FLIP ROBO

EMAIL SPAM CLASSIFIER PROJECT

Submitted by

Raganee Verma

(intern fliprobo **)**

ACKNOWLEDGMENT

I would like to express my special thanks to our mentor Ms. Khushboo Garg mam as well as Flip Robo Technologies who gave me the opportunity to do this project Email Spam Classification, which also helped me in doing lots of knowledge and research work. wherein I came to know about so many new things, especially the Natural Language Processing and Natural Language Toolkit parts.

I am also using a few external resources that helped me to complete this project and I learn from the samples and modify things according to my project requirement. The external resources, research paper and articles that were used in creating this project are listed belowFinally, I would want to convey my sincere thanks Datatrained Academy

The website that I referred are:

https://learning.datatrained.com

https://www.w3schools.com

https://medium.com/coders-camp

https://github.com

https://www.geeksforgeeks.org

https://www.javatpoint.com/nlp

https://www.educative.io/answers/preprocessing-steps-in-natural-

language-processing-nlp

https://www.youtube.com/watch?v=5ctbvkAMQO4

https://www.youtube.com/watch?v=X2vAabgKiuM

INTRODUCTION

● Business Problem Framing

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 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 to ham (legitimate) or spam.

● Conceptual Background of the Domain Problem

The main goal of the assignment is to show how you could design a spam filtering system from scratch.

The files contain one message per line. Each line is composed by two columns:

- v1 contains the label (ham or spam)
- v2 contains the raw text.

● Motivation for the Problem Undertaken

Implementing spam filtering is extremely important for any organization. Not only does spam filtering help keep garbage out of email inboxes, it helps with the quality of life of business emails because they run smoothly and are only used for their desired purpose. Spam filtering is essentially an anti-malware tool, as many attacks through email are trying to trick users to click on a malicious attachment, asking them to supply their credentials, and much more.

Analytical Problem Framing

● Mathematical/ Analytical Modeling of the Problem

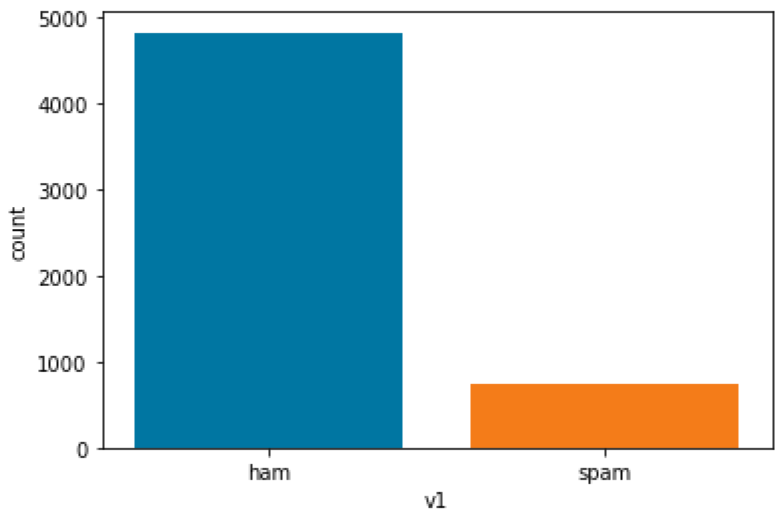
- Information of the dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

- Description of the dataset:

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

-Data visualization



● Data Sources and their formats

- A collection of 5573 rows of 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 involves 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.

	v1	v2
0	0	go jurong point, crazy.. available bugis n gre...
1	0	ok lar... joking wif oni...
2	1	free entry number wkly comp win fa cup final t...
3	0	dun say early hor... c already say...
4	0	nah think go usf, life around though
...	...	...
5567	1	numbernd time tried number contact u. ådollers...
5568	0	ì_ b going esplanade fr home?
5569	0	pity, mood that. so...any suggestions?
5570	0	guy bitching acted like i'd interested buying ...
5571	0	rofl. true name

5572 rows × 2 columns

● Data Preprocessing Done

In data pre-processing, I have done the various steps to clean the dataset, as the dataset contains the comment that are in object datatype, which cannot be read by the model, so before giving the features to the model I had to convert that object datatype to meaningful data and that can be understand by the model, so for this I have used the NLP (Natural Processing Language).

“Natural language processing (NLP) refers to the branch of computer science and more specifically, the branch of artificial intelligence (AI) concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.”

- Converting to lower

```
mail['v2'] = mail['v2'].str.lower()
```

- Stop Words

```
stop_words = stopwords.words('english')
```

- Original Length

```
original = mail['v2'].str.len()
```

- Email Addresses

```
mail['v2'] = mail['v2'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','email')
```

- Website

```
mail['v2'] = mail['v2'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','website')
```

- Applying Lemmatization

```
lem = WordNetLemmatizer()
mail['v2'] = mail['v2'].apply(lambda x: ' '.join(lem.lemmatize(t)
                                                 for t in x.split()))
```

- Cleaned Length

```
clean = mail.v2.str.len()
```

```
print ('Origian Length :', original.sum())
print ('Clean Length :', clean.sum())
```

```
Origian Length : 446422
Clean Length : 336896
```

- Phone Numbe

```
mail['v2'] = mail['v2'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')
```

- Currency

```
mail['v2'] = mail['v2'].str.replace(r'£|\$', 'dollers')
```

- Numbers

```
mail['v2'] = mail['v2'].str.replace(r'\d+(\.\d+)?', 'number')
```

- Dealing with Punctuation

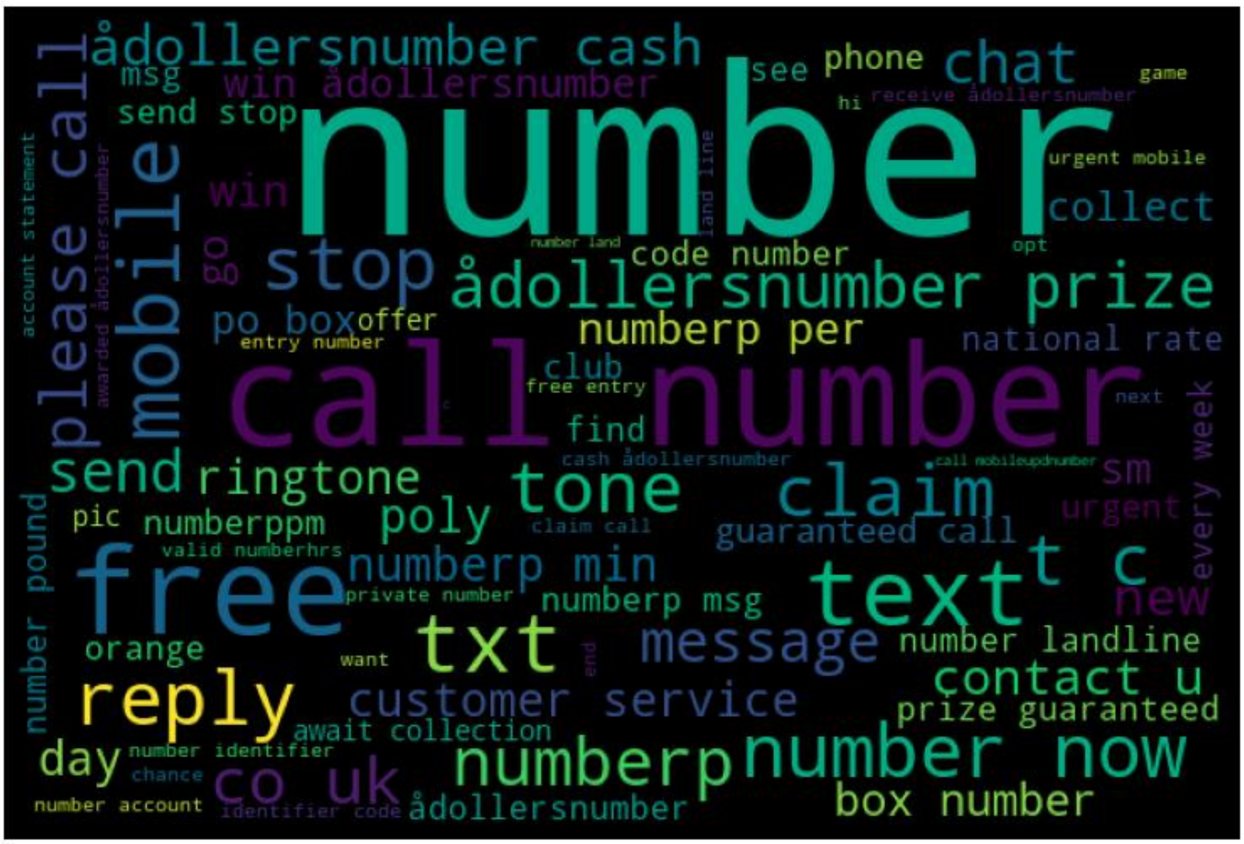
```
mail['v2'] = mail['v2'].apply(lambda x: ' '.join(term for term in x.split()
                                                  if term not in string.punctuation))
```

- Removing Stop word

```
stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
mail['v2'] = mail['v2'].apply(lambda x: ' '.join(term for term in x.split()
                                                  if term not in stop_words))
```

WordCloud

A word cloud (also known as a tag cloud) is **a visual representation of words**. Cloud creators are used to highlight popular words and phrases based on frequency and relevance. They provide you with quick and simple visual insights that can lead to more in-depth analyses.



● Data Inputs- Logic- Output Relationships

Used TF-IDF Vectorizer to encode the comments section.

“TfidfVectorizer is the base building block of many NLP pipelines. It is a simple technique to vectorize text documents i.e. transform sentences into arrays of numbers and use them in subsequent tasks.”

```
tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
features = tf_vec.fit_transform(mail['v2'])
x = features
```

```
y = mail.v1
```

● Hardware and Software Requirements and Tools Used

Anaconda-navigator

jupyter notebook

matplotlib-inline==0.1.6

numpy==1.23.2

packaging==21.3

pickleshare==0.7.5

platformdirs==2.5.2

prompt-toolkit==3.0.30

pyparsing==3.0.9

python-dateutil==2.8.2

scikit-learn==1.1.2

scipy==1.9.0

sklearn==0.05

NLP

Model/s Development and Evaluation

● Identification of possible problem-solving approaches

- EDA

- Description

- Visualization

- Data cleaning

- Data Pre-processing (NLP)

- Word Cloud

- Encoding

- Model Building

- Select the best model

- Cross-Validation

- Hyperparameter tuning

- Function for Training & Testing

```
def score(clas, x_train, x_test, y_train, y_test, train = True):
    if train:
        y_pred = clas.predict(x_train)
        print('\n ----- Train Result ----- \n')
        print('Accuracy Score:', accuracy_score(y_train,y_pred))
        print('\n ----- Classification Report ----- \n', classification_report(y_train,y_pred))
        print('\n ----- Confusion matrix ----- \n', confusion_matrix(y_train,y_pred))

    elif train == False:
        pred = clas.predict(x_test)
        print('\n ----- Test Result ----- \n')
        print('Accuracy Score:', accuracy_score(y_test,pred))
        print('\n ----- Classification Report ----- \n', classification_report(y_test,pred))
        print('\n ----- Confusion matrix ----- \n', confusion_matrix(y_test,pred))
        print('\n ----- Roc Curve ----- \n')
        plot_roc_curve(clas, x_test, y_test)
```

- Model Instantiating

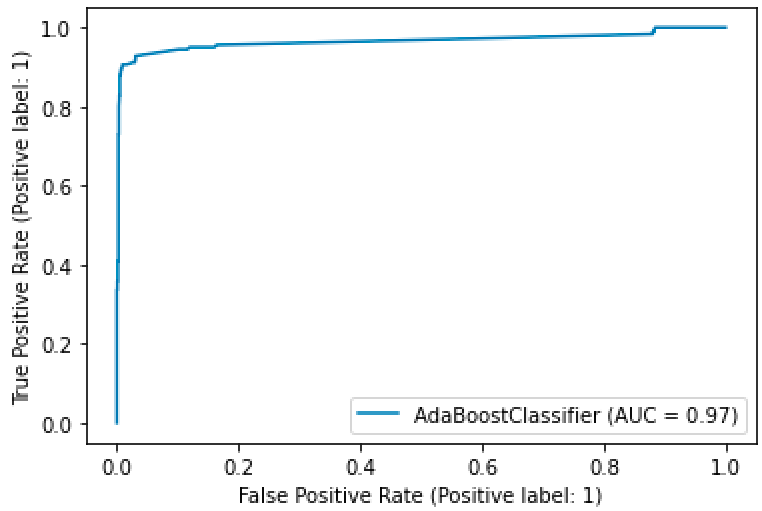
```
]:  ada = AdaBoostClassifier()
    gbr = GradientBoostingClassifier()
    knn = KNeighborsClassifier()
    rf = RandomForestClassifier()
    lr = LogisticRegression()
    navie = GaussianNB()
```

● Run and Evaluate selected models

- AdaBoost Classifier

```
Accuracy Score: 0.9868389566882029

 ----- Classification Report -----
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      3613
           1       0.99      0.92      0.95       566

    accuracy                           0.99      4179
   macro avg       0.99      0.96      0.97      4179
weighted avg       0.99      0.99      0.99      4179


 ----- Confusion matrix -----
 [[3606    7]
 [  48  518]]

 ----- Test Result -----

Accuracy Score: 0.9791816223977028

 ----- Classification Report -----
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1212
           1       0.95      0.88      0.92       181

    accuracy                           0.98      1393
   macro avg       0.97      0.94      0.95      1393
weighted avg       0.98      0.98      0.98      1393


 ----- Confusion matrix -----
 [[1204    8]
 [  21  160]]
```

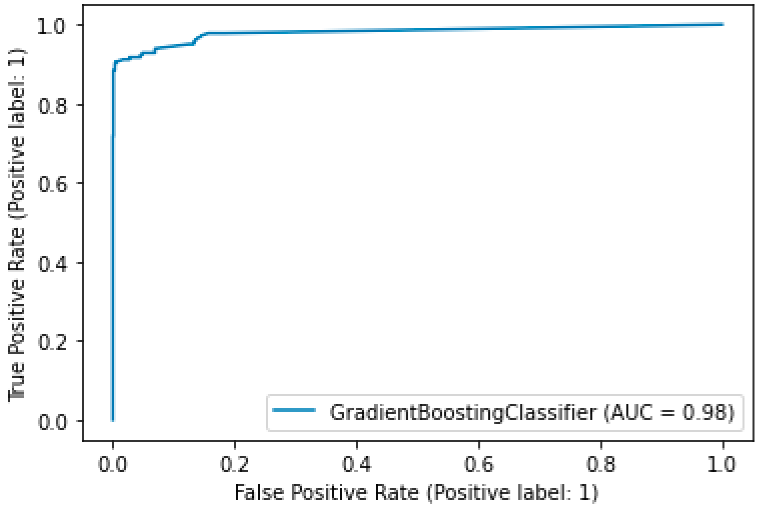
----- Roc Curve -----



- GradientBoosting Classifier

```
Accuracy Score: 0.9882747068676717

 ----- Classification Report -----
               precision    recall  f1-score   support

           0       0.99      1.00      0.99      3613
           1       1.00      0.92      0.95       566

    accuracy                           0.99      4179
   macro avg       0.99      0.96      0.97      4179
weighted avg       0.99      0.99      0.99      4179


 ----- Confusion matrix -----
 [[3612    1]
 [  48  518]]

 ----- Test Result -----

Accuracy Score: 0.9842067480258435

 ----- Classification Report -----
               precision    recall  f1-score   support

           0       0.98      1.00      0.99      1212
           1       0.99      0.88      0.94       181

    accuracy                           0.98      1393
   macro avg       0.99      0.94      0.96      1393
weighted avg       0.98      0.98      0.98      1393


 ----- Confusion matrix -----
 [[1211    1]
 [  21  160]]
```

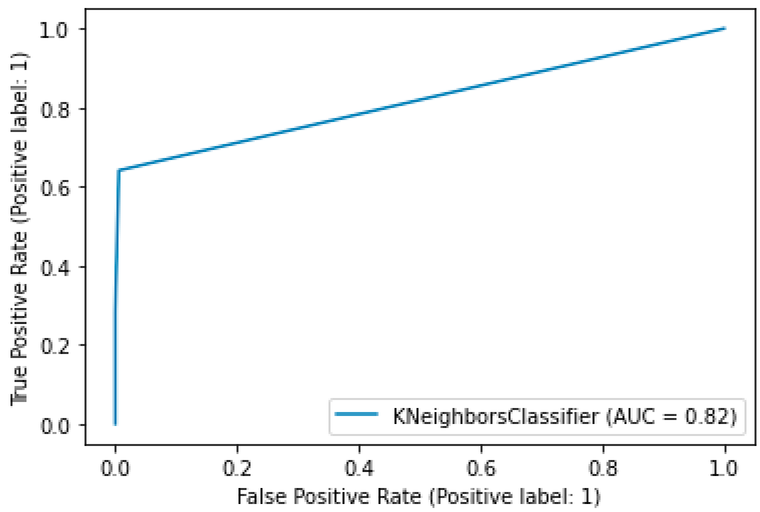
----- Roc Curve -----



- KNeighbors Classifier

```
Accuracy Score: 0.9270160325436707

 ----- Classification Report -----
              precision    recall  f1-score   support

           0       0.92      1.00      0.96      3613
           1       1.00      0.46      0.63       566

    accuracy                           0.93      4179
   macro avg       0.96      0.73      0.80      4179
weighted avg       0.93      0.93      0.92      4179


 ----- Confusion matrix -----
 [[3613    0]
 [ 305  261]]

 ----- Test Result -----

Accuracy Score: 0.914572864321608

 ----- Classification Report -----
              precision    recall  f1-score   support

           0       0.91      1.00      0.95      1212
           1       0.98      0.35      0.51       181

    accuracy                           0.91      1393
   macro avg       0.95      0.67      0.73      1393
weighted avg       0.92      0.91      0.90      1393


 ----- Confusion matrix -----
 [[1211    1]
 [ 118   63]]
```

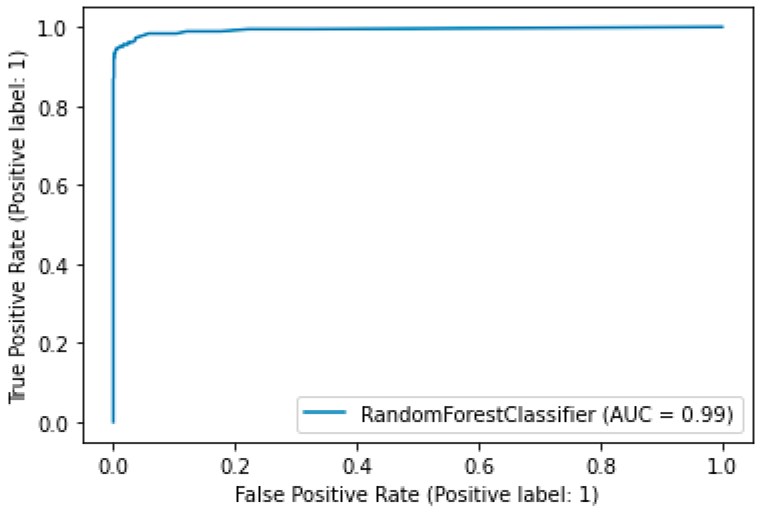
```
----- Roc Curve -----
```



- RandomForest Classifier

```
Accuracy Score: 0.9997607083034219

 ----- Classification Report -----
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3613
           1       1.00      1.00      1.00       566

    accuracy                           1.00      4179
   macro avg       1.00      1.00      1.00      4179
weighted avg       1.00      1.00      1.00      4179

 ----- Confusion matrix -----
 [[3612    1]
 [   0  566]]

 ----- Test Result -----

Accuracy Score: 0.9827709978463748

 ----- Classification Report -----
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1212
           1       0.99      0.87      0.93       181

    accuracy                           0.98      1393
   macro avg       0.99      0.94      0.96      1393
weighted avg       0.98      0.98      0.98      1393

 ----- Confusion matrix -----
 [[1211    1]
 [  23  158]]
```

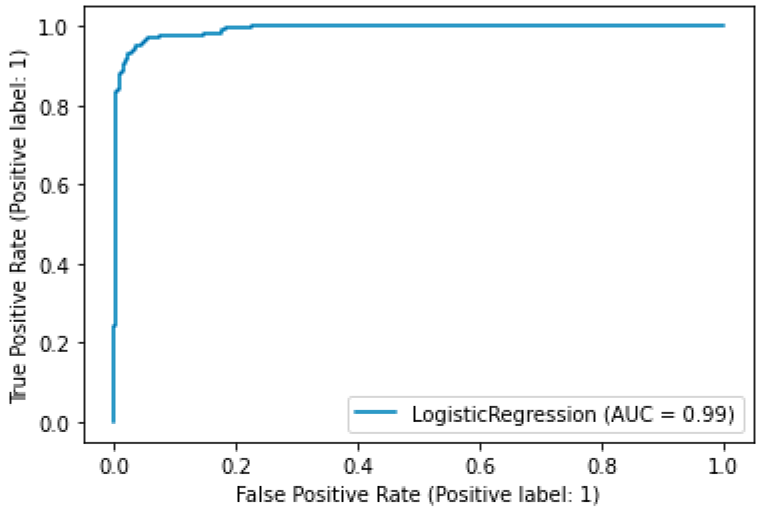
```
----- Roc Curve -----
```



- Logistic Regression

```
 ----- Train Result -----

Accuracy Score: 0.9777458722182341

 ----- Classification Report -----
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      3613
           1       0.97      0.86      0.91       566

    accuracy                           0.98      4179
   macro avg       0.98      0.93      0.95      4179
weighted avg       0.98      0.98      0.98      4179

 ----- Confusion matrix -----
 [[3600   13]
 [  80  486]]

 ----- Test Result -----

Accuracy Score: 0.9712849964106246

 ----- Classification Report -----
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1212
           1       0.99      0.79      0.88       181

    accuracy                           0.97      1393
   macro avg       0.98      0.89      0.93      1393
weighted avg       0.97      0.97      0.97      1393

 ----- Confusion matrix -----
 [[1210    2]
 [  38  143]]
```

----- Roc Curve -----



- Naive Bayes (GaussianNB)

```
 ----- Train Result -----

Accuracy Score: 0.968413496051687

 ----- Test Result -----

Accuracy Score: 0.11198851399856424
```

● Interpretation of the Results

RandomForest Classifier is giving the best result as compared to

Others. So we can perform hyper parameter tuning for this model.

```python
param = {'n_estimators':range(0,100,10),
         'ccp_alpha':[0.0,0.2,0.4,0.5,0.7,0.8,1.0]
         }
```

```python
grid = GridSearchCV(rf,param_grid = param)
grid.fit(x_train,y_train)
print('Best Params = ',grid.best_params_)
```

```
Best Params =  {'ccp_alpha': 0.0, 'n_estimators': 40}
```

```python
rf_hyp = RandomForestClassifier(ccp_alpha = 0.0, n_estimators = 50)
```

```python
rf_hyp.fit(x_train,y_train)
score(rf_hyp, x_train,x_test,y_train,y_test,train = True)
score(rf_hyp, x_train,x_test,y_train,y_test,train = False)
```

```
 ----- Train Result -----

Accuracy Score: 0.9995214166068438

 ----- Classification Report -----
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3613
           1       1.00      1.00      1.00       566

    accuracy                           1.00      4179
   macro avg       1.00      1.00      1.00      4179
weighted avg       1.00      1.00      1.00      4179


 ----- Confusion matrix -----
 [[3612    1]
 [   1  565]]
```

Original vs Predicted

```
a_rfc = np.array(y_test)
predicted_rfc = np.array(rf.predict(x_test))
df_rfc = pd.DataFrame({'Original':a_rfc,'Predicted':predicted_rfc})
df_rfc
```

	Original	Predicted
0	1	1
1	0	0
2	0	0
3	0	0
4	0	0
...	...	...
1388	0	0
1389	0	0
1390	0	0
1391	0	0
1392	0	0

1393 rows × 2 columns

Saving the model.

```
filename = 'Email_spam.pickle'
pickle.dump(rf, open(filename, 'wb'))
```

CONCLUSION

● Learning Outcomes of the Study in respect of DataScience

Apply computing theory, languages, and algorithms, as well as mathematical and statistical models, and the principles of optimization to appropriately formulate and use data analyses. Formulate and use appropriate models of data analysis to solve hidden solutions to business-related challenges. Perform well in a group.