FLIP ROBO

EMAIL SPAM DETECTION

Submitted by:

MAHESHKUMAR OTA

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. Khushboo Garg 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

INTRODUCTION

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 2893 rows and 3 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 textto feature vectors that can be used as input to estimator.

```
In [8]: #Let;s check the detail info of our dataset

        email.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 2893 entries, 0 to 2892
        Data columns (total 3 columns):
         #   Column   Non-Null Count  Dtype
        ---  ------   --------------  -----
         0   subject  2893 non-null   object
         1   message  2893 non-null   object
         2   label    2893 non-null   int64
        dtypes: int64(1), object(2)
        memory usage: 67.9+ KB
```

DATA SOURCES AND THEIR FORMATS

The data was provided to us from the FlipRobo Technologies as a part of our Internship assignment. The data was provided in CSV format and there were 3 attributes and 2893 rows in the data set.

LOADING CSV DATA

```
In [2]: #Let's load the CSV file

        email=pd.read_csv('messages.csv')
        email
```

Out[2]:

	subject	message	label
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0
1	NaN	lang classification grimes , joseph e . and ba...	0
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0
3	risk	a colleague and i are researching the differin...	0
4	request book information	earlier this morning i was on the phone with a...	0
...	...	...	...
2888	love your profile - ysuolvpv	hello thanks for stopping by ! ! we have taken...	1
2889	you have been asked to join kiddin	the list owner of : " kiddin " has invited you...	1

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 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.

```
In [3]:  #Let's check if there are null values in our dataset

         email.isnull().sum()
Out[3]:  subject    62
         message     0
         label       0
         dtype: int64

In [4]:  #Let's replace the null values in our dataset

         email=email.replace(np.nan,"",regex=True)
         email.head()
Out[4]:
```

	subject	message	label
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0
1		lang classification grimes , joseph e . and ba...	0
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0

> **Adding additional attribute :**

In order to analyse the data in a better way while doing pre-processing, we have added an attribute 'Length' which shows length of the message against it. This was done just to compare the length of text before and after preprocessing and to get idea about the memory optimization.

```
In [13]: #Let's check the length

         email['Subject_length'] = email.subject.str.len()
         email['Message_length'] = email.message.str.len()
         email.head(5)
```

Out[13]:

	subject	message	label	Subject_length	Message_length
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0	39	2856
1		lang classification grimes , joseph e . and ba...	0	0	1800
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0	50	1435
3	risk	a colleague and i are researching the differin...	0	4	324
4	request book information	earlier this morning i was on the phone with a...	0	24	1046

> ## Converting all the messages to lower case:

All messages in the 'message' attribute was converted to small case since keeping words in large case does not make sense as same word with small and large case conveys same meaning.

DATA PREPRATION

```
In [14]: # Let's convert all the data to lower case for further processing

         email['subject'] = email['subject'].str.lower()
         email['message'] = email['message'].str.lower()
```

```
In [15]: email
```

Out[15]:

	subject	message	label	Subject_length	Message_length
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0	39	2856
1		lang classification grimes , joseph e . and ba...	0	0	1800
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0	50	1435
3	risk	a colleague and i are researching the differin...	0	4	324
4	request book information	earlier this morning i was on the phone with a...	0	24	1046
...	...	...	...	...	...
2888	love your profile - ysuolypv	hello thanks for stopping by i i we	1	28	262

> ➤ **Performing Regex operations :**

All messages in the 'message' attribute were adjusted to remove unwanted words, characters, numbers etc.

- All mail addresses were replaces by single word 'emailaddress'
- All URL's present in the message were replaced by word 'webaddress'
- All dollars signs ($, £) were replaced by word 'dollers'
- All 10 digit number sequence were replaced by word 'phonenumber'
- All numbers were replaced with word 'numbr'
- Removing punctuations from the message
- Replacing whitespaces between terms with a single space
- Removing leading and trailing whitespaces

DATA CLEANING

```
In [17]: # Let's replace all the unwanted data from the strings of the subject

         # Replace email addresses with 'email'
         strings['subject'] = strings['subject'].str.replace(r'^.+@[^\.].*\.[a

         # Replace URLs with 'webaddress'
         strings['subject'] = strings['subject'].str.replace(r'^http\://[a-zA-

         # Replace 10 digit phone numbers (formats include paranthesis, spaces
         strings['subject'] = strings['subject'].str.replace(r'^\(?[\d]{3}\)?[

         # Replace numbers with 'number'
         strings['subject'] = strings['subject'].str.replace(r'\d+(\.\d+)?', '

         # Replace money symbols with 'moneysymb' (£ can by typed with ALT key
         strings['subject'] = strings['subject'].str.replace(r'£|\$', 'dollers
```

```
In [18]:  # Lets replace all the unwanted data from the strings of the message

          # Replace email addresses with 'email'
          strings['message'] = strings['message'].str.replace(r'^.+@[^\.].*\.[a

          # Replace URLs with 'webaddress'
          strings['message'] = strings['message'].str.replace(r'^http\://[a-zA-

          # Replace 10 digit phone numbers (formats include paranthesis, spaces
          strings['message'] = strings['message'].str.replace(r'^\(?[\d]{3}\)?[

          # Replace numbers with 'number'
          strings['message'] = strings['message'].str.replace(r'\d+(\.\d+)?', '

          # Replace money symbols with 'moneysymb' (£ can by typed with ALT key
          strings['message'] = strings['message'].str.replace(r'£|\$', 'dollers
```

```
In [20]:  # Let's remove the punctuation from subject column

          # Remove punctuation
          strings['subject'] = strings['subject'].str.replace(r'[^\w\d\s]', ' '

          # Remove leading and trailing whitespace
          strings['subject'] = strings['subject'].str.replace(r'^\s+|\s+?$', ''

          # Replace whitespace between terms with a single space
          strings['subject'] = strings['subject'].str.replace(r'\s+', ' ')
```

```
In [21]:  # Let's remove the punctuation from messages column

          # Remove punctuation
          strings['message'] = strings['message'].str.replace(r'[^\w\d\s]', ' '

          # Remove leading and trailing whitespace
          strings['message'] = strings['message'].str.replace(r'^\s+|\s+?$', ''

          # Replace whitespace between terms with a single space
          strings['message'] = strings['message'].str.replace(r'\s+', ' ')
```

```
In [22]:  # Let's check the top 5 entries from our data set post data processing
          strings.head()
```

Out[22]:

	subject	message	label	Subject_length	Message_length
0	job posting apple iss research center	content length number apple iss research cente...	0	39	2856
1		lang classification grimes joseph e and barbar...	0	0	1800
2	query letter frequencies for text identification	i am posting this inquiry for sergei atamas sa...	0	50	1435
3	risk	a colleague and i are researching the differin...	0	4	324
4	request book information	earlier this morning i was on the phone with a...	0	24	1046

> **Removal of Stopwords :**

All unwanted words which do not contribute much in model building i.e. stopwords,were removed from dataset.

BUILDING WORD DICTIONARY

```
In [23]: #Let's create a funtion for Stop words

         stop_words = set(stopwords.words('english'))
         strings['subject'] = strings['subject'].apply(lambda y: ' '.join(term
         strings['message'] = strings['message'].apply(lambda x: ' '.join(term
```

```
In [24]: # Let's create a new columns after puncuations, stopwords removal to

         strings['Subject_clean_length'] = strings.subject.str.len()
         strings['Message_clean_length'] = strings.message.str.len()
         strings.head()
```

Out[24]:

	subject	message	label	Subject_length	Message_length	Subject_clean_len
0	job posting apple iss research center	content length number apple iss research cente...	0	39	2856	
1		lang classification grimes joseph e barbara f ...	0	0	1800	

> ➢ **Adding additional attribute :**

To compare the length of message before preprocessing and after preprocessing an addition column 'Cleaned Length' was added.

After executing all these steps it was found that 2490985 characters were removedfrom the dataset which were of no use and consuming memory.

```
In [25]: # Let's check the total length of the subject and message pre and pos

         print ('Original Length of Subject column:', strings.Subject_length.s
         print ('Length of Subject column post Cleaning:', strings.Subject_cle
         print ('Original Length of messages column:', strings.Message_length.
         print ('Length of messages column post Cleaning:', strings.Message_cl
```

Original Length of Subject column: 91663
Length of Subject column post Cleaning: 79458
Original Length of messages column: 9344743
Length of messages column post Cleaning: 6853758

DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

We have analysed the words that were present in the spam and ham mails, based on the words present and the data we already have which says if the mail is ham or spam, we are going to train the model to predict the same.

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS

USEDHARDWARE:

View basic information about your computer

Windows edition

Windows 10 Home Single
Language
© 2019 Microsoft Corporation.
All rights reserved.

Windows 10

System

Processor:	Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz 2.30 GHz
Installed memory (RAM):	8.00 GB (7.80 GB usable)
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

Computer name, domain, and workgroup settings

Computer name:	Shubham	Change settings
Full computer name:	Shubham	
Computer description:		
Workgroup:	WORKGROUP	

Windows activation

Windows is activated   Read the Microsoft Software License Terms

Product ID: 00327-60000-00000-AA367   Change product key

SOFTWARE:

Jupyter Notebook (Anaconda 3) – Python 3.7.6

Microsoft Excel 2010

LIBRARIES:

LOADING LIBRARIES

```python
In [1]: # Let's Import all the required libraries

        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
        import seaborn as sns


        import nltk, re
        import string
        from nltk.corpus import stopwords
        from collections import Counter
        from nltk.corpus import stopwords
        from wordcloud import WordCloud
        from nltk.stem.porter import PorterStemmer
        from sklearn.feature_extraction.text import CountVectorizer
        from nltk.stem import WordNetLemmatizer, SnowballStemmer
        from sklearn.feature_extraction.text import TfidfVectorizer
```

```python
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.ensemble import AdaBoostClassifier
        from sklearn.ensemble import BaggingClassifier
        from sklearn.model_selection import GridSearchCV

        from sklearn.metrics import precision_score, recall_score, f1_score
        from sklearn.metrics import roc_curve,roc_auc_score,auc
        from sklearn.metrics import accuracy_score, confusion_matrix,classifi
        from sklearn.model_selection import cross_val_score

        import warnings
        warnings.filterwarnings('ignore')
        warnings.simplefilter("ignore")
        warnings.warn("deprecated", DeprecationWarning)
```

MODEL/S DEVELOPMENT AND EVALUATION

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES(METHODS)

As the target column was Bivariant data and the algorithm that we choose dependson this target variable. So, we have chosen classification analysis for this project.

TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

We have used the following algorithms

- o LogisticRegression()
- o DecisionTreeClassifier ()
- o KneighbourClassifier()
- o RandomForestClassifier ()
- o AdaBoostClassifier()
- o MultinomialNB()

```
In [39]: # Let's create a for loop function for our model

models=[]
models.append(('LogisticRegression',LR))
models.append(('DecisionTreeClassifier',DT))
models.append(('KneighborsClassifier',KNN))
models.append(('RandomForestClassifier',RF))
models.append(('AdaBoostClassifier',AD))
models.append(('MultinomialNB',MNB))
```

RUN AND EVALUATE SELECTED MODELS

```
In [40]: model_list=[]
         score=[]
         cvs=[]
         rocscore=[]

         for name, model in models:
             print(name,'Model :-',end='\n\n')

             model_list.append(name)
             model.fit(x_train,y_train)
             print(model,end='\n\n')
             pre=model.predict(x_test)
             print('\n')
             AS=accuracy_score(y_test,pre)
             print('Accuracy score =',AS)
             score.append(AS*100)
             print('\n')
             sc=cross_val_score(model,x,y, cv=10, scoring='accuracy').mean()
             print('cross validation score =',sc)
             cvs.append(sc*100)
             print('\n')
             false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
             roc_auc=auc(false_positive_rate,true_positive_rate)
             print('roc_auc_score = ', roc_auc)
             rocscore.append(roc_auc*100)
             print('\n')
             print('classification_report\n',classification_report(y_test,pre))
             print('\n')
             cm=confusion_matrix(y_test,pre)
             print(cm)
             print('\n')
             plt.figure(figsize=(10,40))
             plt.subplot(911)
             plt.title(name)
```

```
             cvs.append(sc*100)
             print('\n')
             false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
             roc_auc=auc(false_positive_rate,true_positive_rate)
             print('roc_auc_score = ', roc_auc)
             rocscore.append(roc_auc*100)
             print('\n')
             print('classification_report\n',classification_report(y_test,pre))
             print('\n')
             cm=confusion_matrix(y_test,pre)
             print(cm)
             print('\n')
             plt.figure(figsize=(10,40))
             plt.subplot(911)
             plt.title(name)
             print(sns.heatmap(cm,annot=True))
             plt.subplot(912)
             plt.title(name)
             plt.plot(false_positive_rate,true_positive_rate, label='AUC= %0.2f'%roc_auc)
             plt.plot([0,1],[0,1],'r--')
             plt.legend(loc='lower right')
             plt.ylabel('True positive rate')
             plt.xlabel('False positive rate')
             print('\n\n')
```

```
LogisticRegression Model :-

LogisticRegression()


Accuracy score = 0.9475138121546961


cross validation score = 0.9533301515332298


roc_auc_score =  0.866051773965443


classification_report
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       585
           1       0.99      0.73      0.84       139

    accuracy                           0.95       724
   macro avg       0.97      0.87      0.91       724
weighted avg       0.95      0.95      0.94       724


[[584   1]
 [ 37 102]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```
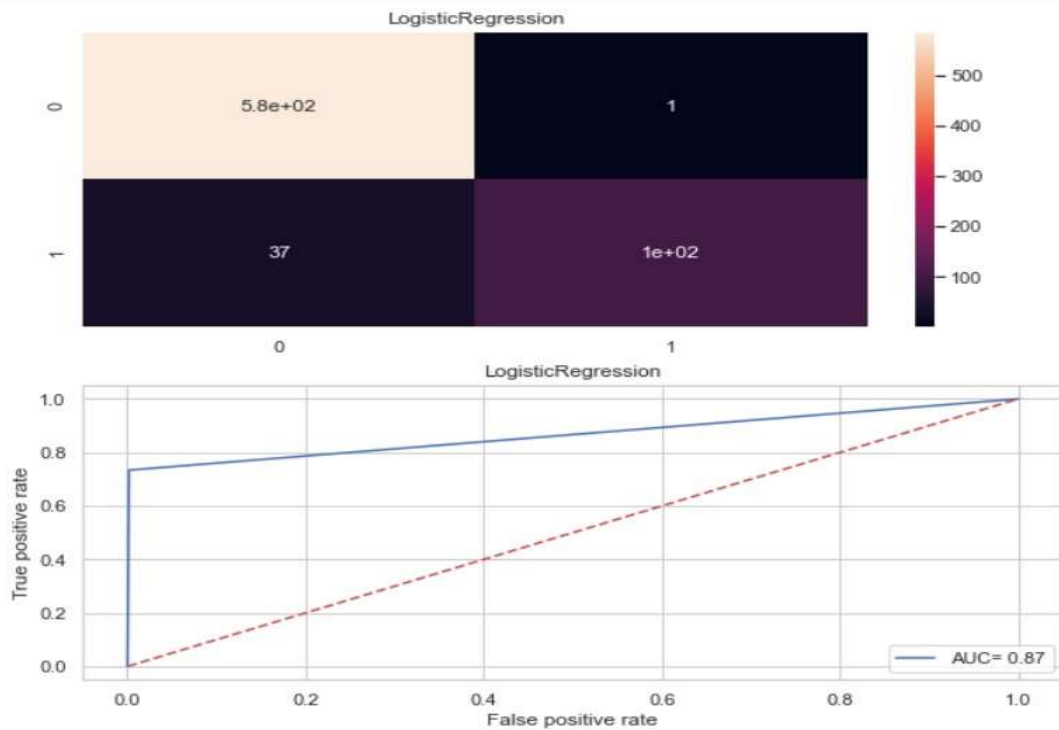
```
DecisionTreeClassifier Model :-

DecisionTreeClassifier(criterion='entropy')


Accuracy score = 0.9668508287292817


cross validation score = 0.9661245674740483


roc_auc_score =  0.9548053864600627


classification_report
              precision    recall  f1-score   support

           0       0.98      0.97      0.98       585
           1       0.90      0.94      0.92       139

    accuracy                           0.97       724
   macro avg       0.94      0.95      0.95       724
weighted avg       0.97      0.97      0.97       724


[[570  15]
 [  9 130]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```
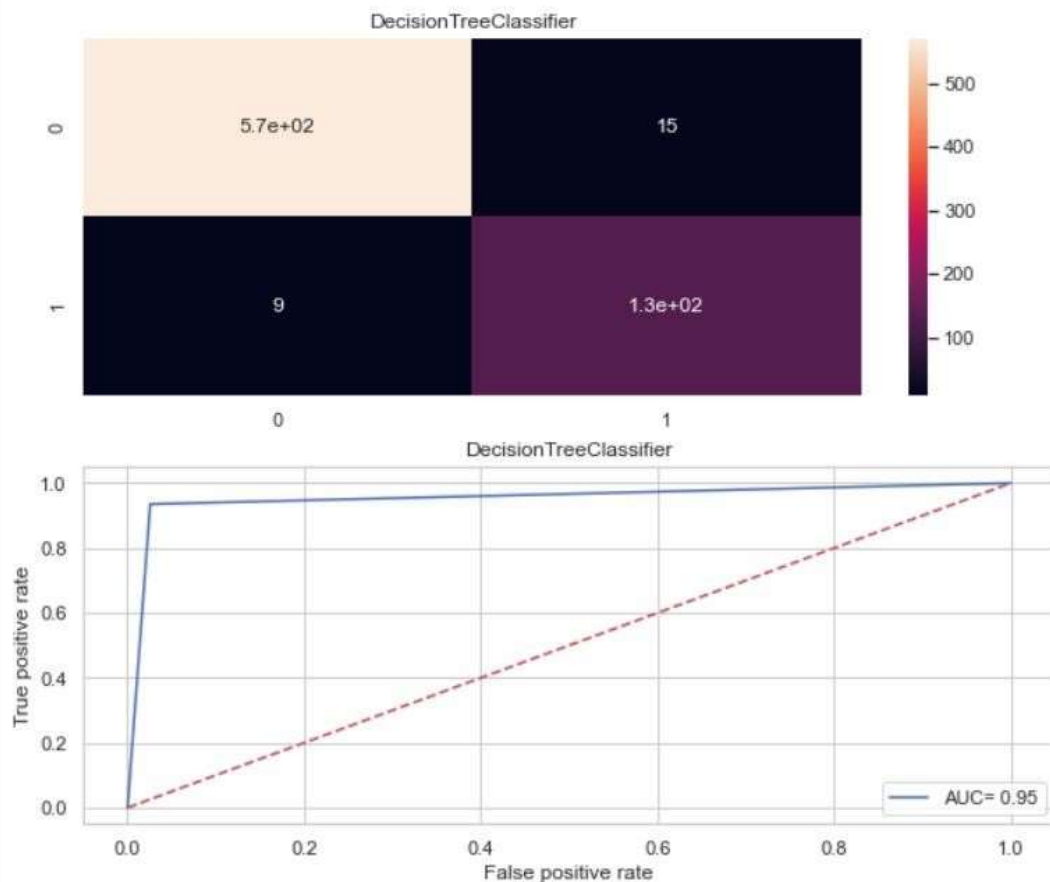
```
KneighborsClassifier Model :-

KNeighborsClassifier(n_neighbors=1)


Accuracy score = 0.9765193370165746


cross validation score = 0.9685431332776518


roc_auc_score =  0.969015556785341


classification_report
              precision    recall   f1-score    support

          0      0.99       0.98       0.99        585
          1      0.92       0.96       0.94        139

   accuracy                            0.98        724
  macro avg      0.96       0.97       0.96        724
weighted avg     0.98       0.98       0.98        724


[[574  11]
 [  6 133]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```
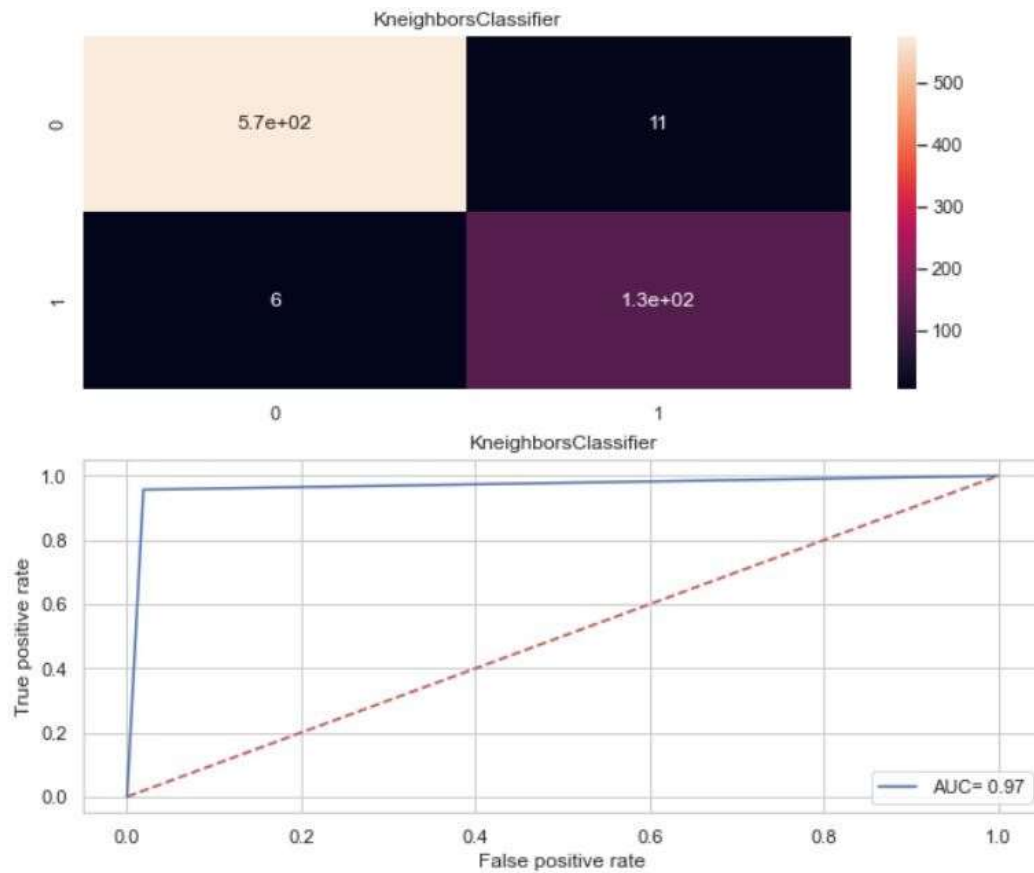
```
RandomForestClassifier Model :-

RandomForestClassifier(random_state=42)


Accuracy score = 0.9751381215469613


cross validation score = 0.9733874239350913


roc_auc_score =  0.935251798561151


classification_report
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       585
           1       1.00      0.87      0.93       139

    accuracy                           0.98       724
   macro avg       0.99      0.94      0.96       724
weighted avg       0.98      0.98      0.97       724


[[585   0]
 [ 18 121]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```


RandomForestClassifier

RandomForestClassifier
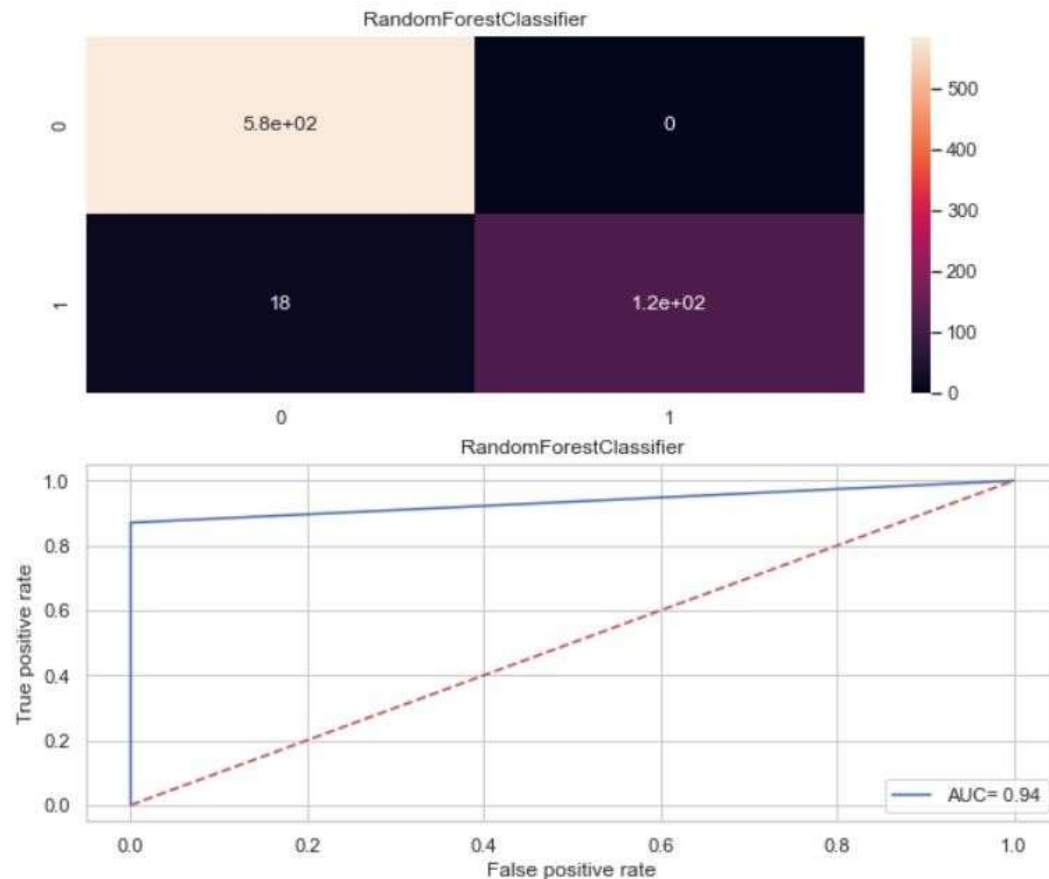
```
AdaBoostClassifier Model :-

AdaBoostClassifier()


Accuracy score = 0.988950276243094


cross validation score = 0.9820200453406513


roc_auc_score =  0.9794502859251061


classification_report
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       585
           1       0.98      0.96      0.97       139

    accuracy                           0.99       724
   macro avg       0.98      0.98      0.98       724
weighted avg       0.99      0.99      0.99       724


[[582    3]
 [   5 134]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```


AdaBoostClassifier

AdaBoostClassifier
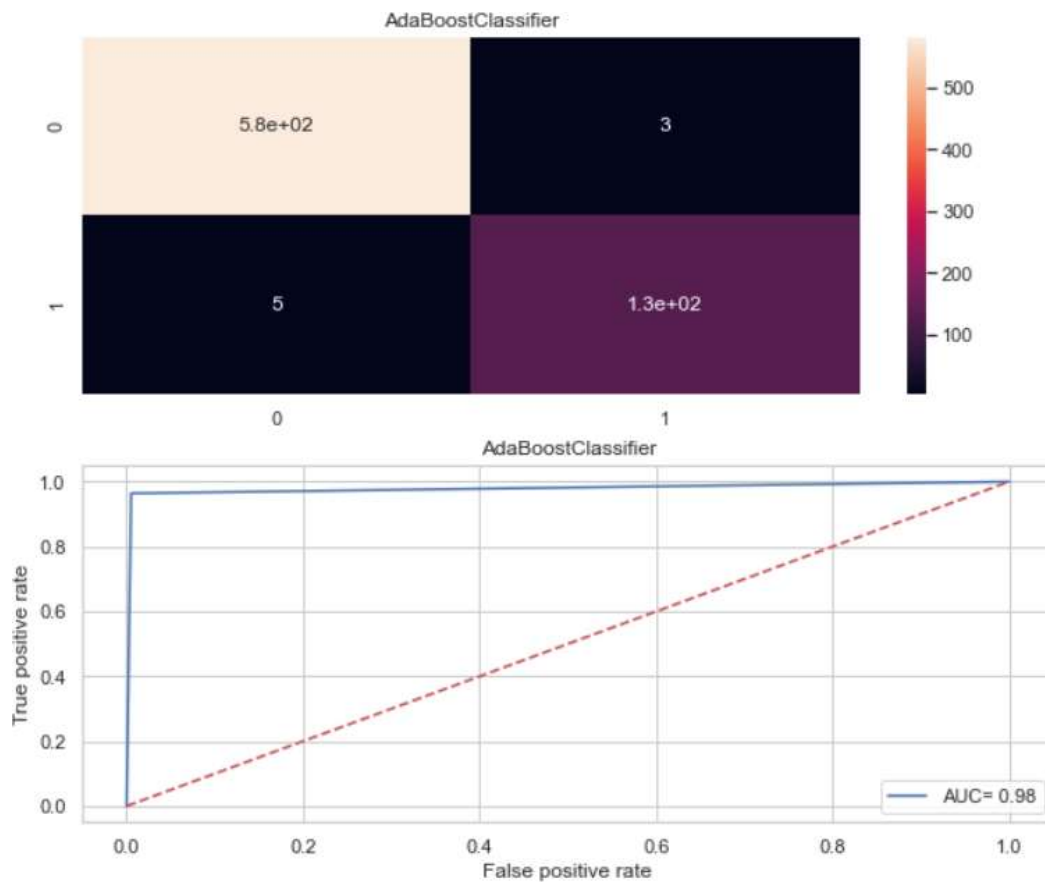
```
MultinomialNB Model :-

MultinomialNB()


Accuracy score = 0.8342541436464088


cross validation score = 0.8606944278725688


roc_auc_score =  0.5683453237410072


classification_report
              precision    recall  f1-score   support

           0       0.83      1.00      0.91       585
           1       1.00      0.14      0.24       139

    accuracy                           0.83       724
   macro avg       0.91      0.57      0.57       724
weighted avg       0.86      0.83      0.78       724


[[585    0]
 [120   19]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)
```
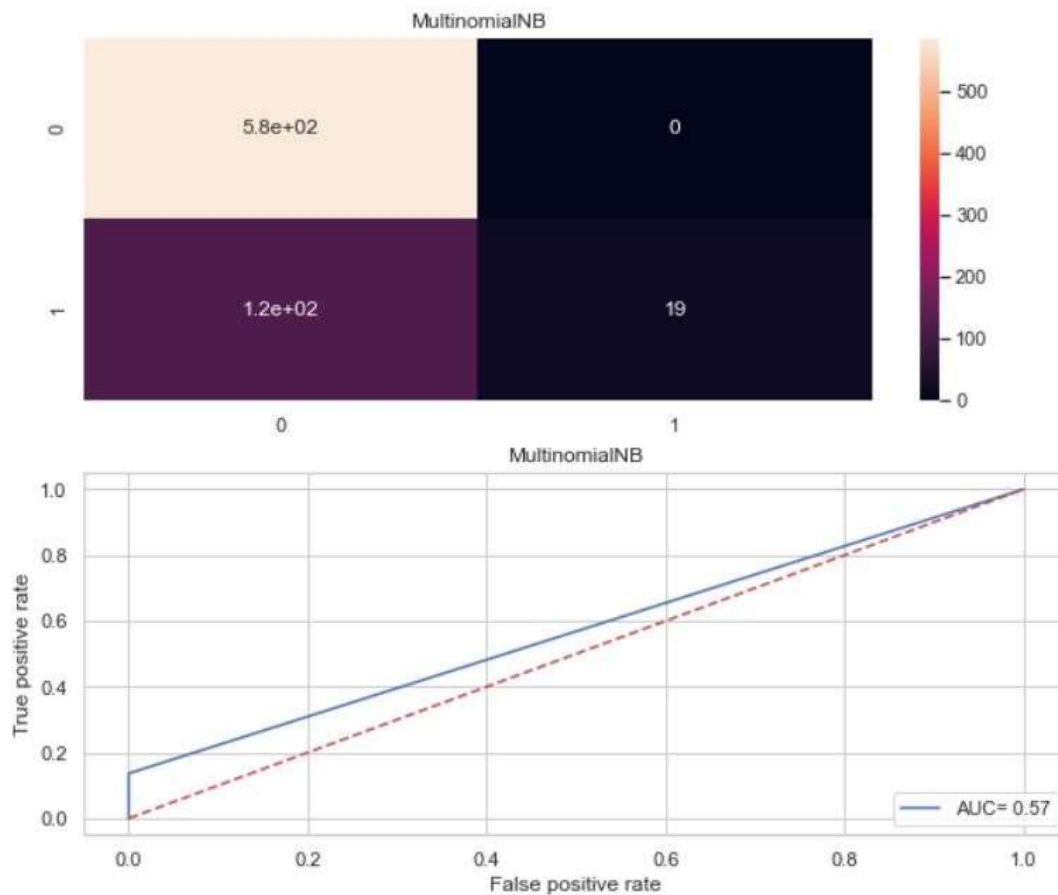
KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDERCONSIDERATION

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

```
In [41]: # Let's create a Dataframe for our above models

         result=pd.DataFrame({'Model': model_list, 'Accuracy_score': score, 'Cross_val_sc
         result
```

Out[41]:

	Model	Accuracy_score	Cross_val_score	Roc_auc_score
0	LogisticRegression	94.751381	95.333015	86.605177
1	DecisionTreeClassifier	96.685083	96.612457	95.480539
2	KneighborsClassifier	97.651934	96.854313	96.901556
3	RandomForestClassifier	97.513812	97.338742	93.525180
4	AdaBoostClassifier	98.895028	98.202005	97.945029
5	MultinomialNB	83.425414	86.069443	56.834532

VISUALIZATIONS

```
In [9]: #Let's plot a pie chart to visualize the Spam and non spam mails

label=email['label'].value_counts().index.tolist()
value=email['label'].value_counts().values.tolist()
explode=(0.030,0)
clour=('green','red')
plt.figure(figsize=(8,5),dpi=100)
sns.set_context('talk',font_scale=0.2)
sns.set(style='whitegrid')
plt.pie(x=value,explode=explode,labels=label,colors=clour,autopct='%2.0f%%',pctd
plt.legend(["0 = No Spam",'1 = Spam'])
plt.show()
```
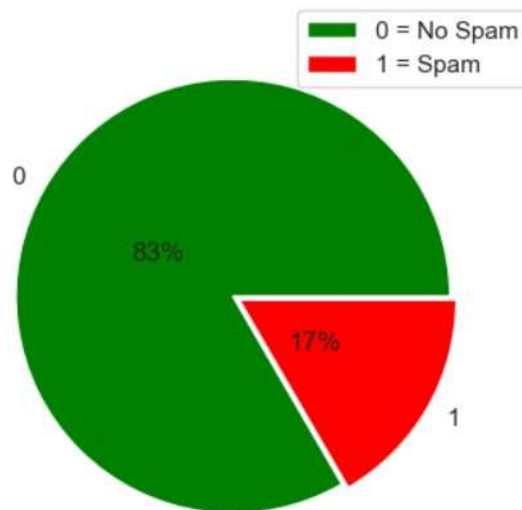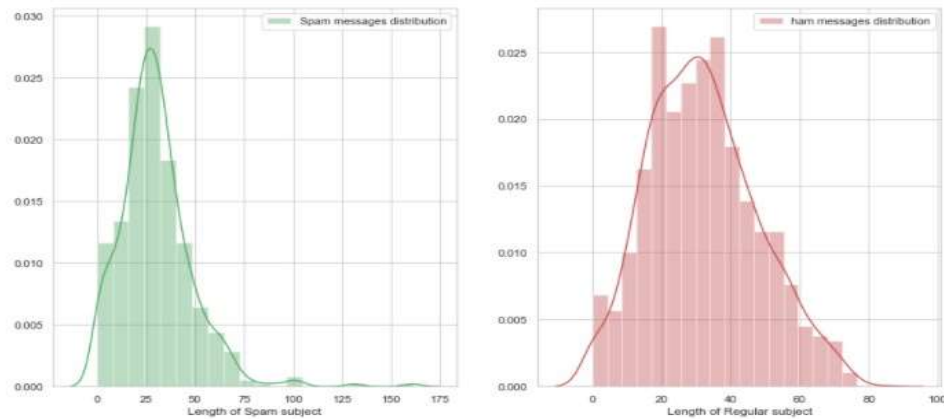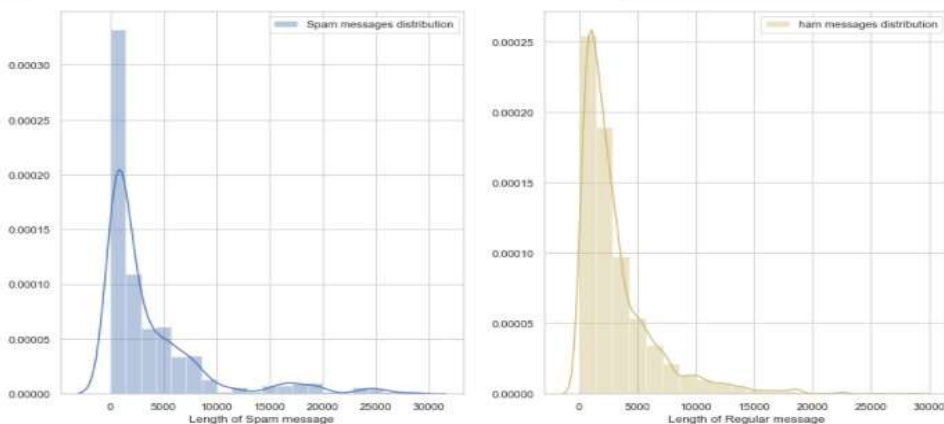
```
In [26]:  # Let's count the distribution of words before cleaning of Subject column

          f,ax = plt.subplots(1,2,figsize = (15,8))

          sns.distplot(strings[strings['label']==1]['Subject_length'],bins=20,ax=ax[0],lab

          ax[0].set_xlabel('Length of Spam subject')
          ax[0].legend()

          sns.distplot(strings[strings['label']==0]['Subject_length'],bins=20,ax=ax[1],lab
          ax[1].set_xlabel('Length of Regular subject')
          ax[1].legend()

          plt.show()
```
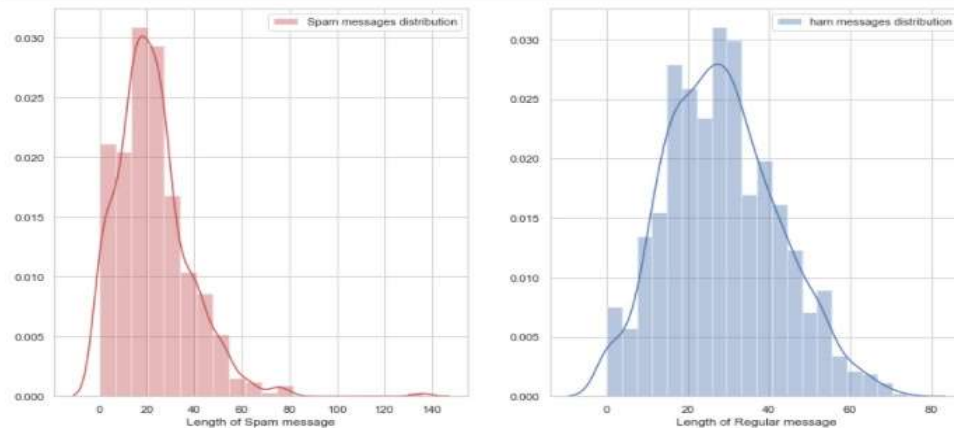


```
In [27]:  # Let's count the distribution of words before cleaning of message column

          f,ax = plt.subplots(1,2,figsize = (15,8))

          sns.distplot(strings[strings['label']==1]['Message_length'],bins=20,ax=ax[0],lab

          ax[0].set_xlabel('Length of Spam message')
          ax[0].legend()

          sns.distplot(strings[strings['label']==0]['Message_length'],bins=20,ax=ax[1],lab
          ax[1].set_xlabel('Length of Regular message')
          ax[1].legend()

          plt.show()
```

In [28]:
```python
# Let's count the distribution of words after cleaning of subject column

f,ax = plt.subplots(1,2,figsize = (15,8))

sns.distplot(strings[strings['label']==1]['Subject_clean_length'],bins=20,ax=ax[
ax[0].set_xlabel('Length of Spam message')
ax[0].legend()

sns.distplot(strings[strings['label']==0]['Subject_clean_length'],bins=20,ax=ax[
ax[1].set_xlabel('Length of Regular message')
ax[1].legend()

plt.show()
```


In [29]:
```python
# Let's count the distribution of words after cleaning of message column

f,ax = plt.subplots(1,2,figsize = (15,8))

sns.distplot(strings[strings['label']==1]['Message_clean_length'],bins=20,ax=ax[
ax[0].set_xlabel('Length of Spam message')
ax[0].legend()

sns.distplot(strings[strings['label']==0]['Message_clean_length'],bins=20,ax=ax[
ax[1].set_xlabel('Length of Regular message')
ax[1].legend()

plt.show()
```
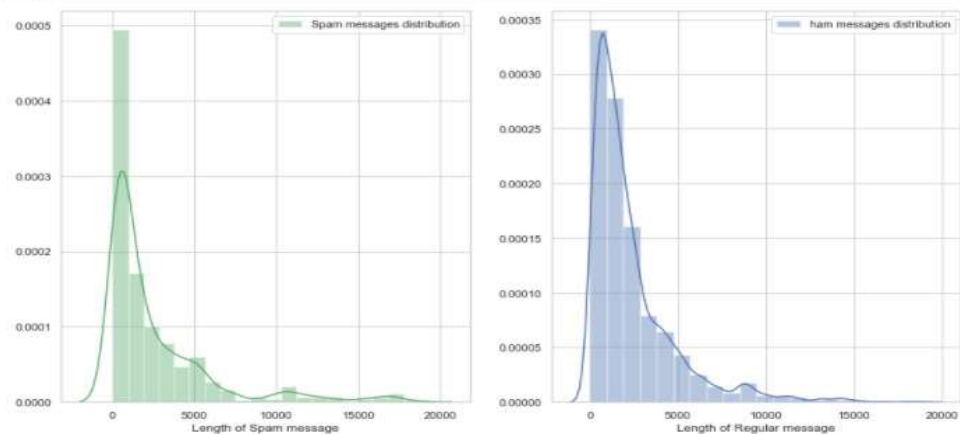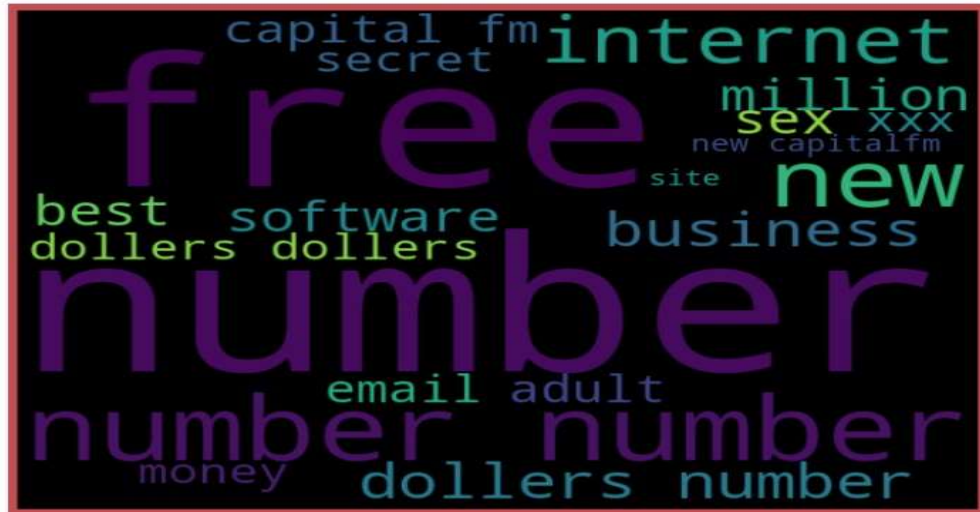
```
In [30]:  # Let's plot the loud words in spam for subject column

          spams = strings['subject'][strings['label']==1]

          spam_cloud = WordCloud(width=700,height=500,background_color='black',max_words=2(

          plt.figure(figsize=(10,8),facecolor='r')
          plt.imshow(spam_cloud)
          plt.axis('off')
          plt.tight_layout(pad=0)
          plt.show()
```
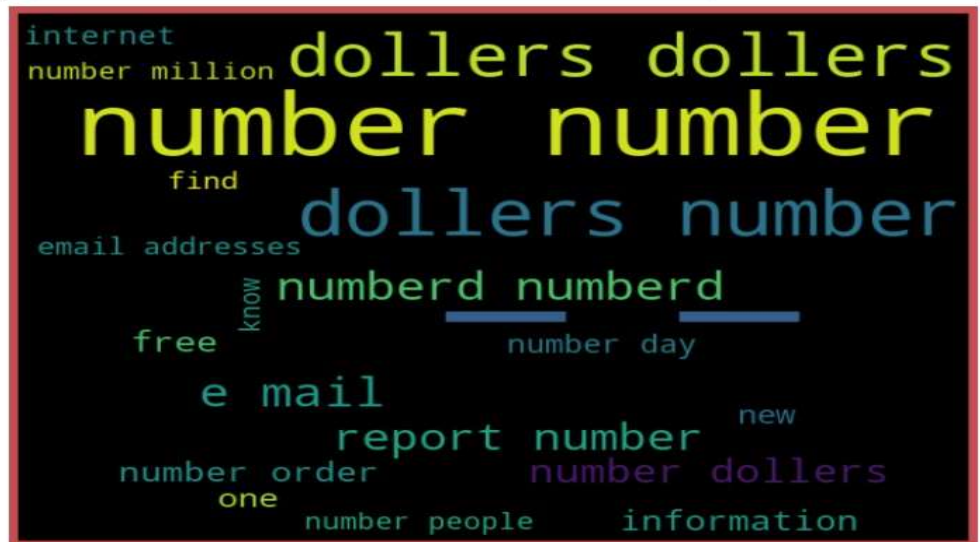


```
In [31]:  # Let's plot the loud words in spam for message column

          spams = strings['message'][strings['label']==1]

          spam_cloud = WordCloud(width=700,height=500,background_color='black',max_words=2(

          plt.figure(figsize=(10,8),facecolor='r')
          plt.imshow(spam_cloud)
          plt.axis('off')
          plt.tight_layout(pad=0)
          plt.show()
```
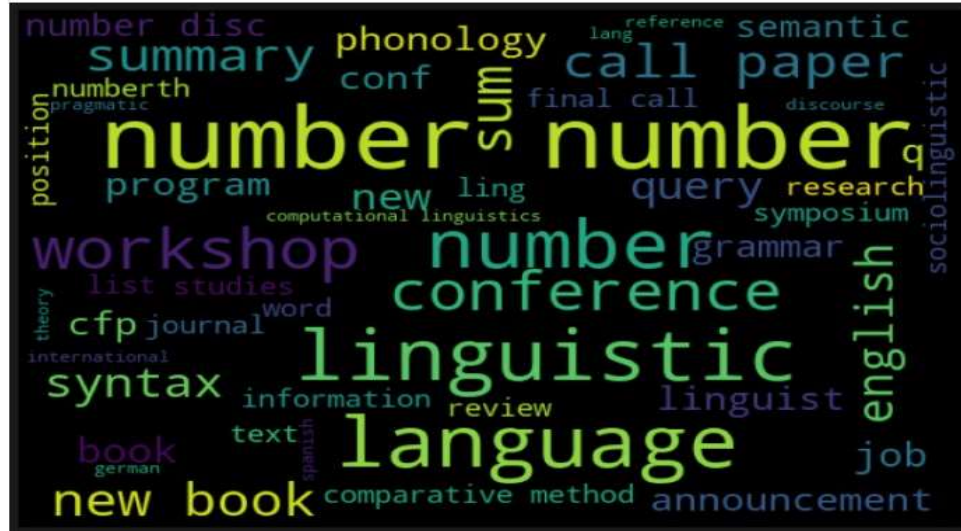
In [32]:
```python
# Let's plo the loud words in regular emails for subject column

regular = strings['subject'][strings['label']==0]
regular_cloud = WordCloud(width=600,height=400,background_color='black',max_word
plt.figure(figsize=(10,8),facecolor='k')
plt.imshow(regular_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```
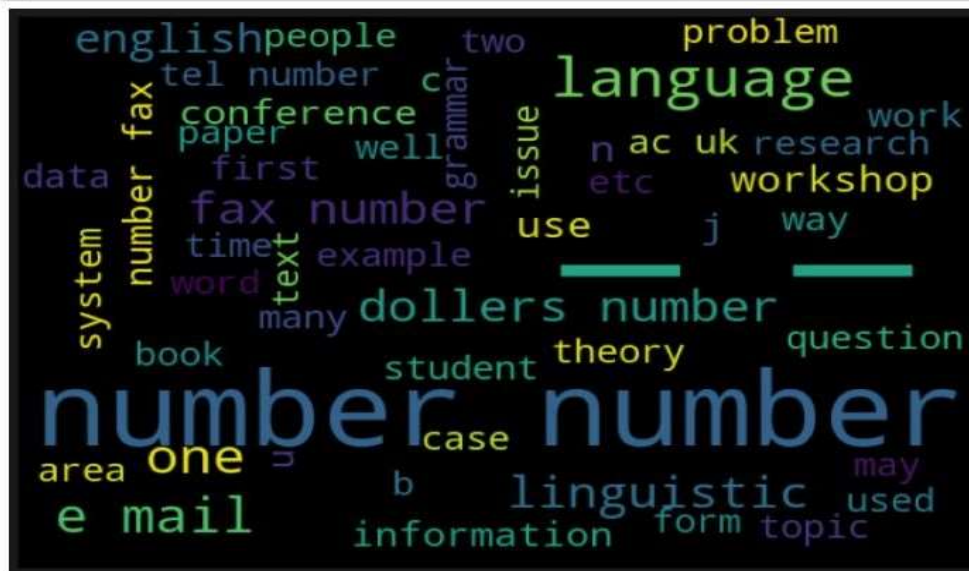

In [33]:
```python
# Let's plot the loud words in regular emails for message column

regular = strings['message'][strings['label']==0]
regular_cloud = WordCloud(width=600,height=400,background_color='black',max_word
plt.figure(figsize=(10,8),facecolor='k')
plt.imshow(regular_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

```
In [41]:  # Let's create a Dataframe for our above models

          result=pd.DataFrame({'Model': model_list, 'Accuracy_score': score, 'Cross_val_sc
          result
```

Out[41]:

	Model	Accuracy_score	Cross_val_score	Roc_auc_score
0	LogisticRegression	94.751381	95.333015	86.605177
1	DecisionTreeClassifier	96.685083	96.612457	95.480539
2	KneighborsClassifier	97.651934	96.854313	96.901556
3	RandomForestClassifier	97.513812	97.338742	93.525180
4	AdaBoostClassifier	98.895028	98.202005	97.945029
5	MultinomialNB	83.425414	86.069443	56.834532

PERFORMANCE EVALUATION USING MULTIPLE METRICS ¶

After comparing all the models I have choosen the AdaBoostClassifier because it has the highest scores.

Accuracy score = 98.89%

cross validation score = 98.20%

roc_auc_score = 97.94%

classification_report

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       585
           1       0.98      0.96      0.97       139
```

accuracy

```
                                        0.99       724
```

macro avg

```
                 0.98      0.98      0.98       724
```

```
weighted avg

                 0.99        0.99        0.99        724
```

Confusion matrix:

```
[582    3]
[  5 134]
```

It only classified 8 out of 716 entries incorrectly.

We ran the results for the "Subject" and "Message" columns separately but we got the same results as when they are combined. So, we have included the analysis of both columns combined.

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'sperformance on identification of Spam mails.