clearly see the words like text, free, reply, call, claim etc. these words are generally seen in the SPAM Message.

```
#Generating Word Cloud for Ham Messages
ham_wordcloud = wc.generate(df4[df4['target']==0]['clean_comments'].str.cat(sep = " "))
plt.figure(figsize=(12,8))
plt.imshow(ham_wordcloud)
plt.show()
```



In the above Word Cloud, we can clearly see the words like love, come, go, call, time etc. these words are generally seen in the Ham Message.

xiv Convert text into vectors using TF-IDF

```
#Convert text into vectors using TF-IDF
tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
features = tf_vec.fit_transform(df4['message'])
x = features
y=df4['target']
```

```
df4.shape
```

```
(5169, 7)
```


5. State the set of assumptions (if any) related to theproblem 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 tools:**

1. Windows laptop
2. i5 processor
3. 4GB ram 4. 250 GB SSD card

- **Software tools:**

1. windows 10
2. Anaconda Navigator
3. Jupyter Notebook
4. Python

- **Libraries and packages:**

1. Pandas
2. NumPy
3. SciPy
4. Seaborn
5. Mat plot
6. Sklearn

And

```
# 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
```

```
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
- Information about Data
- Checked Description of Data
- Dropped irrelevant Columns
- Checked all features through visualization.
- Checked correlation of features
- Converted all messages to lower case
- Removed Punctuation
- Replaced extra space
- Replaced leading and trailing white space
- Removed \n
- Added and removed stop words
- Words of Sentence
- Calculated length of sentence
- Checked the word which are Ham messages
- Checked the word which are Spam messages
- Converted text into vectors using TF-IDF

Testing of Identified Approaches (Algorithms)

1. Logistic Regression
2. Linear Support Vector Classifier

3. Bernoulli NB
4. Multinomial NB
5. XGB Classifier
6. SGD Classifier

2. Run and evaluate selected models

Creating Model

```
# creating new train test split using the random state.
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=20)
```

```
x.shape, y.shape
```

```
[104]: ((5169, 8404), (5169,))
```

```
x_train.shape,y_train.shape, x_test.shape,y_test.shape
```

```
[105]: ((3618, 8404), (3618,), (1551, 8404), (1551,))
```

1. Logistic Regression

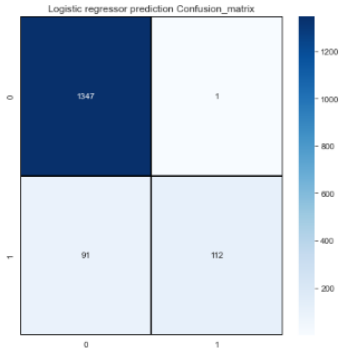
```
lr=LogisticRegression()
lr.fit(x_train,y_train)
pred_lr=lr.predict(x_test)

print("accuracy_score: ", accuracy_score(y_test, pred_lr))
print("confusion_matrix: \n", confusion_matrix(y_test, pred_lr))
print("classification_report: \n", classification_report(y_test,pred_lr))
```

```
accuracy_score:  0.9406834300451322
confusion_matrix:
 [[1347    1]
 [  91  112]]
classification_report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      1348
           1       0.99      0.55      0.71       203

    accuracy                           0.94      1551
   macro avg       0.96      0.78      0.84      1551
weighted avg       0.94      0.94      0.93      1551
```

Confusion Matrix for Logistic Regression

```
cm = confusion_matrix(y_test,pred_lr)
x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("Logistic regressor prediction Confusion_matrix")
```

```
[18]: Text(0.5, 1.0, 'Logistic regressor prediction Confusion_matrix')
```



Cross Validation Score for Logistic Regression

```
print('CV score for Logistic Regression: ',cross_val_score(lr,x,y,cv=5).mean())
```

```
CV score for Logistic Regression:  0.9450568380765493
```

2.LinearSVC

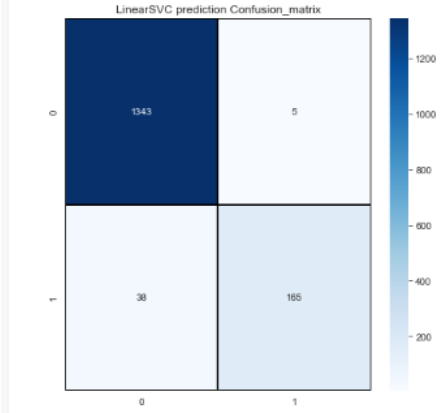
```
svc = LinearSVC()
svc.fit(x_train,y_train)
pred_svc=svc.predict(x_test)

print("accuracy_score: ", accuracy_score(y_test, pred_svc))
print("confusion_matrix: \n", confusion_matrix(y_test, pred_svc))
print("classification_report: \n", classification_report(y_test,pred_svc))
```

```
accuracy_score:  0.9722759509993553
confusion_matrix:
 [[1343    5]
 [  38  165]]
classification_report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1348
           1       0.97      0.81      0.88       203

    accuracy                           0.97      1551
   macro avg       0.97      0.90      0.93      1551
weighted avg       0.97      0.97      0.97      1551
```

Confusion Matrix for LinearSVC

```
cm = confusion_matrix(y_test,pred_svc)
x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("LinearSVC prediction Confusion_matrix")
```

```
Text(0.5, 1.0, 'LinearSVC prediction Confusion_matrix')
```



Cross Validation Score for LinearSVC

```
print('CV score for LinearSVC: ',cross_val_score(svc,x,y,cv=5).mean())
```

```
CV score for LinearSVC:  0.9750429258081006
```

3.BernoulliNB

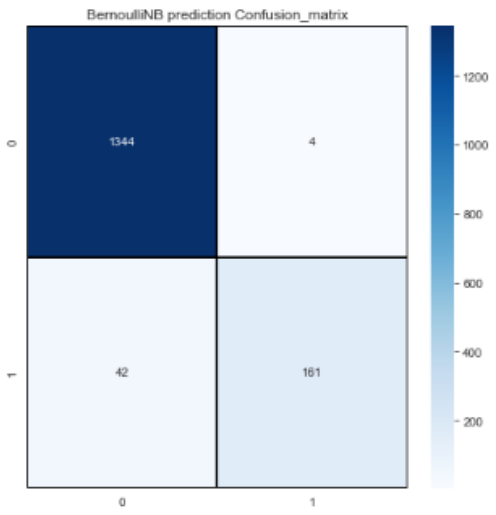
```
bnb = BernoulliNB()
bnb.fit(x_train,y_train)
pred_bnb=bnb.predict(x_test)

print("accuracy_score: ", accuracy_score(y_test, pred_bnb))
print("confusion_matrix: \n", confusion_matrix(y_test, pred_bnb))
print("classification_report: \n", classification_report(y_test,pred_bnb))
```

```
accuracy_score:  0.970341715022566
confusion_matrix:
 [[1344    4]
 [  42  161]]
classification_report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1348
           1       0.98      0.79      0.88       203

    accuracy                           0.97      1551
   macro avg       0.97      0.90      0.93      1551
weighted avg       0.97      0.97      0.97      1551
```

Confusion Matrix for BernoulliNB

```
cm = confusion_matrix(y_test,pred_bnb)
x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("BernoulliNB prediction Confusion_matrix")
```

```
Text(0.5, 1.0, 'BernoulliNB prediction Confusion_matrix')
```



Cross Validation Score for BernoulliNB

```
print('CV score for BernoulliNB: ',cross_val_score(bnb,x,y,cv=5).mean())
```

```
CV score for BernoulliNB:  0.9725289807718595
```

4.MultinomialNB

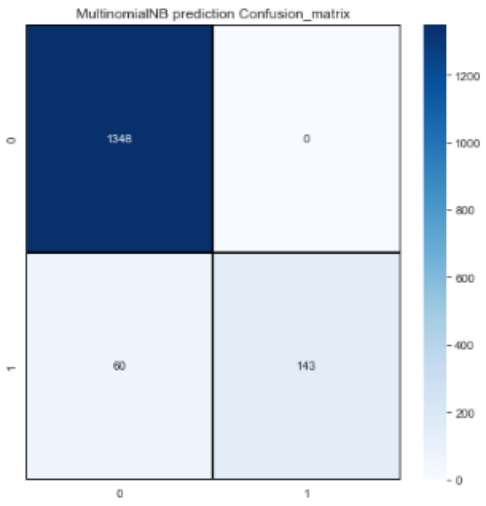
```
mnb = MultinomialNB()
mnb.fit(x_train,y_train)
pred_mnb=mnb.predict(x_test)

print("accuracy_score: ", accuracy_score(y_test, pred_mnb))
print("confusion_matrix: \n", confusion_matrix(y_test, pred_mnb))
print("classification_report: \n", classification_report(y_test,pred_mnb))
```

```
accuracy_score:  0.9613152804642167
confusion_matrix:
 [[1348    0]
 [  60  143]]
classification_report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      1348
           1       1.00      0.70      0.83       203

    accuracy                           0.96      1551
   macro avg       0.98      0.85      0.90      1551
weighted avg       0.96      0.96      0.96      1551
```

Confusion Matrix for MultinomialNB

```
cm = confusion_matrix(y_test,pred_mnb)
x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("MultinomialNB prediction Confusion_matrix")
```

```
Text(0.5, 1.0, 'MultinomialNB prediction Confusion_matrix')
```



cross validation score for MultinomialNB

```
print('CV score for MultinomialNB: ',cross_val_score(mnb,x,y,cv=5).mean())
```

```
CV score for MultinomialNB:  0.9630482285731405
```

5. XGBClassifier ¶

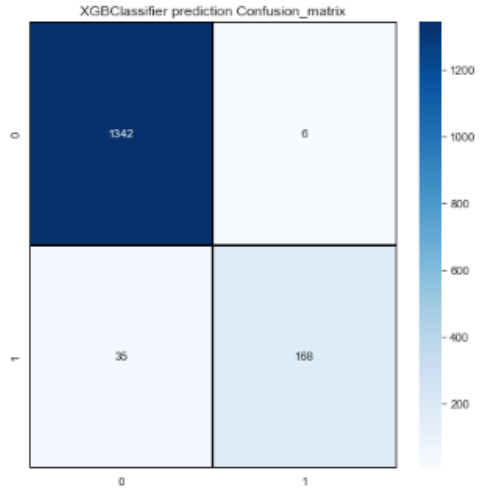
```
xgb = XGBClassifier(verbosity=0)
xgb.fit(x_train,y_train)
pred_xgb=xgb.predict(x_test)

print("accuracy_score: ", accuracy_score(y_test, pred_xgb))
print("confusion_matrix: \n", confusion_matrix(y_test, pred_xgb))
print("classification_report: \n", classification_report(y_test,pred_xgb))
```

```
accuracy_score:  0.973565441650548
confusion_matrix:
 [[1342    6]
 [  35  168]]
classification_report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1348
           1       0.97      0.83      0.89       203

    accuracy                           0.97      1551
   macro avg       0.97      0.91      0.94      1551
weighted avg       0.97      0.97      0.97      1551
```

Confusion Matrix for XGBClassifier

```
cm = confusion_matrix(y_test,pred_xgb)
x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("XGBClassifier prediction Confusion_matrix")
```

```
Text(0.5, 1.0, 'XGBClassifier prediction Confusion_matrix')
```



cross validation score for XGBClassifier

```
print('CV score for XGBClassifier: ',cross_val_score(xgb,x,y,cv=5).mean())
```

```
CV score for XGBClassifier:  0.9725286062828029
```

6.SGDClassifier

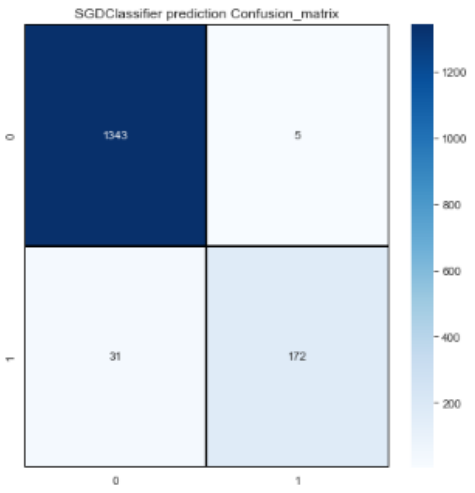
```
sgd= SGDClassifier()
sgd.fit(x_train,y_train)
pred_sgd=sgd.predict(x_test)

print("accuracy_score: ", accuracy_score(y_test, pred_sgd))
print("confusion_matrix: \n", confusion_matrix(y_test, pred_sgd))
print("classification_report: \n", classification_report(y_test,pred_sgd))
```

```
accuracy_score:  0.97678916827853
confusion_matrix:
 [[1343    5]
 [  31  172]]
classification_report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1348
           1       0.97      0.85      0.91       203

    accuracy                           0.98      1551
   macro avg       0.97      0.92      0.95      1551
weighted avg       0.98      0.98      0.98      1551
```

Confusion Matrix for SGDClassifier

```
cm = confusion_matrix(y_test,pred_sgd)
x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Blues",
xticklabels=x_axis_labels,
yticklabels=y_axis_labels)
plt.title("SGDClassifier prediction Confusion_matrix")
```

[148]: Text(0.5, 1.0, 'SGDClassifier prediction Confusion_matrix')



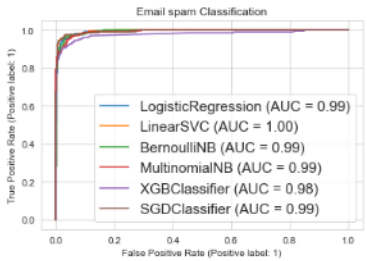
cross validation score for SGDClassifier

```
print('CV score for SGDClassifier: ',cross_val_score(sgd,x,y,cv=5).mean())
```

```
CV score for SGDClassifier:  0.9779446542623408
```

ROC & AUC Curve for all model

```
#Lets plot roc curve and check auc and performance of all algorithms
from sklearn.metrics import roc_curve, auc, roc_auc_score, accuracy_score,classification_report, confusion_matrix, plot_roc_c
disp = plot_roc_curve(lr, x_test, y_test)
plot_roc_curve(svc, x_test, y_test, ax = disp.ax_)
plot_roc_curve(bnb, x_test, y_test, ax = disp.ax_)
plot_roc_curve(mnb, x_test, y_test, ax = disp.ax_)
plot_roc_curve(xgb, x_test, y_test, ax = disp.ax_)
plot_roc_curve(sgd, x_test, y_test, ax = disp.ax_)
plt.title("Email spam Classification")
plt.legend(prop={"size" :15} ,loc = 'lower right')
plt.show()
```



HyperParameter Tuning

Linear SVC with GridSearchCV

```
grid_params =  {'C':(0.001, 0.01, 0.1, 1, 10),
                'penalty':('l1','l2'),
               'loss':('hinge','squared_hinge')}

# Train the model with given parameters using GridSearchCV
LSVC =  GridSearchCV(svc, grid_params, cv=5)
LSVC.fit(x_train, y_train)
```

```
GridSearchCV(cv=5, estimator=LinearSVC(),
             param_grid={'C': (0.001, 0.01, 0.1, 1, 10),
                         'loss': ('hinge', 'squared_hinge'),
                         'penalty': ('l1', 'l2')})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
# Selecting the best parameters found by GridSearchCV
print(LSVC.best_params_)
print(LSVC.best_score_)
```

```
{'C': 10, 'loss': 'squared_hinge', 'penalty': 'l2'}
0.9762289570008329
```

Best model

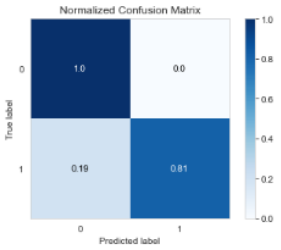
```
best_model = LinearSVC(C= 1, loss= 'squared_hinge', penalty= 'l2')
best_model.fit(x_train,y_train)
pred = best_model.predict(x_test)
accuracy = accuracy_score(y_test, pred)*100
print("ACCURACY SCORE:", accuracy)
print(f"\nCLASSIFICATION REPORT: \n {classification_report(y_test, pred)}")
print(f"\nCONFUSION MATRIX: \n {confusion_matrix(y_test, pred)}")
```

```
ACCURACY SCORE: 97.22759509993553

CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1348
           1       0.97      0.81      0.88       203

    accuracy                           0.97      1551
   macro avg       0.97      0.90      0.93      1551
weighted avg       0.97      0.97      0.97      1551


CONFUSION MATRIX:
 [[1343    5]
 [  38  165]]
```

CONFUSION MATRIX

```
skplt.metrics.plot_confusion_matrix(y_test, pred, normalize=True)
```

```
<AxesSubplot:title={'center':'Normalized Confusion Matrix'}, xlabel='Predicted label', ylabel='True label'>
```

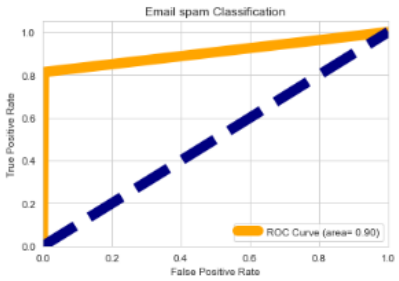


So the best accuracy score is 97.2.

ROC-AUC Curve

```
fpr, tpr, threshold = roc_curve(y_test,pred)
auc = roc_auc_score(y_test,pred)
```

```
plt.figure()
plt.plot(fpr,tpr,color="orange",lw=10,label="ROC Curve (area= %0.2f)" % auc)
plt.plot([0,1],[0,1],color="navy",lw=10,linestyle="--")
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Email spam Classification")
plt.legend(loc="lower right")
plt.show()
```



Saving the Model

```
joblib.dump(best_model, "Email_Spam_Detection_Classifier.pkl")
```

```
['Email_Spam_Detection_Classifier.pkl']
```

Checking predicted and original values

```
Model = joblib.load("Email_Spam_Detection_Classifier.pkl")
# Predicting test data using loaded model
prediction = Model.predict(x_test)
# Analysing Predicted vs Actual results
Email_Spam_Detection_Classifier = pd.DataFrame()
Email_Spam_Detection_Classifier['Predicted Spam Messages Detection'] = prediction
Email_Spam_Detection_Classifier['Actual Spam Messages Detection'] = y
Email_Spam_Detection_Classifier
```

	Predicted Spam Messages Detection	Actual Spam Messages Detection
0	0	0.0
1	0	0.0
2	1	1.0
3	0	0.0
4	0	0.0
...	...	...
1546	0	0.0
1547	0	0.0
1548	1	0.0
1549	0	0.0
1550	0	0.0

```
# Converting the dataframe into CSV format and saving it
Email_Spam_Detection_Classifier.to_csv('Email_Spam_Detection_Classifier.csv', index=False)
```

- **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 and AUC-ROCCurve is used for success.

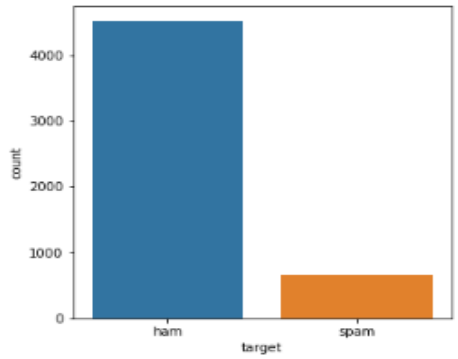
- **Visualizations**
    - Count plot

Countplot

```
print(df2['target'].value_counts())
plt.figure(figsize=(5,5))
sns.countplot('target', data=df2)
```

```
ham     4516
spam     653
Name: target, dtype: int64
```

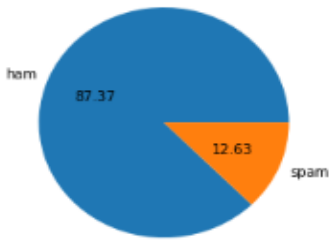
```
<AxesSubplot:xlabel='target', ylabel='count'>
```



    - Pie-plot

2. Pie-Plot

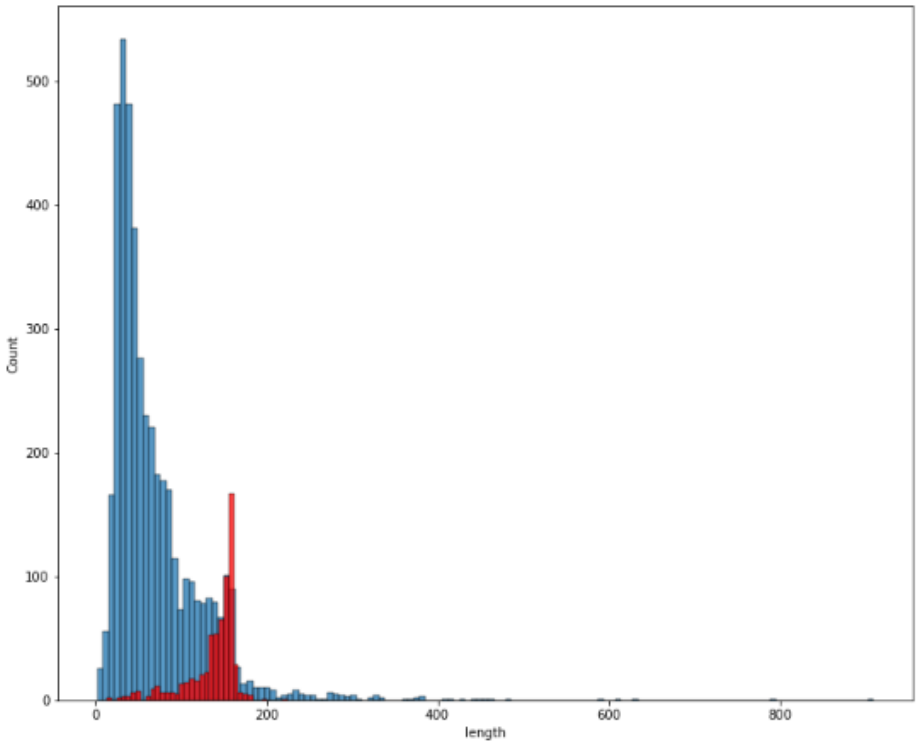
```
plt.pie(df2['target'].value_counts(), labels =['ham', 'spam'], autopct= "%.02f")
plt.show()
```



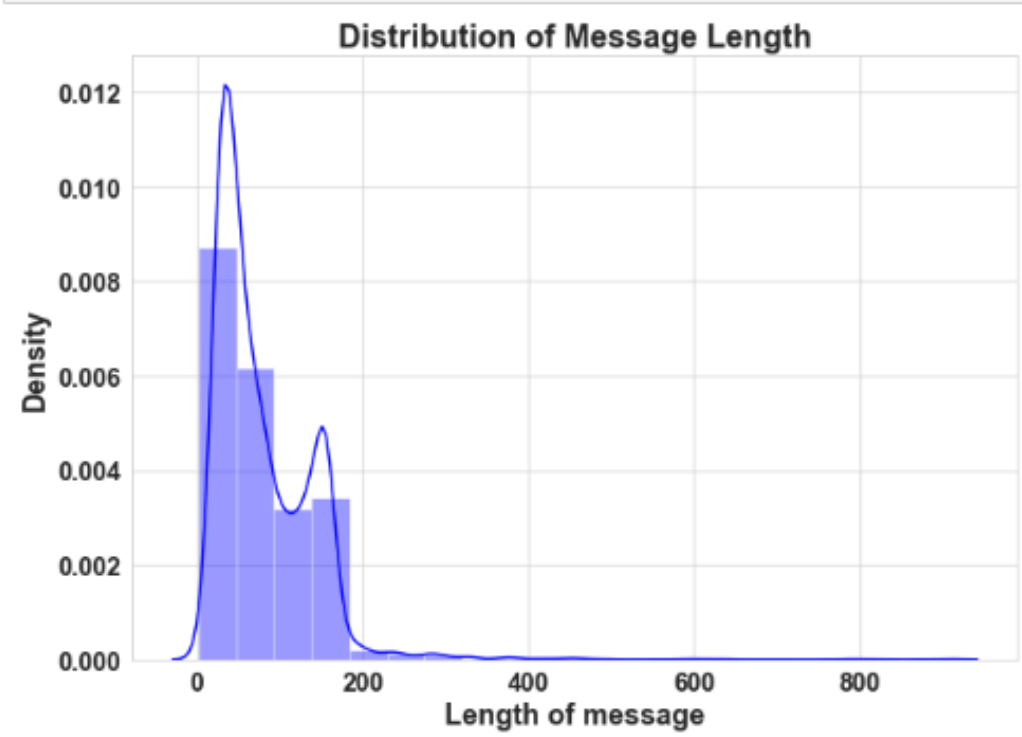
- Histplot

```
plt.figure(figsize=(12,10))
sns.histplot(df3[df3['target']==0]['length'])
sns.histplot(df3[df3['target']==1]['length'],color = 'red')
```

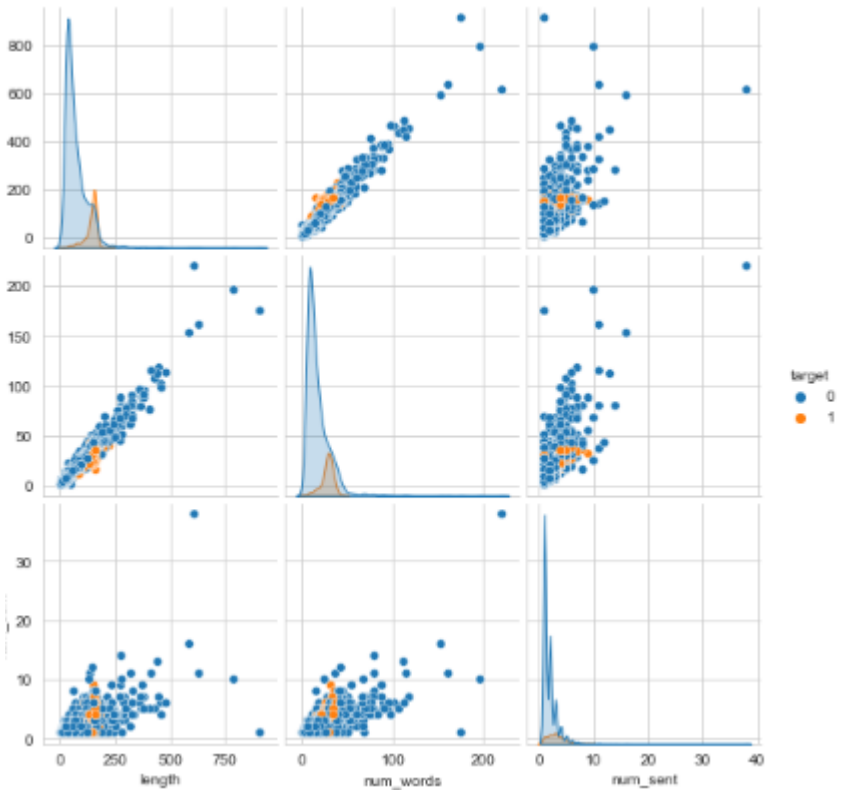
```
<AxesSubplot:xlabel='length', ylabel='Count'>
```



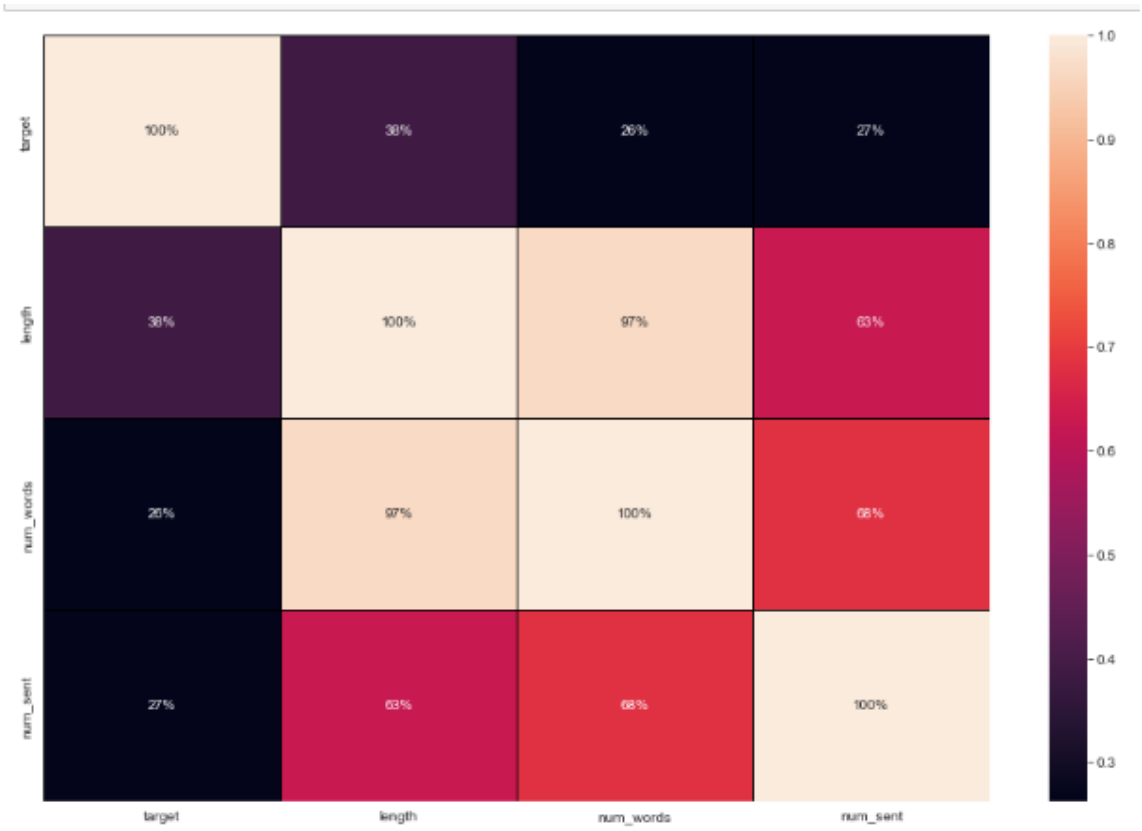
- Distplot



- Pair plot



- Heatmap



- **Interpretation of the Results**

  - Through Pre-processing it is interpretated that all text 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
  - Used 6 Machine Learning Algorithms for choosing one best model which is giving best accuracy than others
  - By creating/building model we get best model: Linear SVC

CONCLUSION

1. 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.

2. 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 analyze and interpret the huge data and with the help of pie plot, count plot & word cloud, I am able to see ham and spam messages.

- Through data cleaning we were able to remove unnecessary columns, values, special characters, symbols, stop-words and punctuation from our dataset due to which our model would have suffered from over fitting or under fitting.

The few challenges while working on this project were: -

- To find punctuations & stop words, which took time to run using NLP.
- The data set is huge it took time to run some algorithms & to check thecross-validation score.

3. 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.