Email Spam Detection Project

Submitted by:

KHUSHBOO KHATRI

ACKNOWLEDGMENT

The internship opportunity I had with FlipRobo Technologies was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it.

I would also like to appreciate Datatrained for provided with right training skills to deal with current problem statement.

INTRODUCTION

- ## 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.

  Here we need to build an NLP Model to classify the messages as HAM or Spam.

- ## Conceptual Background of the Domain Problem

  What is a Spam Filtering?

  Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the nonspam texts. It uses a binary type of classification containing the labels such as '**ham'** (nonspam) and **spam**. Application of this can be seen in Google Mail (GMAIL) where it segregates the spam emails in order to prevent them from getting into the user's inbox.

  The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text.

  This corpus has been collected from free or free for research sources at the Internet:

  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.

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.

- Review of Literature

Spam detection is one of the important Application of NLP, where it is used to detect unwanted e-mails getting to a user's inbox. NLP consists of various applications, like **speech recognition, machine translation,** and **machine text reading**. When we combine all these applications then it allows the artificial intelligence to gain knowledge of the world. Let's consider the example of AMAZON ALEXA, using this robot you can ask the question to Alexa, and it will reply to you.

NLP helps users to ask questions about any subject and get a direct response within seconds. NLP offers exact answers to the question means it does not offer unnecessary and unwanted information. NLP helps computers to communicate with humans in their languages. It is very time efficient. Most of the companies use NLP to improve the

efficiency of documentation processes, accuracy of documentation, and identify the information from large databases.

Below are some of the most popular machine learning methods:

a) Naïve Bayes classifier: It is a supervised machine learning algorithm where words probabilities play the main rule here. If some words occur often in spam but not in ham, then this incoming e-mail is probably spam. Naïve bayes classifier technique has become a very popular method in mail filtering software. Bayesian filter should be trained to work effectively. Every word has certain probability of occurring in spam or ham email in its database. If the total of words probabilities exceeds a certain limit, the filter will mark the e-mail to either category.

b) Artificial Neural Networks classifier: An artificial neural network (ANN), also called simply a "Neural Network" (NN), is a computational model based on biological neural networks. It consists of an interconnected collection of artificial neurons. An artificial neural network is an adaptive system that changes its structure based on information that flows through the artificial network during a learning phase.

- Motivation for the Problem Undertaken

Email Spam Detection Project helps us to understands the basics of NPL model. We see some common NLP tasks that one can perform easily and how one can complete an end-to-end project. **Natural Language Processing (NLP)** is the art and science which helps us

extract information from the text and use it in our computations and algorithms.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

We are provided with a dataset consisting of text column and a label column. We read the data, and here is the statistical analysis of the dataset.

```
# statistical description of final dataset.
data.describe(include= object)
```

7]:

	v1	v2
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

From here it is clear that the dataset consist of 5572 rows, we include object here as our columns has text data in it.

Our label column v1 consist of two classes, Spam and Ham(authenticate mail/not spam). And v2 defines the feature column which consist of basically text data with was provided to us, mostly has all unique values.

```
# checking the classes distribution in label column
data['v1'].value_counts()
```

]:
```
ham     4825
spam     747
Name: v1, dtype: int64
```

We have total number of 4825 data with are not spam, while 747 data is spam text. Which indicate sure imbalance among the classes of label.

- Data Sources and their formats

We get the csv file for our model building. In that file we got two columns v1 and v2, where v1 represents the label column containing ham and spam, while v2 consist of one line message which is predicted as ham or spam.

We get to know through description that this corpus has been collected from free or free for research sources at the Internet.

```
# getting the data
df= pd.read_csv(r'C:\Users\DELL\Documents\spam.csv', encoding='latin-1')
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

The Unnamed columns are the result of adding additional parameters while calling the dataset, so we will drop them and keep only v1 and v2. As these are the required columns for model building. The final dataset will look like this:

```
# dropping the unwanted columns
data= df.drop(columns= ['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis= 1)
data.head()
```

]:

	v1	v2
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...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

The dimension for final dataset is known by using shape function, where first element shows the number of rows while second feature shows number of columns.

```
# checking the dimension of our final dataset.
data.shape
```

]: (5572, 2)

- Data Preprocessing Done

In this step, we explore data and gain insights such as shape, accuracy, and initial patterns in the data.

First, we check the information of the given dataset and extract information about the dataset. And we see that the dataset contains 5572 entries with no NaN values in the feature message & label.

We create a new feature named text_lenght to check the length of each text in v2 column.

```
# finding the lenght of v2 which is our text data
data['text_lenght']=data['v2'].str.len()
data.head(2)
```

	v1	v2	text_lenght
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29

Besides all these steps, now we do some common tasks that can be done on every NLP project.

We have to clean the data using regex, matching patterns in the e-mail messages, and replace them with more organized counterparts or just remove them. Cleaner data leads to a more efficient model and higher accuracy. Following steps are involved in pre-processing the messages:

1. We start with importing all the necessary libraries.

```
# importing NLP libraries
from IPython.display import display, HTML
display(HTML("<style>.container {width: 100% !important; }</style>"))

import re

import string
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

2. We define a function ‘deconstructed’ to expand all the English language contractions.

```
#expanding english language contractions:
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", "s", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    phrase = re.sub(r"\'u", " you", phrase)

    return phrase
```

3. Defining the stopwords and lemmatizer.

```
# defining stop words and lemmatizer
stop_words=set(stopwords.words('english')+ ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure', 'dun'])
lemmatizer= WordNetLemmatizer()
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
True
```

4. Then we pass our text data to this function, then we clean the data by remove all the hyperlinks, emails, numbers, punctuations, whitespaces, stop_words, special characters.
5. Then we perform lemmatization, and convert the entire text to lower case.
6. We perform above two steps on our text feature column and store the data into list.

```
preprocessed_text=[]
from tqdm import tqdm
#tqdm is for printing the status bar
for sentance in tqdm(data['v2'].values):
    sent= decontracted(sentance)
    sent = re.sub(r'https?:\/\/.*[\r\n]*', '', sent)    #remove hyperlinks
    sent = re.sub(r'^.+@[^\.].*\.[a-z]{2,}$','', sent)    # remove emails
    sent = re.sub(r'[^\w\d\s]', '',sent )      # removing punctuation
    sent = re.sub('[^A-Za-z]+', ' ', sent)      #remove special characters, numbers
    sent = ' '.join(e for e in sent.split() if e not in stop_words)  # removing stop words
    sent = ' '.join(lemmatizer.lemmatize(e) for e in sent.split())   # lemmatization
    preprocessed_text.append(sent.lower().strip()) # converting the text to lower case
```

```
100%|██████████| 5572/5572 [00:00<00:00, 13441.02it/s]
```

7. We assigned this processed_data list to the feature column v2 of our dataset.

```
data['v2']= preprocessed_text    # assigning the clean value to the dataset
```

8. Then we check the length of clean data, so that we get a clear idea about how much our data is processed.

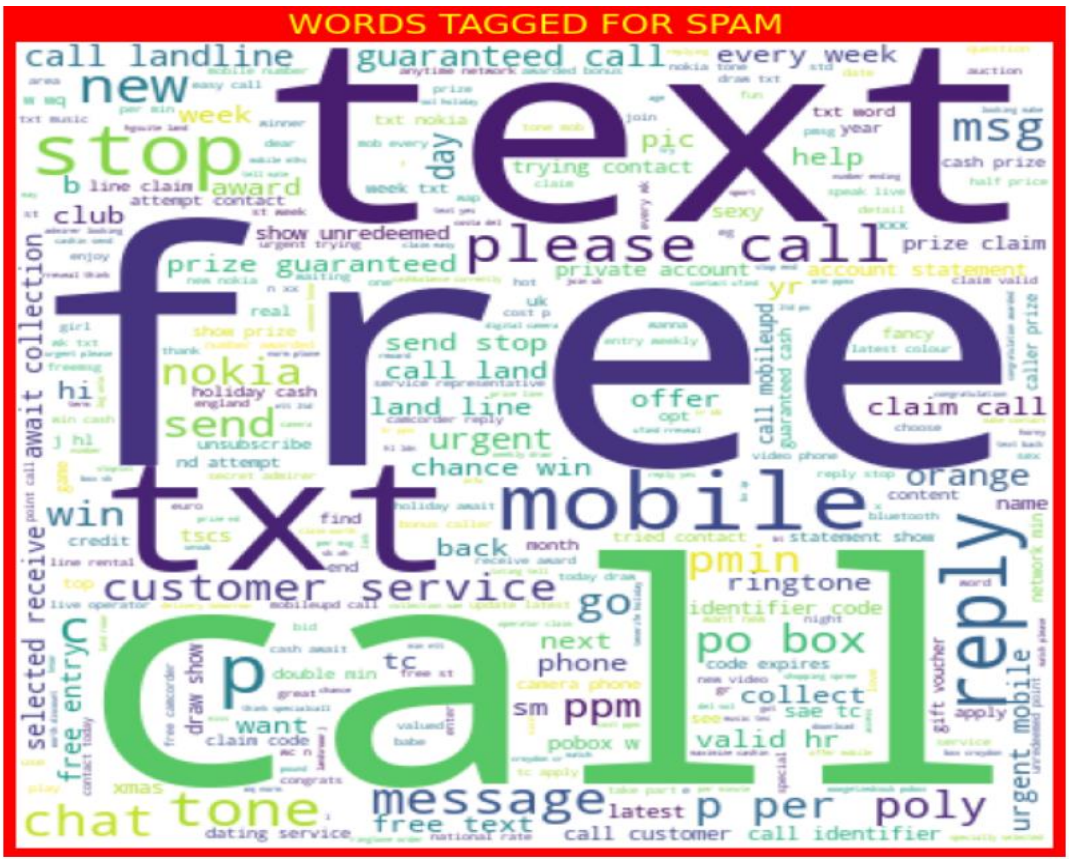
```
#checking clean length
data["clean_len"] = data["v2"].str.len()
```

```
data.head()
```

	v1	v2	text_lenght	clean_len
0	ham	go jurong point crazy available bugis n great ...	111	82
1	ham	ok lar joking wif oni	29	21
2	spam	free entry wkly comp win fa cup final tkts st ...	155	101
3	ham	say early hor c already say	49	27
4	ham	nah think go usf life around though	61	35

- Data Inputs- Logic- Output Relationships

For defining the input out logic we plot word cloud for the words we are commonly seen in ham message, and a word cloud consist of common words used in spam meassages.



These are top 400 words which are generally seen in a spam message. The larger font words(free, text, call and so on) are those which are most occurred words in spam message.

These are set of words which generally find in a ham(not spam) messages. Here also the large fonts denotes frequently appearing words in the ham text.

- State the set of assumptions (if any) related to the problem under consideration

  We are considering that the data provided to us is collected properly, and we are building an algorithm which perfectly defines and predicts in the current dataset.

- Hardware and Software Requirements and Tools Used
  1. Python
  2. Pandas
  3. Matplotlib.pyplot
  4. Warnings
  5. Numpy
  6. Seaborn
  7. Re
  8. String
  9. Nltk
  10. Stopwords
  11. WordNetLemmatizer
  12. Wordnet
  13. Tqdm
  14. Wordcloud
  15. Tfidfvectorizer

  Here we build a classification model using classification algorithms.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  We perform tfidf vectorization in our feature column, to vectorized the data, then we use that vectorized data in our ensemble classification approach (in our case its RandomForest Classifier) to build the final model.

- Testing of Identified Approaches (Algorithms)

As we know that this NLP problem is a classification problem, so we follow following classification approaches to predict the best result for our dataset.

1. Logistic Regression

2. Naive Bayes (MultinomialNB)
3. Decision Tree classifier
4. RandomForest Classifier
5. Support Vector Matrix (SVC)

- Run and Evaluate selected models

  Our final model is RandomForest, with tfidf vectorization

```
# 1. Convert text into vectors using TF-IDF
# 2. Split feature and label

from sklearn.feature_extraction.text import TfidfVectorizer
tf_vec = TfidfVectorizer()
features = tf_vec.fit_transform(data['v2'])

x = features


# we also need to encode label column
data['v1'].replace(to_replace='spam', value=1, inplace=True)
data['v1'].replace(to_replace='ham', value=0, inplace=True)

# defining label as y
y= data['v1']
```

As our dataset is not balanced so we use smote technique to balance our dataset, so that we don't get biased results.

```
#handling imbalanced data through smote technique
from imblearn.combine import SMOTETomek
from collections import Counter
os=SMOTETomek(sampling_strategy = {0: 5572, 1: 5572})
x1,y1=os.fit_resample(x,y)
print("The number of classes before fit{}".format(Counter(y)))
print("The number of classes after fit {}".format(Counter(y1)))
```

```
The number of classes before fitCounter({0: 4825, 1: 747})
The number of classes after fit Counter({0: 5572, 1: 5572})
```

Now ,We split the data into train and test using train_test_split.

```
# splitting data into test and train sets
x_train,x_test,y_train,y_test = train_test_split(x1,y1,test_size=.30)
```

Our final selected model is RandomForest Classifier.

```
# Model
rf= RandomForestClassifier()

#fit
rf.fit(x_train,y_train)

#predict
y_pred=rf.predict(x_test)

print("\nClassification report :\n",classification_report(y_test,y_pred))
print("\n----------------------------------------------------------------\n")
print("\n --------------------Scores for metrics-----------------\n")
a_rf=accuracy_score(y_test,y_pred)
c_rf=cross_val_score(rf,x1,y1,cv=3).mean()
f_rf=f1_score(y_pred,y_test,average="weighted")
print("accuracy_score : ",a_rf,"\n","cross validation score :",c_rf,"\nF1 Score :",f_rf)
auc_roc_rf= roc_auc_score(y_test,y_pred)
print(" Auc ROC score is : ", auc_roc_rf)
rf_logloss= log_loss(y_test, y_pred)
print("Log Loss is : ", rf_logloss)

print("\n\n----------------------------------------------------------------\n\n")
conf_mat=confusion_matrix(y_test,y_pred)
print("\n CONFUSION MATRIX \n",conf_mat)
```

The outcome is as follows:

```
Classification report :
               precision    recall  f1-score   support

           0       0.99      1.00      0.99      1654
           1       1.00      0.99      0.99      1690

    accuracy                           0.99      3344
   macro avg       0.99      0.99      0.99      3344
weighted avg       0.99      0.99      0.99      3344


----------------------------------------------------------


 --------------------Scores for metrics-----------------

accuracy_score :  0.9946172248803827
 cross validation score : 0.9934493977536526
F1 Score : 0.9946170766138221
 Auc ROC score is :  0.9946616772679465
Log Loss is :  0.18591494446867704


----------------------------------------------------------------


 CONFUSION MATRIX
 [[1652    2]
 [  16 1674]]
```

We will plot the confusion matrix to get a proper visualization of the algorithm.

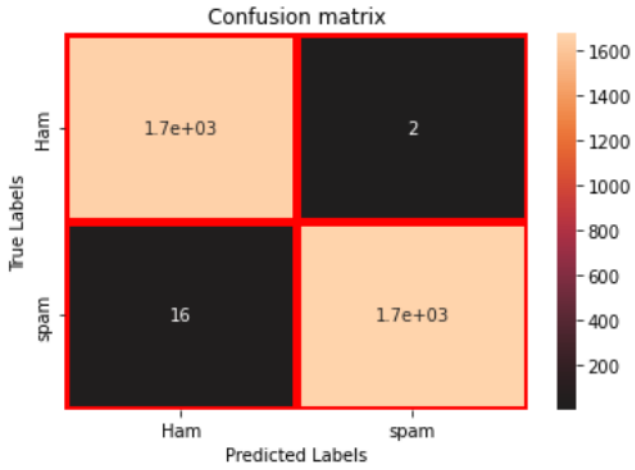
```
# plot confusion matrix heatmap
conf_mat = confusion_matrix(y_test,y_pred)

ax=plt.subplot()

sns.heatmap(conf_mat,annot=True,ax=ax,linewidths=5,linecolor='r',center=0)

ax.set_xlabel('Predicted Labels');ax.set_ylabel('True Labels')

ax.set_title('Confusion matrix')
ax.xaxis.set_ticklabels(['Ham','spam'])
ax.yaxis.set_ticklabels(['Ham','spam'])
plt.show()
```



- Key Metrics for success in solving problem under consideration

  Our primary metrics is accuracy and log_loss for the valuation of the model, the model with highest accuracy and lowest log_loss seems to the best fitted model for the dataset.

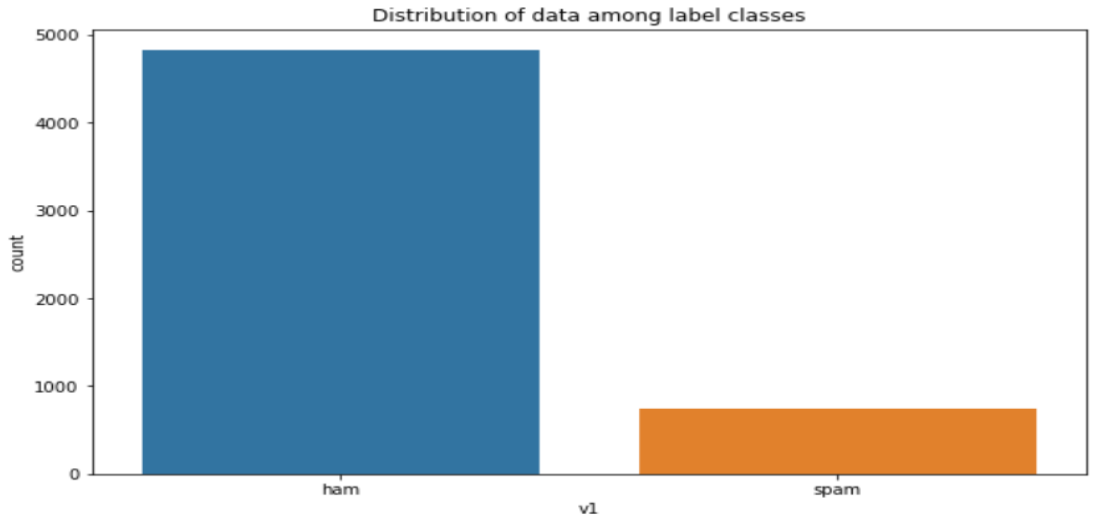
  While our secondary metrics is F1 score and ROC AUC score, highest values for both serves as best fitted model.

  We also perform cross validation for each model for checking the authenticity of the model.

- Visualizations

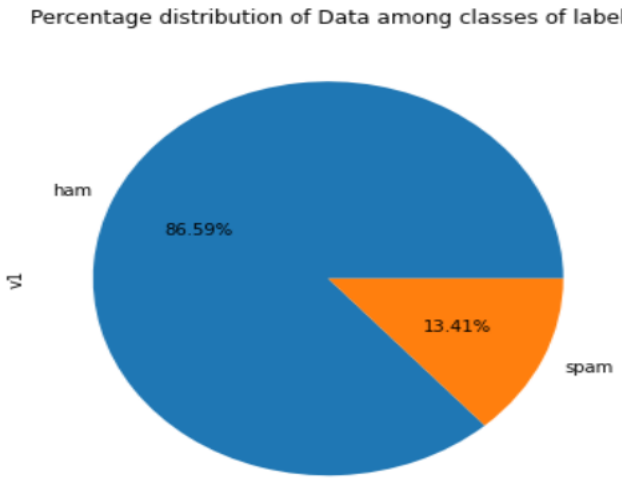
  Following the graphs which we plot during the EDA or visualization of data.

```
plt.figure(figsize= (10,6))
sns.countplot(data['v1'])
plt.title("Distribution of data among label classes")
plt.show()
```



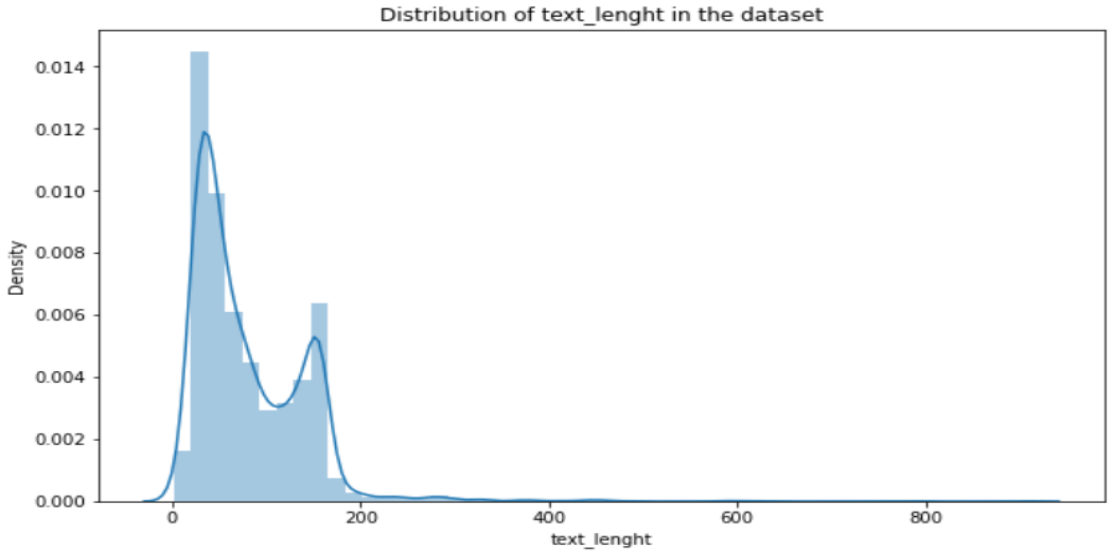
We can observe that we have maximum amount of data for ham than around 4825 while for spam we have 747 messages in total. We need to take care of this while building model.

```
plt.figure(figsize=(10,6))
(data['v1'].value_counts()*100.0 /len(data)).plot.pie(autopct='%.2f%%')
plt.title("Percentage distribution of Data among classes of label")
plt.show()
```



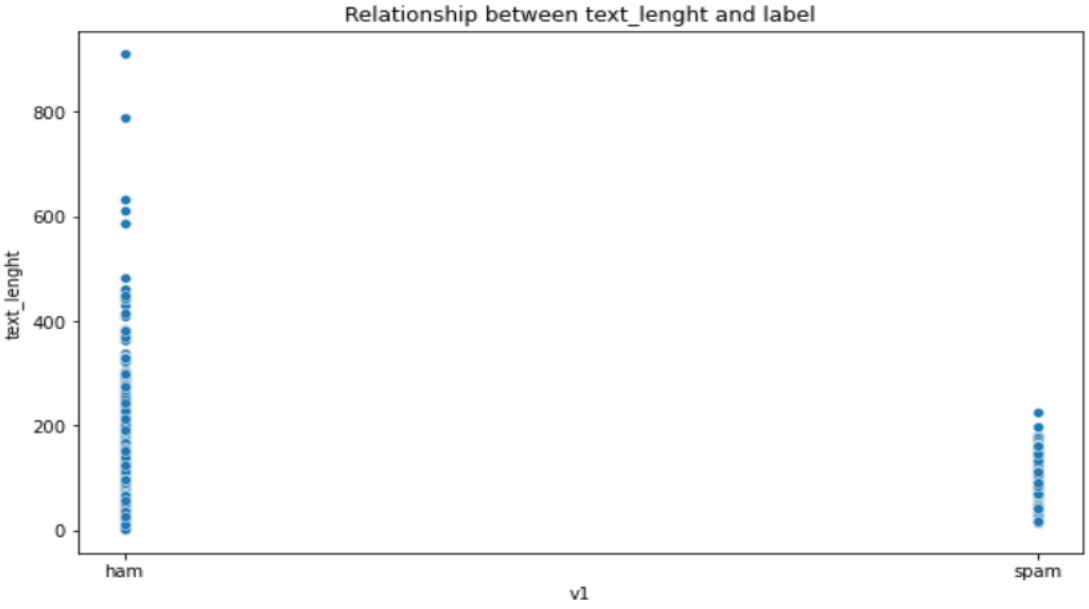
From here also we can see that around 86.59% of data is ham while 13.41% of data is spam. Above graph represents the percentage distribution of data among classes of labels.

```
plt.figure(figsize=(10,6))
sns.distplot(data['text_lenght'])
plt.title("Distribution of text_lenght in the dataset")
plt.show()
```



Above plot represents the text length of all the text feature, in the dataset. We observe that mostly the length of the text message is between 0-200. Again we observe couple larger text as well, but they are very less in number.

```
plt.figure(figsize=(10,6))
sns.scatterplot(x='v1', y= 'text_lenght', data= data)
plt.title("Relationship between text_lenght and label")
plt.show()
```



Its unique to see that mostly spam mail, is consist of usual text_lenght i.e 0-200 while ham can be a bigger text. However this can't be the deciding factor as most of ham also lies well within the word limit of 0-200.

- Interpretation of the Results

1. Done all the pre-processing steps to make data ready for model building.

2. Removed stop words and create another feature ‘clean_length’ for comparing cleaned and unprocessed message length.

3. Understood relationship and gain insights by using Data Visualization:

    a. Plotted count plot for checking spam and non-spam email counts.

    b. Lastly, visualize some popular terms in spam messages using the word cloud.

4. Used **Tf-idf vectorizer** to convert text into vector.

5. Found the best model as **RandomForest Classifier** which provides max accuracy of **99.46%.**

6. Found **high precision and recall score of 0.99.**

7. Confusion matrix shows **high classification accuracy** with only 16 out of 747 are incorrect.

8. Overall model fit is good.

CONCLUSION

- Key Findings and Conclusions of the Study

    With the current distribution of train data and test data, RandomForest seems to be the best fit for our model.

As our dataset consist of only two columns, first is v1 which is our label column and the second is v2 which is our feature text column.

In our case the label column is also in categorical form, so we replace the ham with ‘0’ and spam with ‘1’ to make it a numerical column.

For converting the text feature column to machine understandable form, we perform NLP pre-processing techniques we clean the data and then perform vectorization in other to make it machine understandable for.

We perform TFIDF vectorization as, it vectorized the feature data or in simple term it encode and standardized the data, so that we can take this data as it is for model building.

The other method for vectorization is WordVec , there we encode our data, but we need to standard scale the data before building the model.

The best fitted model for our dataset is TFIDF vectorized Random Forest Classifier. It has both highest accuracy and lowest log loss.

Even the roc_auc_score of randomforest is highest of all 99.40%

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

Today, spamming mails is one of the biggest issues faced by everyone in the world of the Internet. In such a world, email is mostly shared by everyone to share the information and files because of their easy way of communication and for their low cost. But such emails are mostly affecting the professionals as well as individuals by the way of sending spam emails. Every day, the rate of spam emails and spam messages is increasing. Such spam emails are mostly sent by people to earn income or for any advertisement for their benefit. This increasing amount of spam mail causes traffic congestion and waste of time for those who are receiving that spam mail. The real cost of spam emails is very much higher than one can imagine. Sometimes, the spam emails also have some links which

have malware. And also, some people will get irritated once they see their inbox which is having more spam mails. Sometimes, the users easily get trapped into financial fraud actions, by seeing the spam mails such as job alert mails and commercial mails and offer emails. It may also cause the person to have some mental stress. To reduce all these risks, the system has proposed a machine learning model which will detect spam mail and non-spam emails, and also this system will optimize the data by removing the unwanted mails which contain the advertisement mails and also some useless emails and also some fraud mails. This proposed system will detect the spam mails and ham emails with the dataset consisting of spam mails.

This is one of the most popular NLP model, as it really deals with the real time scenario.

- Limitations of this work and Scope for Future Work

Here we build a model which detect the spam and ham messages/emails, we can also build a system which after identifying spam mails this system will remove that spam emails and this proposed system will calculate the amount of storage before and after the removal of spam mails.
