Spam Detection using NLP Techniques

Abstract: -

Natural Language Processing is a vital field of research having applications in different subjects. Text Classification is a part of NLP where the text is converted into a machine-readable form by performing various methods. Tokenizing, part-of-speech tagging, stemming, chunking are some of the text classification methods. Implementing these methods on our data gives us a classified data on which we will train the model to detect spam and ham messages using Scikit-Learn Classifiers. We proposed a model to solve the issue of classifying messages as spam or ham by experimenting and analyzing the relative strengths of several machine learning algorithms such as K-Nearest Neighbors (KNN), Decision Tree Classifier, Random Forest Classifier, Logistic Regression, SGD Classifier, Multinomial Naive Bayes (NB), Support Vector Machine (SVM) to have a logical comparison of the performance measures of the methods we utilized in this research.

Introduction: -

We get hundreds of messages from unknown sources and our inbox is filled with unwanted emails. These unwanted messages are called spam and essential messages are called ham mails. We will prepare a model that will categorize messages in mobile devices as spam or ham. In order to achieve this, data from the messages is to be collected first and natural language processing techniques are to be applied on it. The spam filtering among messages helps the mobile user to have a good visualization of the inbox. Unnecessary messages will be marked as spam so users need not waste their time reading them. In this paper, we propose to classify data in the messages as either spam (unwanted) or ham(wanted) messages.

Spam is unsolicited bulk messages that are not required for the users but are forced into their inbox. Spams are sent mostly by advertisers, tricksters or by fraud people. Understanding if a message is spam or not can be easily done by reading it once. Our purpose is to detect spam by using various algorithms and measuring their accuracy to find the best fitting algorithm. The common classical approaches which use white-lists and black-lists methods do not work properly as they are only capable of blocking an entire server (source) from sending messages, that may contain some important messages (false positives). Therefore, spam filtration is to be done using text classification techniques.

Methods Used for Building Machine Learning Model.

The naïve bayes is one of the popularly used methods for spam classification and is given by the bayes theorem.

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)},$$

where A and B are events and $P(B) \neq 0$.

- $P(A)$ and $P(B)$ are the probabilities of observing A and B without regard to each other.
- $P(A \mid B)$, a conditional probability, is the probability of observing event A given that B is true.
- $P(B \mid A)$ is the probability of observing event B given that A is true.

We have a message m = (w*1*, w*2*, ... , w*n*), where (w*1*, w*2*, ... , w*n*) is a set of unique words contained in the message. We need to find.

$$P(spam|w1 \cap w2 \cap \ldots \cap wn) = \frac{P(w1 \cap w2 \cap \ldots \cap wn|spam).\, P(spam)}{P(w1 \cap w2 \cap \ldots \cap wn)}$$

If we assume that occurrence of a word are independent of all other words, we can simplify the above expression to

$$\frac{P(w1|spam).P(w2|spam)...P(wn|spam).P(spam)}{P(w1).P(w2)..P(wn)}$$

Random Forest Classifier

It is an ensemble learning method for classification, regression and other tasks is Random forests (or random decision forests) which operate at training time by constructing a multitude of decision trees and giving the class as output which is the mean prediction of the individual trees or mode of the classes.

Decision Tree Classifier A decision tree has a flowchart-like structure containing nodes. Each internal node is a "test" on an attribute which branches into two nodes. These two nodes are the outputs of the decision in the test. The tree ends with leaf nodes which represent a class label (decision taken after computing all attributes). The path from the root node to reach one leaf node is a single classification rule.

Logistic Regression

Although many sophisticated statistical models exit, Logistic regression(or logit regression) uses a logistic function to model a binary dependent variable. In regression analysis, it estimates the parameters of a logit model which is in the form of binary regression. In mathematical words a binary logistic model has a dependent variable with two possible outcomes. These outcomes can be labelled as “0” and “1” which usually represent two opposite classes such as pass/fail, win/loose.

Dataset: -

Model is trained by giving a complete data on which supervised learning can be done. To achieve this, user message data has to be collected and mark them as either spam or ham. The data set used is given by the UCI Machine Learning Repository. It has over 2500 SMS messages which are labelled and are collected for message spam research. The below image is a screenshot of the dataset that has been collected for Spam research which contains all the mobile messages. These messages are tagged accordingly as legitimate(ham) or spam. There are 2893 rows with three columns in the dataset.

```
In [106]: #importing the dataset
          df=pd.read_csv('messages.csv',header=0)

In [107]: df
```

Out[107]:

	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
2890	anglicization of composers ' names	judging from the return post , i must have sou...	0
2891	re : 6 . 797 , comparative method : n - ary co...	gotcha ! there are two separate fallacies in t...	0
2892	re : american - english in australia	hello ! i ' m working on a thesis concerning a...	0

2893 rows × 3 columns

```
In [108]: df.shape
```

Out[108]: (2893, 3)

Experiments performed on the dataset: -

Various experiments are applied on the dataset which were based on Natural Language Processing(NLP) concepts like label encoding, tokenization, stemming, stop word removal, generating features and then applied ensemble method – voting classifier. All these experiments performed in the model classify the data set accurately.

Pre-Processing :-

The messages have to be pre-processed for the removal of unwanted punctuation, grammar, stop words etc.

Label Encoding: -

Label Encoder encode labels with values between „0" and „n-1" where n represents the number of distinct labels for the classes. Same value is as assigned to the labels which are repeated earlier. In our experiment, we convert the class labels to binary values, where „0" is ham and „1" is spam. Already done in dataset.

Stop Word Removal: -

When using Natural Language Processing (NLP), our goal is to perform some analysis or processing so that a computer can respond to text appropriately. A machine cannot understand the human readable form. So, data has to be pre-processed in order to make it machine-readable. This is "pre-processing" of which one of the major forms is to filter out useless data. This useless data (words) is generally referred as „stop words" in Natural Language Processing (NLP).

Stemming: -

Stemming is another pre-processing step that normalize sentences. Stemming is a way to account for the variations of words and sentences which often have a same meaning; furthermore, it will help us shorten the sentences and shorten our lookup. For example, consider the following sentence: 1. I was taking a ride on my horse. 2. I was riding my horse. These sentences mean the same thing, as noted by the same tense in each sentence; however, that isn't intuitively understood by the computer. To account for all the variations of words in the English language, we can use the Porter stemmer, which has been around since 1979.

Feature Generation: -

Feature engineering is the process of constructing features for machine learning algorithms by using the knowledge of that specific domain. The words in each text message are the features on which the algorithm will predict the output. For this purpose, it will be necessary to tokenize each word. The most common 1500 words that are generated in feature generation will be used as our features. Then the data is split in training and testing datasets with a test size of 33%.

TF-IDF: -

TF-IDF stands for Term Frequency-Inverse Document Frequency. In addition to Term Frequency, we compute Inverse document frequency.

$$IDF(w) = \log \frac{Total\ number\ of\ messages}{Total\ number\ of\ messages\ containing\ w}$$

For example, there are two messages in the dataset. 'hello world' and 'hello foo bar'. TF('hello') is 2. IDF('hello') is log(2/2). If a word occurs a lot, it means that the word gives less information. In this model each word has a score, which is TF(w)*IDF(w). Probability of each word is counted as: -

$$P(w) = \frac{TF(w) * IDF(w)}{\sum_{\forall\ words\ x \in train\ dataset} TF(x) * IDF(x)}$$

$$P(w|spam) = \frac{TF(w|spam) * IDF(w)}{\sum_{\forall\ words\ x \in train\ dataset} TF(x|spam) * IDF(x)}$$

Implementation of Algorithms: -

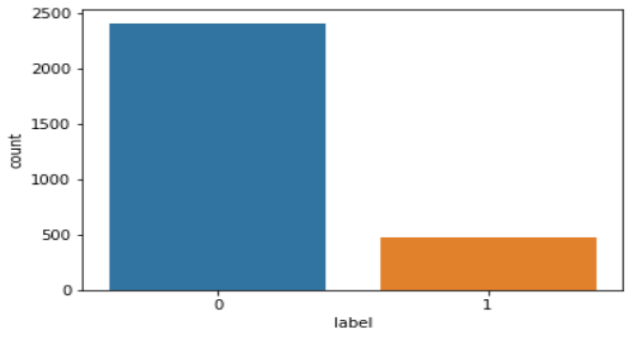
We need to import each algorithm from scikit-learn library along with performance metrics. We require accuracy score and classification report metrics to predict the accuracy and give a classified report on the output.

Visualizing data: -

Let us see which are the most repeated words in the spam messages! We are going to use WordCloud library for this purpose.

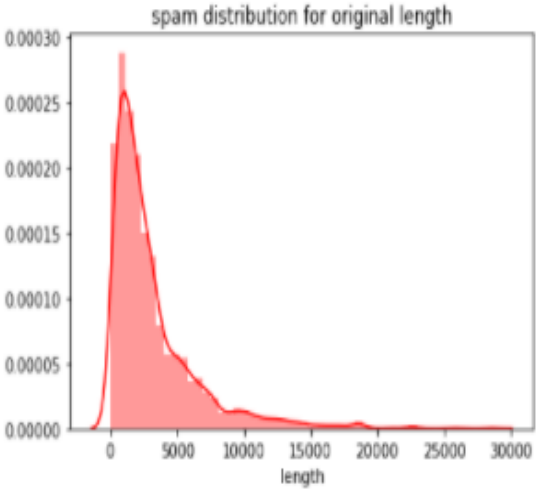
Count of spam and ham: -

```
In [128]: import seaborn as sns
          from matplotlib import pyplot as plt

In [171]: sns.countplot(df['label'])
Out[171]: <matplotlib.axes._subplots.AxesSubplot at 0x217ec617948>
```

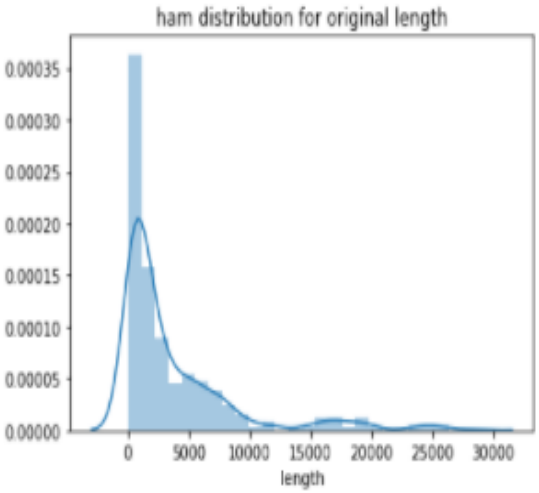


Distribution of Ham and Spam: -

In [129]:
```
sns.distplot(df[df['label']==0]['length'],label='spam dist',color='r')
plt.title('spam distribution for original length')
plt.show()
plt.figsize=(16,8)
```

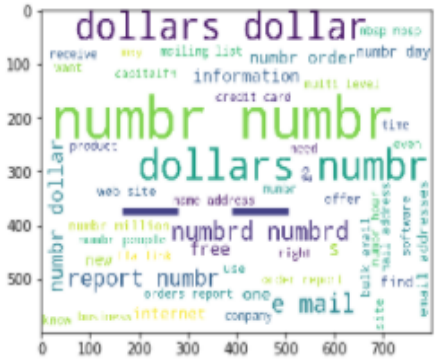


In [130]:
```
sns.distplot(df[df['label']==1]['length'],label='ham dist')
plt.title('ham distribution for original length')
plt.show()
plt.figsize=(16,8)
```

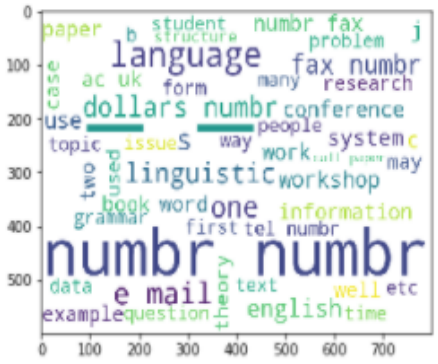


Word cloud for Spam and Ham: -

```
In [169]: #plotting the wordcloud

In [133]: from wordcloud import WordCloud

In [134]: spams=df['message'][df['label']==1]

In [135]: spam_cloud=WordCloud(width=800,height=600,background_color='white',max_words=50).generate(' '.join(spams))

In [136]: spam_cloud
          plt.imshow(spam_cloud)

Out[136]: <matplotlib.image.AxesImage at 0x217ebe1be08>
```




```
In [137]: hams=df['message'][df['label']==0]

In [138]: ham_cloud=WordCloud(width=800,height=600,background_color='white',max_words=50).generate(' '.join(hams))

In [139]: plt.imshow(ham_cloud)

Out[139]: <matplotlib.image.AxesImage at 0x217ec6e3248>
```



Results predicted by the classifier: -

```
In [170]: #using Random forest Classifier

In [149]: from sklearn.ensemble import RandomForestClassifier

In [150]: rf=RandomForestClassifier(n_estimators=50,max_depth=20)

In [151]: from sklearn.model_selection import KFold,cross_val_score

In [152]: k_fold=KFold(n_splits=5)
          cross_val_score(rf,x,y,cv=5,scoring='accuracy',n_jobs=-1)

Out[152]: array([0.93091537, 0.92055268, 0.93782383, 0.93079585, 0.92214533])

In [153]: from sklearn.metrics import precision_recall_fscore_support as score

In [154]: rf_model=rf.fit(x_train,y_train)

In [155]: predrf=rf_model.predict(x_test)
          precision,recall,fscore,support=score(y_test,predrf,pos_label=1 ,average='binary')

In [156]: print('precision:{} recall:{} fscore:{} support:{} '.format(precision,recall,fscore,support))

          precision:1.0 recall:0.5494505494505495 fscore:0.7092198581560284 support:None
```

```
In [157]: from sklearn.naive_bayes import MultinomialNB
          from sklearn.model_selection import train_test_split

In [158]: mnb=MultinomialNB()

In [159]: x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42,test_size=0.33)

In [160]: mnb.fit(x_train,y_train)

Out[160]: MultinomialNB()

In [161]: predmnb=mnb.predict(x_test)

In [173]: #performance metrics

In [162]: from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,f1_score

In [163]: accuracy_score(y_test,predmnb)

Out[163]: 0.8240837696335078

In [164]: print('F1 score is',f1_score(y_test,predmnb))

          F1 score is 0.14285714285714288

In [165]: print(classification_report(y_test,predmnb))

                        precision    recall  f1-score   support

                     0       0.82      1.00      0.90       773
                     1       1.00      0.08      0.14       182

              accuracy                           0.82       955
             macro avg       0.91      0.54      0.52       955
          weighted avg       0.86      0.82      0.76       955
```

Conclusion & Future Scope: -

Spam mails are a serious concern to and a major annoyance for many Internet users. The mode proposed as a solution in this paper is highly beneficial because it introduces a threshold counter which helps overcome congestion on the web server and also maintain the spam filter efficiency but at the same time, it also requires overhead storage space for the databases. Since NLP is a relatively underdeveloped area for research, further enhancements can be made in the field of spam detection for online security using Natural Language Processing in future.