SMS Spam Classifier

Submitted by:

ASHUTOSH MISHRA

ACKNOWLEDGMENT

I'd like to extend my gratitude to my mentor Mr. Md. Kashif for giving me this opportunity to work upon this project. Below are all the details of the project which I consumed while preparing and drafting the project –

The references, research papers, data sources, professionals, etc are majorly referred from "https://www.educative.io/answers/preprocessing-steps-in-natural-language-processing-nlp" study collaterals and data repository. Also, some of the other resources like "**Ineuron**" were explored to gain a deep understanding on the assigned subject.

Above stated details stands correct to the best of my knowledge and I hereby acknowledge the same.

INTRODUCTION

See, we all are aware that SMS's are the best and quick way to text anyone regarding anything that one want to share to other person. As everybody is having their own devices and since there are tons of data's are present over digitally hence spammer use to sneak the data of individuals for their own benefits and this might lead to cybercrime as now-a-days most of the messages that we do receive are basically spam texts and by clicking on any of the spam links our personal data like- Passwords, Bank-details could be under risk.

Today, the world is more digitally connected than ever before. Criminals take advantage of this online transformation to target weaknesses in online systems, networks and infrastructure. There is a massive economic and social impact on governments, businesses and individuals worldwide.

Phishing, ransomware and data breaches are just a few examples of current cyberthreats, while new types of cybercrime are emerging all the time. Cybercriminals are increasingly agile and organized – exploiting new technologies, tailoring their attacks and cooperating in new ways.

Business Problem Framing

According to the some researchers, SMS scam is a big problem across the world as scammers are fooling more than thousands of the people on a daily basis by sending fraud messages and texts and they are sneaking to their personal, professional and banking information to get benefited.

Here we are looking to build a SMS spam detection model as having said that SMS's can be of different types like- rude, abusive, threatening and loathe; so it's really important to identify the SMS type so that person can let themselves aware about the same.

See, the demand to build such models is increasing rapidly as almost most of the people are facing a lot of spam texts and to tackle the same they want to be proactive by identifying such highly spam utterances and in-order to do so NLP and machine learning could be that uprising thing to predict and classify these SMS's.

Our goal is to build a prototype of SMS Spam Classifier which can be used to classify **ham** and **spam** texts so that it can be controlled and restricted from spreading cybercrime and cyberbullying.

Conceptual Background of the Domain Problem

Natural Language Process (NLP) and Machine Learning comes as a vital tool to solve almost any type of business problems to help companies optimize the standards resulting in overall revenue and profits growth. Moreover, it also improves their marketing strategies and demands focus on changing trends.

Now, if we talk about the SMS Spam Classifier project what I feel is that Classification models like Naive-Bayes, Decision Tree Classifier, Random Forest Classifier and Gradient Boosting Classifier would be the perfect algorithms in identifying the texts types.

Also, implementation of NLP techniques like- **Tokenization**, **Stemming**, **Stopwords**, **Lemmatization**, **Bag of Words**, **TF-IDF** and **Word2Vec** would be really interesting while performing text preprocessing and predicting our target variable.

Review of Literature

There has been a remarkable increase in the cases of cybercrime as spammer has really been advanced with their technologies that's why they are easily looting the people and making them suffer mentally without any much efforts. This can take a toll on anyone and affect them mentally leading to depression, mental illness, self-hatred and even suicidal thoughts. Our goal is to build a SMS Spam Classifier which can be used to classify spam and ham texts so that it can be controlled and restricted from spreading cybercrime.

Being a Data Scientist, I have used NLP techniques for text preprocessing and cleansing the corpus and in later part I've used Classification ML Models to design and optimize the results as the target variable is in the form of binary classification.

Some of the classes which I explored are Hyper Tuning Method, CV Ratings, Predictive Modelling and Classification Model whereas I've used NLP libraries like nltk, re(regular expression), stopwords, PorterStemmer, CountVectorizer, TfidfVectorizer and etcetera.

In totality, my project comprises of two NLP techniques as **Bag of Words** (BOW) & **Term Frequency-Inverse Document Frequency** (TF-IDF) and seven different test cases of Classification models where the objective was to train the model with the help of train dataset and later test for accuracy score and different classification reports using ROC_AUC score, F1 score and ROC_AUC curve to understand the accuracy of the different algorithms.

Eventually, the dataset which was provided to me was having 5572 rows and 2 different attributes name as- **texts** and **label**. Attribute texts is basically collection of all the sentences or documents present in dataset i.e. in other word we can say that texts column is basically representing our **Corpus** and label column is our **target** variable containing only two classes as **ham** and **spam**.

Also, I have observed that both the columns are of object data types and there are no null values are present in dataset. Since our corpus are having most of the special characters in it hence I've used regular expression and stopwords for text preprocessing with the help of NLP techniques.

As a result of the NLP techniques like TF-IDF and BOW I've managed to achieve below top model, which are given below as following–

Classification Models (TF-IDF)	Testing Accuracy (in %)
Gaussian NB	92
Gradient Boosting Classifier	96
K Neighbors Classifier	62
Random Forest Classifier	**99**
Decision Tree Classifier	96
Ada Boost Classifier	97
Logistic Regression	96

Motivation for the Problem Undertaken

See, we all are receiving a bulk of spam texts and e-mails almost every day in our daily life and due to the advancement in technologies over the past year's cybercrime has been increased so does our applications to filter-out these spams. Now-a-days we are using a good number of applications like Truecaller, email-spam detector to let ourselves be untapped from these spammers.

My motivation to undergo this project was basically to understand the working of NLP techniques like BOW, CBOW, TF-IDF, Ngram, Stopwords, Tokenization, Lemmatization, Stemming and how the text preprocessing could be helpful to find out the vector representation of each words, corpus and its vocabulary. Hence by building a blueprint of SMS Spam Classifier with the help of NLP and Machine Learning Algorithms such spam could be controlled and be restricted in spreading cybercrimes.

Also, this project guided me to baseline each aspect carefully and be concrete on the decision making process regardless it's an individual or an entity like a company.

In order to cater to the above project, my current knowledge and skill set has aided me a lot which I'd explore on this exponentially further.

Analytical Problem Framing

- Text preprocessing (1st Step) - Converting the corpus into words with the help of Regular Expression, Tokenization, Lemmatization, Stemming, Stopwords and Ngrams.
- Text preprocessing (2nd Step) - Converting the words into vector form with the help of BOW, TF-IDF and Word2Vec methods.
- Label Encoder- To convert the target column into binary classes 0 and 1.
- SMOTE- To balance the class of our target variable into equal number.
- Hyper Parameter Tuning- To find out the best parameter.
- Train Test Split – Building the model and then predicting our target variable
- Cross Validation- Applied K Fold method

Data Sources and their formats

I've been provided with one CSV dataset having 5572 rows and 2 columns.

Let's have a look on our data sources and their attributes.

```
In [63]: #uploading the spam dataset
         df=pd.read_csv('C://Users//Admin//Desktop//Ass//spam.csv',encoding="cp1252")
         df
```

Out[63]:

	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

5572 rows × 5 columns

```
#deleting the unnecessary columns
df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1,inplace=True)
```

```
#renaming the columns for better understanding
df.rename(columns={'v1':'label','v2':'texts'},inplace=True)
```

```
df.head()
```

5]:

	label	texts
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...

As we can see that there are only two columns are given; one is label and other is texts and here our goal is to predict whether the texts are spam or ham. Since here the input is in the form of sentences or corpus so we can't use machine learning model directly as we can't use encoding techniques from machine learning libraries to encode them into vector. So here we are going to use "Natural Learning Process (NLP) techniques as Tokenization,Stemming,Lemmatization,Stopwords,Ngrams,Bag of Words(BOW),Word2Vec,AvgWord2Vec" for feature representation and once when we extract our feature in form of vector will use machine learning libraries to train and test our model accuracy and other parameters.

```
#extracting the data's
df.loc[15] #This will give all the data present in 15th row
```

```
label                                              spam
texts    XXXMobileMovieClub: To use your credit, click ...
Name: 15, dtype: object
```

```
df.texts.loc[15] #This will give texts column data present in 15th row
```

```
'XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemovieclub.com?n=QJKGIGHJJGCBL'
```

```
df.loc[25]
```

```
label                                               ham
texts    Just forced myself to eat a slice. I'm really ...
Name: 25, dtype: object
```

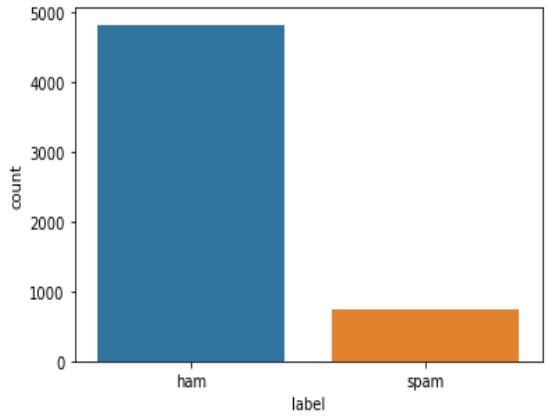
```
df.texts.loc[25]
```

```
"Just forced myself to eat a slice. I'm really not hungry tho. This sucks. Mark is getting worried. He knows I'm sick when I turn down pizza. Lol"
```

```
df.loc[1000]
```

```
label                        ham
texts    No..but heard abt tat..
Name: 1000, dtype: object
```

```
ham     4825
spam     747
Name: label, dtype: int64
```



oint to note that here in the above graph our class is imbalanced so we must need to balance our classes so that we don't get overfitting or nderfitting in our data.(will balance it later in this notebook)

here are 4827 ham texts and 747 spam texts in our dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   label   5572 non-null   object
 1   texts   5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

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

```
label    0
texts    0
dtype: int64
```

No null values are prsent in our dataset and each columns are of the object data type. Also, column label is in the form of category hence will encode it into binary classes. This represent that we have to use classification model going further

```
df.nunique()
```

```
label       2
texts    5169
dtype: int64
```

```
print('Row"s are',df.shape[0])
print('Columns are',df.shape[1])
print('Shape is',df.shape)
```

```
Row"s are 5572
Columns are 2
Shape is (5572, 2)
```

```
#two dimensional dataframe
df.ndim
```

```
2
```

```
#Total datapoints in this dataframe
df.size
```

```
11144
```

```
#indexes are-
df.index
```

```
RangeIndex(start=0, stop=5572, step=1)
```

```
#columns of the dataframes are-
df.columns
```

```
Index(['label', 'texts'], dtype='object')
```

Data Preprocessing Done

Firstly, when I did go through our corpus dataset then I founded that each of the document consists of some special characters in it with numbers and upper-lower case alphabets hence I applied NLP techniques like regular expressions and English stopwords to clean each sentences of our corpus into lower and upper case alphabets only. Then I applied tokenization and stemming to get words that might not be that meaningful and at the end our corpus got free from English stopwords (is, am, he, she, from, for, at, there, it and etcetera).

After data cleaning part done I've applied Bag of Words and TF-IDF techniques to convert the words into vector and then created Vocabulary for both the model.

Since, our target class was imbalanced eventually hence I did balanced it with the help of SMOTE technique and once the class got balanced I've separated my feature variable and target variable for the model building and then with the help of Machine Learning algorithms I've trained different model and later did testing of each model with respect to some valuable parameter like Cross Validation Score, ROC AUC Score, F1-Score to find out the best model in predicting our target variable.

No null values were present in our dataset and I've used Label Encoder method to convert our target variable into binary class as 0 and 1.

After performing the Synthetic Minority Oversampling Technique (SMOTE) we've increased the number of cases in our label in a balanced way as you can see in the below attached screenshot.

```
from imblearn.over_sampling import SMOTE
sm=SMOTE()
X,y=sm.fit_resample(X,y)
```

```
y.value_counts()
```

```
1]: 0    4825
    1    4825
    Name: label, dtype: int64
```

```
print(X.shape)
print(y.shape)
```

```
(9650, 2500)
(9650,)
```

Now we have got our feature and target variable seperately and class of our target variable is now balanced

Data Inputs- Logic- Output Relationships

Since, as mentioned in our problem statement we've to build SMS Spam Classifier model with the help of the documents that are available in our corpus and as we know that there are only one feature are present in our dataset and each documents are having more than one words in it hence after converting each corpus into vector we can find out the correlation w.r.t. our target variable.

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

No.

Hardware and Software Requirements and Tools Used

I have used **Python IDE** (Integrated Development environment) as a dedicated software throughout solving this project.

```
import numpy as np
import pandas as pd
import scipy.stats
from scipy.stats import zscore,boxcox
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Python Libraries that I've used throughout the process are-

- Numpy- It is use for linear algebra
- Pandas- data analysis/manipulation library
- Sklearn- Machine Learning Tool
- Imblearn- Deal with classification problems of Imbalanced classes
- Nltk- NLP Library
- Re- NLP Library regular expression
- Gensim- Word2Vec

Classes-

- Label Encoder- Encoding the categorical variables into number category
- Porter Stemmer- Stemming
- Simple Preprocess - Stopwords
- SMOTE
- Count Vectorizer- BOW
- Tfidf Vectorizer- TF-IDF
- Word2Vec
- Word Net Lemmatizer- Lemmatization
- Sent tokenize- Tokenization
- Cross_Val_Score- CV score
- Grid Search CV- Find out the best parameters for the model

Model/s Development and Evaluation

Identification of possible problem-solving approaches

(methods)

Bag of Words (BOW)-

Bow is a text preprocessing technique which is use to convert the words into vector form. As you can see in the below attachment the word **CountVectorizer** represents BOW.

I've taken **max_features** as **2500** which indicate that there are 2500 features got created and **binary =True** indicate that our target variable has only binary classes as 0 and 1;

ngram_range= (2,2) will create features joining only two words all-together and that we can see in the vocabulary screenshot which is added next to 1st screenshot.

1.Creating the models with the help of Bag of Words (BOW)

```
### text_preprocessing---- word to vector transformation
### finding our features(x) with the help of BOW
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=2500,binary=True,ngram_range=(2,2)) #max_feature will give us top 2500 fatures where as ngram
X=cv.fit_transform(corpus).toarray()
```

```
###our feature for BOW model
X ###we have converted each words of each sentences into vector
```

```
]: array([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```
X.shape #there are 5572 rows and 2500 features
```

```
]: (5572, 2500)
```

```
X[1] #displaying vector for first sentence
```

```
]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
In [45]: cv.vocabulary_  ##we are getting all the unique vocabulary which is generated from our texts dataset

Out[45]: {'here re': 1054,
          'ok lr': 1555,
          'free enr': 673,
          'enr wkl': 510,
          'wkl cp': 2362,
          'quen re': 1678,
          're ppl': 1731,
          'hnk he': 1090,
          'he ge': 888,
          'he lve': 918,
          'here hugh': 1047,
          'freemg he': 681,
          'he here': 897,
          'week nw': 2247,
          'lke fun': 1288,
          'up fr': 2034,
          'fr ll': 637,
          'the re': 1950,
          'per ur': 1591,
```

Term Frequency-Inverse Document Frequency (TF-IDF)-

TF-IDF is also a text preprocessing technique which is use to convert the words into vector form. With the help of this method we give more weightage to those words having rare presence in the sentences. We apply term frequency in rare words and Inverse Document Frequency in most use words of corpus.

Term Frequency (TF) = (No. of the times that particular word in the sentence / No. of the words in that sentence)
Inverse Document Frequency (IDF) = log (Total no. of sentences / No. of sentences containing that word)

As you can see in the below screenshot the word **TfidfVectorizer** represents TF-IDF.

I've taken **max_features** as **2500** which indicate that there are 2500 features created and in the last I've selected **ngram_range= (1,2)** and this will create features of one and two words all-together and that we can see in the vocabulary screenshot which is added next to 1st screenshot.

2.Creating the models with the help of Term Frequency - Inverse Document Frequency (TF-IDF)

```
### text_preprocessing---- word to vector transformation
### finding our features(x) with the help of TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
tv=TfidfVectorizer(max_features=2500,ngram_range=(1,2)) #max_feature will give us top 2500 fatures where as ngram_range will
X=tv.fit_transform(corpus).toarray()
```

```
###our feature for TF-IDF model
X ###we have converted each words of each sentences into vector
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

```
print(X.shape)
print('\n')
print('X[1]',X[1])
print('\n')
print('X[10]',X[10])
print('\n')
print('vocabulary_',tv.vocabulary_) ##we are getting all the unique vocabulary in the combination of one and two pair feature
```

```
(5572, 2500)


X[1] [0. 0. 0. ... 0. 0. 0.]


X[10] [0. 0. 0. ... 0. 0. 0.]


vocabulary_ {'unl': 2075, 'pn': 1663, 'crz': 407, 'nl': 1483, 'bug': 170, 'gre': 827, 'wrl': 2432, 'cne': 363, 'here': 97
5, 're': 1747, 'here re': 983, 'ok': 1565, 'lr': 1337, 'jkng': 1146, 'wf': 2267, 'ok lr': 1569, 'free': 723, 'enr': 563,
'wkl': 2348, 'cp': 380, 'wn': 2368, 'cup': 423, 'fnl': 679, 'tex': 1985, 'receve': 1791, 'quen': 1736, 'ppl': 1679, 've
r': 2193, 'free enr': 727, 'enr wkl': 566, 're ppl': 1772, 'un': 2062, 'erl': 584, 'hr': 1048, 'lre': 1342, 'hen': 955,
'nh': 1471, 'hnk': 1016, 'he': 860, 'ge': 768, 'uf': 2054, 'lve': 1354, 'run': 1894, 'hugh': 1061, 'hnk he': 1020, 'he g
e': 884, 'he lve': 903, 'freemg': 733, 'rlng': 1861, 'been': 76, 'week': 2243, 'nw': 1537, 'wr': 2417, 'bck': 42, 'lke':
1266, 'fun': 760, 'up': 2083, 'fr': 684, 'll': 1278, 'xxx': 2462, 'en': 525, 'rcv': 1746, 'he here': 889, 'lke fun': 126
8, 'up fr': 2087, 'fr ll': 700, 'even': 593, 'brher': 132, 'pek': 1608, 'wh': 2270, 'the': 1996, 'pen': 1610, 'the re': 1
999, 'per': 1615, 'ur': 2103, 'reque': 1836, 'melle': 1382, 've': 2183, 'cllerune': 314, 'cller': 311, 'pre': 1696, 'fre
n': 736, 'per ur': 1618, 'ur reque': 2148, 've been': 2184, 'been ur': 81, 'cllerune fr': 315, 'll cller': 1284, 'cller p
```

Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

- from sklearn.linear_model import LogisticRegression
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import

  accuracy_score,classification_report,confusion_matrix,roc_auc_score,roc_curve
- from sklearn.model_selection import cross_val_score
- from sklearn.model_selection import GridSearchCV
- from sklearn.naive_bayes import GaussianNB
- from sklearn.neighbors import KNeighborsClassifier
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.ensemble import

  RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
- from sklearn.metrics import roc_curve,roc_auc_score
- lr=LogisticRegression()
- gb=GaussianNB()
- neighbor=KNeighborsClassifier()
- dtc=DecisionTreeClassifier()

25-Dec-22 1:11 AM Ashutosh Mishra

- rfc=RandomForestClassifier()
- ad=AdaBoostClassifier()
- grd=GradientBoostingClassifier()
- import re
- import nltk
- nltk.download('stopwords')
- from nltk.corpus import stopwords
- from nltk.stem.porter import PorterStemmer
- from sklearn.feature_extraction.text import CountVectorizer
- from imblearn.over_sampling import SMOTE
- from sklearn.feature_extraction.text import TfidfVectorizer

Run and Evaluate selected models

Best Model (Random Forest Classifier)

I've tested seven different model with the help of BOW and TF-IDF and the best model that I got is **Random Forest Classifier** from **TF-IDF** having **99%** of testing accuracy, F1-score each and **0.99** as ROC_AUC score.

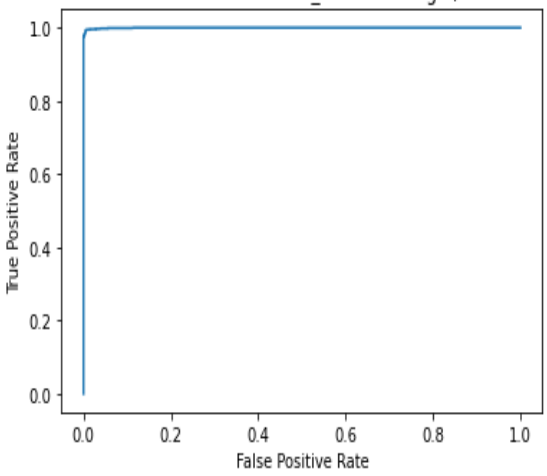
Kindly see the below attachments for better understanding of the same-

```
model(rfc,X,y)
```

```
For model RandomForestClassifier(class_weight='balanced', criterion='log_loss',
                       max_features='log2')
Training_Accuracy_Score= 0.9996113989637305
Testing_Accuracy_Score= 0.9870466321243523

 Classification Report-
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       985
           1       1.00      0.97      0.99       945

    accuracy                           0.99      1930
   macro avg       0.99      0.99      0.99      1930
weighted avg       0.99      0.99      0.99      1930


 Confusion Metrix-
 [[984   1]
 [ 24 921]]
```

AUC_ROC CURVE

ROC_Curve for the model RandomForestClassifier(class_weight='balanced', criterion='log_loss', max_features='log2')



ROC AUC SCORE is- 0.9867939730883895

Finding out the best K-Fold Value

At the K-Fold 2 the CV score of model RandomForestClassifier(class_weight='balanced', criterion='log_loss', max_features='log2') is 0.9859067357512954

At the K-Fold 3 the CV score of model RandomForestClassifier(class_weight='balanced', criterion='log_loss', max_features='log2') is 0.9895337322299497

Key Metrics for success in solving problem under consideration

Below are the some **Classification Models** where I used below metrics -

- CV Score – Model testing Accuracy
- F1-Score – It combines precision and recall into a single number
- ROC AUC Score – ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes.
- ROC AUC Curve- The Reciever operating characteristic curve plots the true positive (TP) rate versus the false positive (FP) rate at different classification thresholds

CONCLUSION

With the help of BOW and TF-IDF I've plotted the below tables to find out which model has given good accuracy while predicting the target variable.

Here in the very first table I've plotted all the machine models with the help of BOW technique-

Classification Models (BOW)	Training Accuracy (in %)	Testing Accuracy (in %)	ROC AUC Score (in %)	F1-Score (in %)	Ranking
Logistic Regression	77	77	0.76	77	4
Gaussian NB	82	80	0.80	80	2
Gradient Boosting Classifier	73	74	0.73	74	6
K Neighbors Classifier	78	76	0.75	76	5
Random Forest Classifier	86	81	0.81	81	1
Decision Tree Classifier	86	78	0.79	78	3
Ada Boost Classifier	72	72	0.72	73	7

As we can see that Random forest classifier is best algorithm among all as it gives highest score as compare to others and also the CV score of this algorithm is equal to its testing accuracy which is highest among other. ROC AUC score of this algorithm is 0.81 which is highest among all and it indicates that out of 100 times, 81 times our model is predicting right class. Hence RFC model works well while predicting the target variable. Now let's find out the Model accuracy with the help of TF-IDF model and check whether our model accuracy would change or not...Lets find out the same.

Classification Models (TF-IDF)	Training Accuracy (in %)	Testing Accuracy (in %)	ROC AUC Score (in %)	F1-Score (in %)	Ranking
Logistic Regression	96	96	0.96	97	4
Gaussian NB	91	92	0.92	92	6
Gradient Boosting Classifier	96	96	0.96	96	5
K Neighbors Classifier	100	62	0.63	62	7
Random Forest Classifier	100	99	0.99	99	1
Decision Tree Classifier	100	96	0.96	96	2
Ada Boost Classifier	97	97	0.97	97	3

As we can see that after applying the TF-IDF model our model's accuracy got increased by 15-25% which is really a great achievement in predicting the text spam detections. Random Forest Classifier is the best model for achieving the same having almost 100% accuracy across all the parameters and that indicates our model is predicting, spam as spam and ham as ham, 100 out of 100 times.

Learning Outcomes of the Study in respect of Data Science

I have used the NLP techniques like BOW & TF-IDF for this problem statement and also used Machine Learning Classification Model to design and optimize the results. Some of the classes which I explored from nltk libraries were – PorterStemmer, WordNetLemmatizer, sent_tokenize and Stopwords etcetera.

One of the key challenges was like applying the Word2Vec NLP technique as I was getting an error while training the model and the error was "only length-1 arrays can be converted to Python scalars". However, I've founded some really interesting similar words concepts, vocabulary and their vector representations as well.

At the end, It was really a good experience working on this project as I learned some new techniques and libraries of Natural Language Process and which will definitely going to be helpful in upcoming projects as well.

Limitations of this work and Scope for Future Work

The same analysis which is done for this project can be used for other spam classifier projects because at the end we will be getting some corpus that will help us to predict the target variable.

This project could also be useful for predicting the ratings classification; the only difference in this scenario would be the number of classes (1, 2, 3, 4, and 5) of the target variable.