FLIP ROBO

EMAIL SPAM DETECTION

Submitted by:

Shubham Hajari

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I am also grateful to Ms. Gulshana Chaudhary for her constant guidance and support.

Some of the reference sources are as follows:

- Internet
- Coding Ninjas
- Medium.com
- Analytics Vidhya
- Using Naive Bayes Model and Natural Language Processing for Classifying Messages on Online Forum (Research Paper)

TABLE OF CONTENTS

<u>INTRODUCTION</u>

BUSINESS PROBLEM FRAMING

You were recently hired in a Start-up Company and was asked to build a system to identify spam emails. We will explore and understand the process of classifying Emails as Spam or Not Spam by build Machine Learning and NPL model to detect the HAM and SPAM mails. The model will detect the unsolicited and unwanted emails and thus we can prevent them from creeping into user's inbox and therefore, increase the user Experience.

CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

As we know how a machine translates language, or how voice assistants respond to questions, or how mail gets automatically classified into spam or not spam, all these tasks are done through Natural Language Processing (NLP), which processes text into useful insights that can be applied to future data. In the field of artificial intelligence, NLP is one of the most complex areas of research due to the fact that text data is contextual. It needs modification to make it machine-interpretable and requires multiple stages of processing for feature extraction.

Classification problems can be broadly split into two categories: binary classification problems, and multi-class classification problems. Binary classification means there are only two possible label classes, e.g. a patient's condition is cancerous or it isn't, or a financial transaction is fraudulent or it is not. Multi-class classification refers to cases where there are more than two label classes. An example of this is classifying the sentiment of a movie review into positive, negative, or neutral.

There are many types of NLP problems, and one of the most common types is the classification of strings. Examples of this include the classification of movies/news articles into different genres and the automated classification of emails into a spam or not spam. We shall be looking into this last example in more detail for this project.

REVIEW OF LITERATURE

In recent times, unwanted commercial / promotional bulk emails also known as spam has become a huge problem on the internet and for our mail inbox. An individual / organization sending the spam messages are referred to as the spammers. Such a person gathers email addresses from different websites, chatrooms, and other sources to send the mail to bulk audience. Spam prevents the user from making full and good use of time, storage capacity and network bandwidth. The huge volume of spam mails flowing through the computer networks have destructive effects on the memory space of email servers, communication bandwidth, CPU power and user time. The menace of spam email is on the increase on yearly basis and is responsible for over 80% of the whole global email traffic (Source google).

Users who receive spam emails that they did not request find it very irritating. It is also resulted to untold financial loss to many users who have fallen victim of internet scams and other fraudulent practices of spammers who send emails pretending to be from reputable companies with the intention to persuade individuals to disclose sensitive personal information like passwords, Bank Verification Number (BVN) and credit card numbers.

MOTIVATION FOR THE PROBLEM UNDERTAKEN

Motivation for this project has been undertaken because it is a project which is assigned to me during my internship at Flip Robo Technologies. This project will help Start-up companies to detect and filter the SPAM mails in their Email inbox and therefore, increase the user experience and save their server from unwanted mails, phishing mails or other viruses.

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL/ ANALYTICAL MODELING OF THE PROBLEM

Throughout the project multiple mathematical and analytical models have been used, first we have checked the ratio of spam and ham emails in our dataset. The shape of our data set is 5572 rows and 5 columns.

Then we have used regular expressions to clean the message column which contained body of the email. Then we have used TfidfVectorizer, to transforms text to feature vectors that can be used as input to estimator.

In [7]:
```
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 SOURCES AND THEIR FORMATS

The data was provided to us from the Flip Robo Technologies as a part of our Internship assignment. The data was provided in CSV format and there were 5 attributes and 5572 rows in the data set.

In [2]:
```
df=pd.read_csv("spam.csv",sep="\t")
df
```

Out[2]:

	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
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will Ì_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

DATA PREPROCESSING DONE

After loading all the data, we will proceeded with the data pre-processing. FollowingSteps were followed during data pre-processing:

- **Removing unwanted and renaming attribute from Dataset :**

It's quite hard to find whether a mail is a spam or not just by looking at the subject. So we started by replacing the null values.

1. Data Cleaning

In [7]:
```
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
```

In [8]:
```
# drop last 3 columns
df.drop(columns=['Unnamed: 2','Unnamed: 3', 'Unnamed: 4'], inplace = True)
```

In [9]:
```
df.sample(5)
```

Out[9]:

	v1	v2
791	ham	All e best 4 ur driving tmr :-)
1626	ham	Dear how you. Are you ok?
1492	ham	In the end she might still vomit but its okay....
753	ham	When did you get to the library
747	spam	U are subscribed to the best Mobile Content Se...

In [10]:
```
# Renaming the columns
df.rename(columns={'v1':'target', 'v2':'text'}, inplace=True)
```

- **Label Encoding:**

Using label encoding method converted target column data type into int type for in order to get the better accuracy while training and testing the model.

In [12]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

In [13]:
```
df['target']= encoder.fit_transform(df['target'])
```

In [14]:
```
df.head()
```

Out[14]:

	target	text
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...

- **Remove duplicated values:**

In [16]:
```
# check for duplicate values
df.duplicated().sum()
```

Out[16]:
```
403
```

In [17]:
```
# remove duplicates
df = df.drop_duplicates(keep='first')
```

In [18]:
```
df.duplicated().sum()
```

Out[18]:
```
0
```

In [19]:
```
df.shape
```

Out[19]:
```
(5169, 2)
```

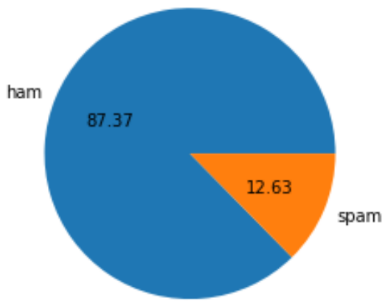
- **EDA:**

In [21]:
```
df['target'].value_counts()
```

Out[21]:
```
0    4516
1     653
Name: target, dtype: int64
```

In [22]:
```
import matplotlib.pyplot as plt
```

In [23]:
```
plt.pie(df['target'].value_counts(), labels= ['ham', 'spam'], autopct='%0.2f')
plt.show()
```



In [27]:
```
df['num_characters'] = df['text'].apply(len)
```

In [28]:
```
df.head()
```

Out[28]:

	target	text	num_characters
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

In [29]:
```
# num of words
df['num_words'] = df['text'].apply(lambda x : len(nltk.word_tokenize(x)))
```

In [30]:
```
df.head()
```

Out[30]:

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

In [31]:
```
# number of sentences
df['num_sentences'] = df['text'].apply(lambda x : len(nltk.sent_tokenize(x)))
```

In [32]:
```
df.head()
```

Out[32]:

	target	text	num_characters	num_words	num_sentences
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

In [33]:
```
df[['num_characters','num_words','num_sentences']].describe() # to get the insights of data
```

Out[33]:

	num_characters	num_words	num_sentences
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

In [34]:
```
# Ham messages
df[df['target'] == 0][['num_characters','num_words','num_sentences']].describe()
```

Out[34]:

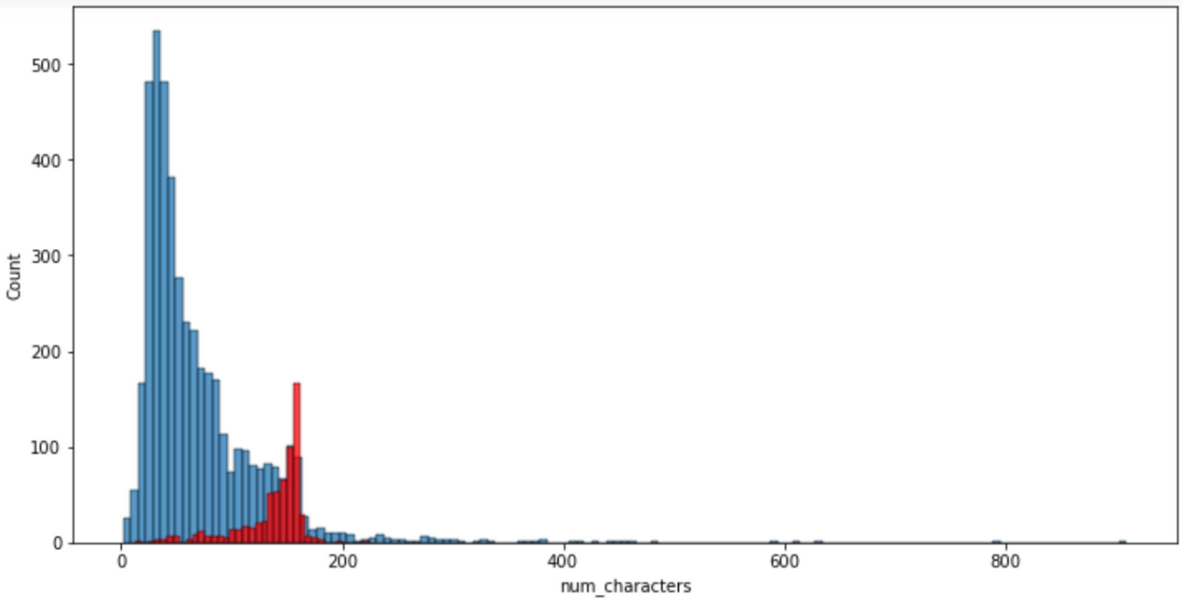
	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.459256	17.123339	1.815545
std	56.358207	13.491315	1.364098
min	2.000000	1.000000	1.000000
25%	34.000000	8.000000	1.000000
50%	52.000000	13.000000	1.000000
75%	90.000000	22.000000	2.000000
max	910.000000	220.000000	38.000000

In [35]:
```
# Spam messages   --- bigger in words, char, sent
df[df['target'] == 1][['num_characters','num_words','num_sentences']].describe()
```

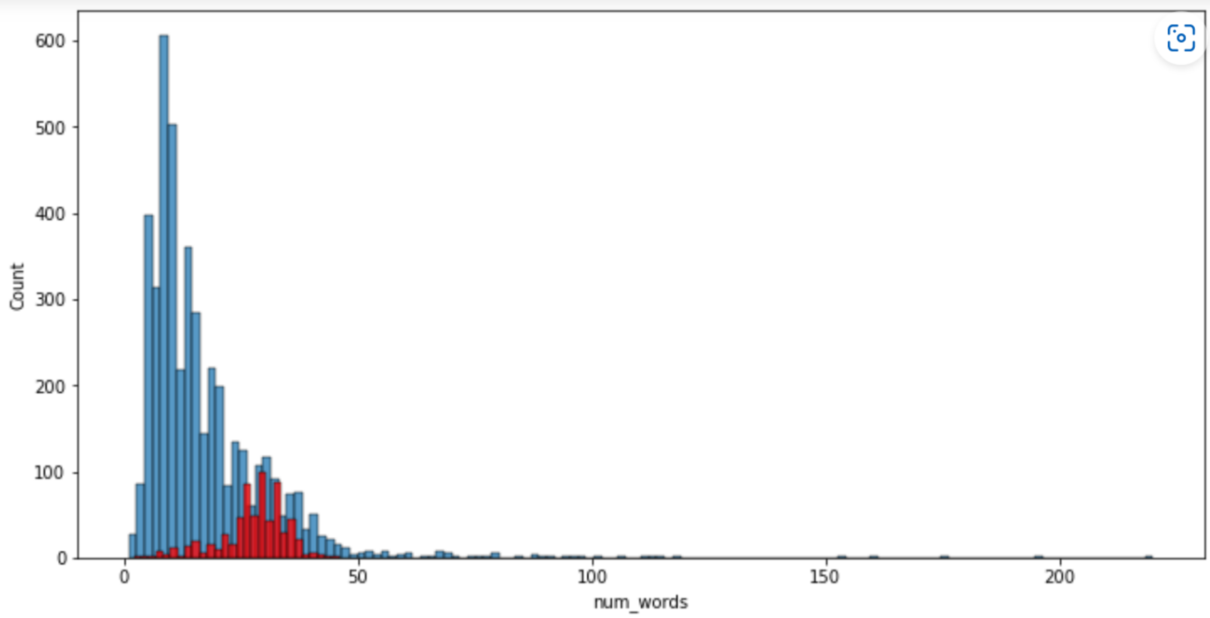
Out[35]:

	num_characters	num_words	num_sentences
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

In [37]:
```
plt.figure(figsize =(12,6))
sns.histplot(df[df['target'] == 0]['num_characters']);
sns.histplot(df[df['target'] == 1]['num_characters'], color = 'red');
```



In [38]:
```
plt.figure(figsize =(12,6))
sns.histplot(df[df['target'] == 0]['num_words']);
sns.histplot(df[df['target'] == 1]['num_words'], color = 'red');
```

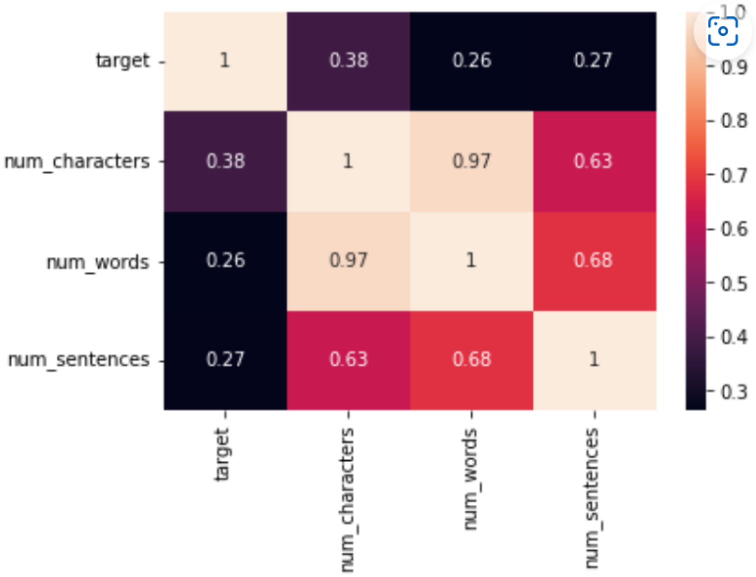


In [41]:
```
df.corr()
```

Out[41]:

	target	num_characters	num_words	num_sentences
target	1.000000	0.384717	0.262969	0.267602
num_characters	0.384717	1.000000	0.965784	0.626118
num_words	0.262969	0.965784	1.000000	0.680882
num_sentences	0.267602	0.626118	0.680882	1.000000

In [42]:
```
sns.heatmap(df.corr(), annot = True); # will keep num characters cokumn
```



```
In [43]: from nltk.corpus import stopwords   # sentence formation not for meaning
         #stopwords.words('english')
```

```
In [44]: from nltk.stem.porter import PorterStemmer   # to root form
         ps = PorterStemmer()
```

```
In [45]: import string
         #string.punctuation
```

```
In [46]: 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[:]  # Copying list we have to clone it
             y.clear()

             for i in text:
                 if i not in stopwords.words('english') and i not in string.punctuation:
                     y.append(i)
             text = y[:]
             y.clear()

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

             return " ".join(y)
```

```
In [48]: df['transformed_text'] = df['text'].apply(transform_text)
```

```
In [49]: df.head()
```

Out[49]:

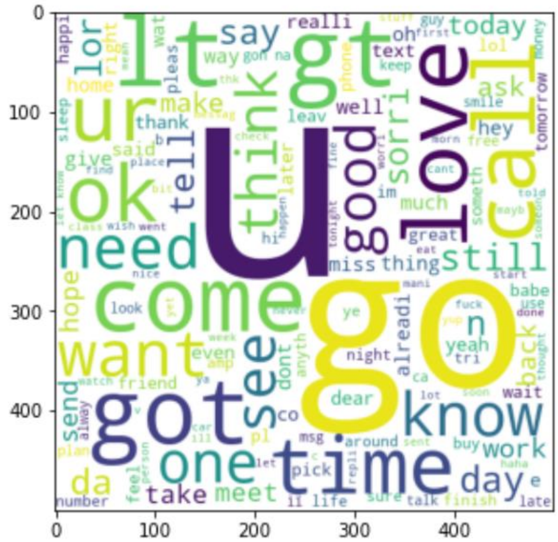
	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

```
In [50]: # Word Cloud

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

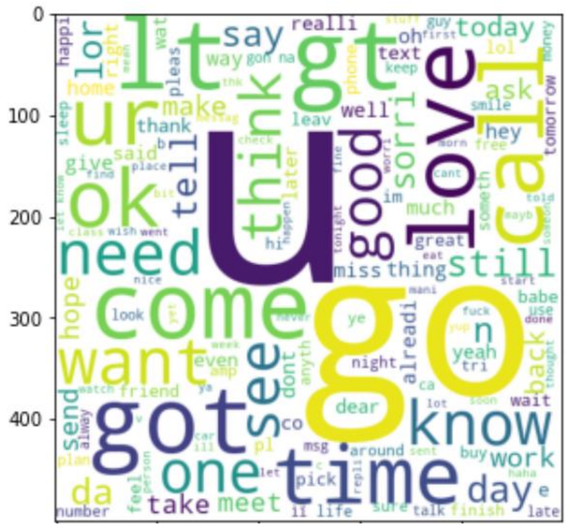
```
In [51]: spam_wc = wc.generate(df[df['target']==1]['transformed_text'].str.cat(sep = " "))
```

```
In [98]: plt.figure(figsize=(5,8))
         plt.imshow(spam_wc);
```




```
In [53]: ham_wc = wc.generate(df[df['target']==0]['transformed_text'].str.cat(sep = " "))
```

```
In [99]: plt.figure(figsize=(5,8))
         plt.imshow(ham_wc);
```



4. Model Building

In [62]:
```
# textual data ---> Naive bayes best performance
# numerical input  ---> vectorize  (bag of words, tfidf, word2vec)
```

In [63]:
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

In [64]:
```
X = tfidf.fit_transform(df['transformed_text']).toarray()  # it gives sparse array --> convert to dense
```

In [65]:
```
#from sklearn.preprocessing import MinMaxScaler
#scaler = MinMaxScaler()
#X = scaler.fit_transform(X)
```

In [66]:
```
X.shape
```

Out[66]:
```
(5169, 3000)
```

In [67]:
```
y = df['target'].values
```

In [68]:
```
y
```

Out[68]:
```
array([0, 0, 1, ..., 0, 0, 0])
```

In [69]:
```
from sklearn.model_selection import train_test_split
```

In [70]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

In [71]:
```
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score
# spam classifier --> less false positive
```

In [72]:
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [73]:
```
gnb.fit(X_train, y_train)  # Low precision model
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
```

In [74]:
```
mnb.fit(X_train, y_train)  # imbalanced data precision data as high we can get not accuracy
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
```

```
In [78]: 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)
```

```
In [79]: clfs = {
             'SVC': svc,
             'KN':knc,
             'NB': mnb,
             'DT': dtc,
             'LR': lrc,
             'RF':rfc,
             'AdaBoost': abc,
             'BgC': bc,
             'ETC': etc,
             'GBDT': gbdt,
             'xgb': xgb
         }
```

```
In [80]: def train_classifier(clf, X_train, y_train):
             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
```

```
In [81]: train_classifier(svc, X_train, y_train)

Out[81]: (0.9758220502901354, 0.9747899159663865)
```

```
In [82]: accuracy_scores = []
         precision_scores = []

         for name, clf in clfs.items():
             current_accuracy, current_precision = train_classifier(clf, X_train, y_train)
             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 0.9294003868471954
Precision 0.8282828282828283
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
```

In [84]:
```
performance_df
```

Out[84]:

	Algorithm	Accuracy	Precision
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
5	RF	0.975822	0.982906
0	SVC	0.975822	0.974790
8	ETC	0.974855	0.974576
4	LR	0.958414	0.970297
10	xgb	0.967118	0.933333
6	AdaBoost	0.960348	0.929204
9	GBDT	0.946809	0.919192
7	BgC	0.958414	0.868217
3	DT	0.929400	0.828283

5. Model Improve

In [88]:
```
# 1. Change the max feature of tfidf
```

In [89]:
```
# 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
```

In [90]:
```
voting  = VotingClassifier(estimators=[('svm',svc), ('nb', mnb), ('et',etc)], voting='soft') # Weithage
```

In [91]:
```
voting.fit(X_train, y_train)
```

Out[91]:
```
VotingClassifier
svm        nb                 et
SVC        MultinomialNB      ExtraTreesClassifier
```

In [92]:
```
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
```

In [93]:
```
# applying stacking  ---> give weightage using a final estiamtor
estimators = [('svm', svc), ('nb', mnb), ('et',etc)]
final_estimator = RandomForestClassifier()
```

In [94]:
```
from sklearn.ensemble import StackingClassifier
clf = StackingClassifier(estimators = estimators, final_estimator = final_estimator)
```

In [95]:
```
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 0.9541984732824428
```

In [96]:
```
import pickle
pickle.dump(tfidf, open('vecotizer.pkl','wb'))
pickle.dump(mnb, open('model.pkl', 'wb'))
```

KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION

Precision: can be seen as a measure of quality, higher precision means that an algorithm returns more relevant results than irrelevant ones

Recall is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

Accuracy score is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar

F1-score is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.

Cross_val_score: To run cross-validation on multiple metrics and also to return train scores, fit times and score times. Get predictions from each split of cross-validation for diagnostic purposes. Make a scorer from a performance metric or loss function.

---

roc _auc _score : ROC curve. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0

CONCLUSION

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

From the whole evaluation we found out that the spam emails can be classified and can be stopped doing harm to the users.

LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE

I found visualisation a very useful technique to infer insights from dataset.

The ROC AUC plot gives large info about the false positive rate and True positive rate at various thresholds.

We are able to classify the emails as spam or non-spam. With high number of emails lots if people using the system it will be difficult to handle all possible mails as our project deals with only limited amount of corpus

LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

Since the data contained less number of ‘1’ target labels. The trained model will be limited in scope for this label. More data of spam can definitely improve the model’s performance on identification of Spam mails.