PROJECT REPORT

INTRODUCTION

OVERVIEW:

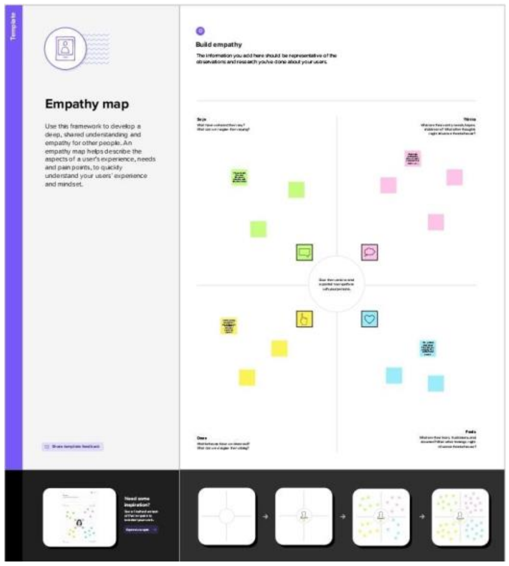
Since machine learning have the capacity to adapt to varying conditions, Gmail and Yahoo mail spam filters do more than just checking junk emails using pre-existing rules. They generate new rules themselves based on what they have learnt as they continue in their spam filtering operation

PURPOSE:

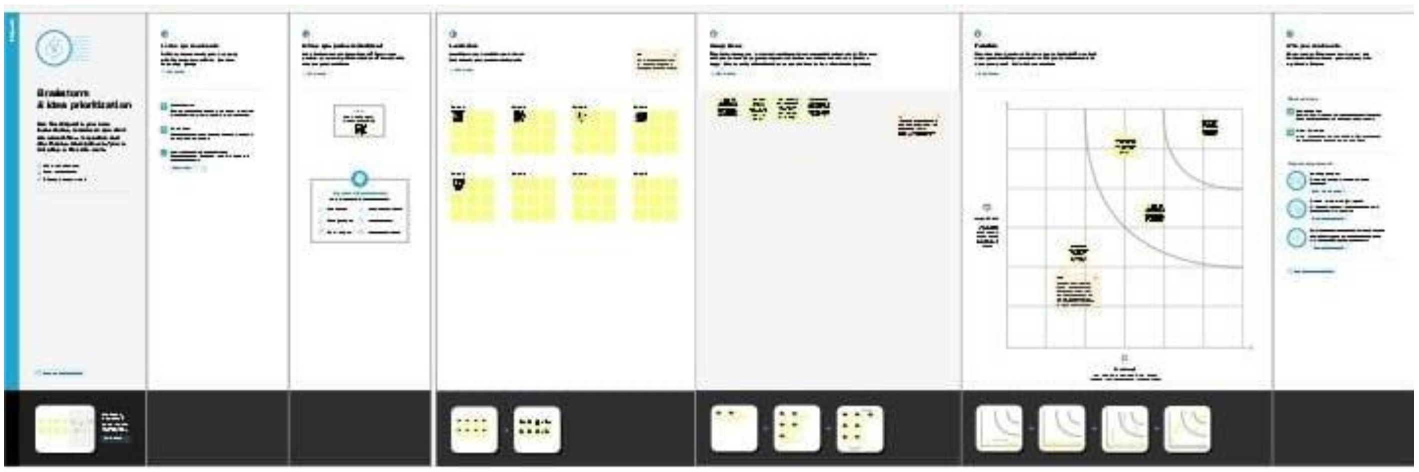
Thus consuming time and resources, Machine learning makes it easier because it learns to recognise the unsolicited emails (spam) and legitimate emails (ham) automatically and then applies those learned instructions to unknown incoming emails[2].

PROBLEM DEFINITION & DESIGN THINKING

Empathy Map


Template
Empathy map
Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.
Build empathy
The information you add here should be representative of the observations and research you've done about your users.

Ideation & Brainstorming Map



Result

Machine learning algorithms have been extensively applied in the field of spam filtering. Substantial work have been done to improve the effectiveness of spam filters for classifying emails as either ham (valid messages) or spam (unwanted messages) by means of ML classifiers. They have the ability to recognise distinctive characteristics of the contents of emails. Many significant work have been done in the field of spam filtering using techniques that does not possess the ability to adapt to different conditions; and on problems that are exclusive to some fields e.g. identifying messages that are hidden inside a stego image. Most of the machine learning algorithms used for classification of tasks were designed to learn about inactive objective groups. The authors in [119] posited that when these algorithms are trained on data that has some data that have been poisoned by an enemy, it makes the algorithms susceptible to a number of different attacks on the reliability and accessibility of the data. As a matter of fact, manipulating as minute as 1% of the training data is enough in certain instances [120]. Though it might be strange to hear that the data supplied by an enemy is used to train a system, it does happen in some real world systems. Examples include spam detection systems, spam connection, financial fraud, credit card fraud, and other unwelcome deeds where the earlier deeds of the enemy are a major origin of training data. The unfortunate thing is that a good number systems are re-trained regularly using the new instances of undesirable activities. This serves as a launching pad for attacker to launch more attacks on such system.
One of the open problem that needs to be addressed is handling of threat to the security of the spam filters. Though some attempt have been made to address this problem. For example, the threat model for adaptive spam filters proposed by [121] categorises attacks based to whether they are causative or exploratory, targeted or indiscriminate, and if they are meant to interrupt reliability or

accessibility. The purpose of causative attack is to trigger error in categorisation of messages, whereas an exploratory attack aims to determine the classification of a message or set of messages. An attacks on integrity is meant to have a negative influence on the classification of spam, on the other hand, attacks on accessibility is meant to have a negative influence on the c classification of ham. The fundamental purpose of a spammer is to send spam which cannot be seized by the filter (or user) and labeled as spam. There are other potential capabilities of attack which all depend entirely on the ability to send random messages grouped as spam. A larger percentage of spam filters are nevertheless susceptible to different kinds of attack. For example, Bayes filter is susceptible to mimicry attack [120]. Naïve Bayes and AdaBoost also demonstrated endless deterioration to adversary control attack.
Further research work need to be conducted to tackle the fact that email spam filtering is a concept drift problem. As such, while the spam filter researchers are trying to increase the prognostic accuracy of the filter, the spammers are also evolving and trying to surpass the efficiency of the spam filters. It becomes very important to develop more efficient techniques that will adequately handle the trend or progression in spam features that makes them to evade many spam filters undetected. The most successful technique applied in filtering spam is the content based spam filtering approach which classify emails as either spam or ham depending on the data that made up the content of the message. Examples of this technique include Bayesian Filtering, SVM, kNN classifier, Neural Network, AdaBoost classifier, and others. Systems based on machine learning approach facilitates learning and adjustment to recent dangers posed to the security of spam filters. They also have the capacity to counter curative channels that spammers are using.

We hereby suggest that the future of email spam filters lies in deep learning for content-based classification and deep adversarial learning techniques. Deep learning is a kind of machine learning technique that allows computers to learn from experience and knowledge devoid of explicit programming and mine valuable patterns from primitive data [122]. The traditional machine learning algorithms finds it very hard to mine adequately-represented features because to the limitations that characterised such algorithms. The shortcomings of the usual machine learning algorithms include: need for knowledge from expert in a particular field, curse of dimensionality, and high computational cost. Deep learning have been applied to solve representation problem by creating several naive features to represent a complicated concept. Deep learning will be far more effective in solving the problem of spam email because as number of available training data is increasing, the effectiveness and efficiency of deep learning

becomes more pronounced. Deep learning models have the capacity to solve sophisticated problems by using intricate and huge models. Thus, they exploit the computational power of modern CPUs and GPUs. Deep learning is generally considered to be a black box since we have imperfect knowledge of the explanations behind its high performance. Despite the huge success of deep learning in solving many problems, it has been discovered lately that deep neural networks are susceptible to adversarial examples. Adversarial examples are unnoticeable to human but can effortlessly fool deep neural networks during the testing/deploying phase. The helplessness to adversarial examples becomes one of the foremost dangers for using deep neural networks in situations where safety is very crucial. Therefore, the adversarial deep learning technique is a great method that is yet to be exploited in email spam filtering.
Summarily, the open research problems in email spam filtering are itemized below:

Advantage

The machine learning model used by Google have now advanced to the point that it can detect and filter out spam and phishing emails with about 99.9 percent accuracy. The implication of this is that one out of a thousand messages succeed in evading their email spam filter.

Disadvantage

Thousands of spam emails may reach Inboxes before a spammer's email address, IP or domain is blacklisted. Spam filtering is machine-based so there is a room for mistakes called “false positives.” Bayesian filters may be fooled by spammers, e.g. in a case of using large blocks of legitimate text

Applications

A spam filter is a program used to **detect unsolicited, unwanted and virus-infected emails and prevent those messages from getting to a user's inbox**.

Conclusion

In this study, we reviewed machine learning approaches and their application to the field of spam filtering. A review of the state of the art algorithms been applied for classification of messages as either spam or ham is provided. The attempts made by different researchers to solving the problem of spam through the use of machine learning classifiers was discussed. The evolution of spam messages over the years to evade filters was examined. The basic architecture of email spam filter and the processes involved in filtering spam emails were looked into. The paper surveyed some of the publicly available datasets and performance metrics that can be used to measure the effectiveness of any spam filter. The challenges of the machine learning algorithms in efficiently handling the menace of spam was pointed out and comparative studies of the machine learning technics available in literature was done. We also revealed some open research problems associated with spam filters. In general, the figure and volume of literature we reviewed shows that significant progress have been made and will still be made in this field. Having discussed the open problems in spam filtering, further research to enhance the effectiveness of spam filters need to be done. This will make the development of spam filters to continue to be an active research field for academician and industry practitioners researching machine learning techniques for effective spam filtering. Our hope is that research students will use this paper as a spring board for doing qualitative research in spam filtering using machine learning, deep leaning and deep adversarial learning algorithms.

Future Scope

- Lack of effective strategy to handle the threats to the security of the spam filters. Such an attack can be causative or exploratory, targeted or indiscriminate attack.
- •
  The inability of the current spam filtering techniques to effectively deal with the concept drift phenomenon.
- •
  Majority of the existing email spam filters does not possess the capacity to incrementally learn in real-time. Conventional spam email classification techniques are no longer viable to cope in real time environment that is characterised by evolving data streams and concept drift.
- •
  Failure of many spam filters to reduce their false positive rate.
- •
  Development of more efficient image spam filters. Most spam filters can only classify spam messages that are text. However, many savvy spammers send spam email as text embedded in an image (stego image) thereby making the spam email to evade detection from filters.

- •
  The need to develop adapted, scalable, and integrated filters by applying ontology and semantic web to spam email filtering.

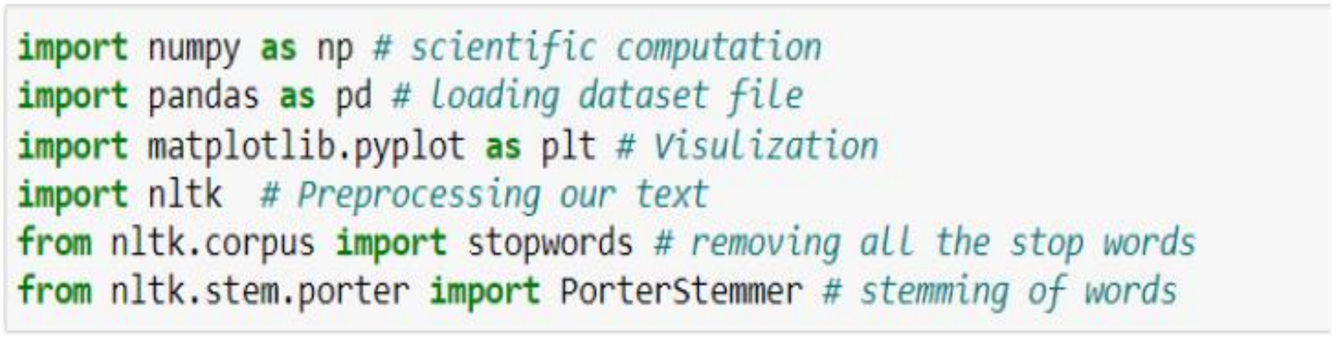
- •
  Lack of filters that have the capacity to dynamically update the feature space. Majority of the existing spam filters are unable to incrementally add or delete features without re-creating the model totally to keep abreast of current trends in email spam filtering.

- •
  The need to apply deep learning to spam filtering in order to exploit its numerous processing layers and many levels of abstraction to learn representations of data.

- •
  The inevitable need to design spam filters with lower processing and classification time using Graphics Processing Unit (GPU) and Field-Programmable Gate Array (FPGA) with their advantage of low power consumption, reconfigurability, and real-time processing capability for real-time processing and classification.

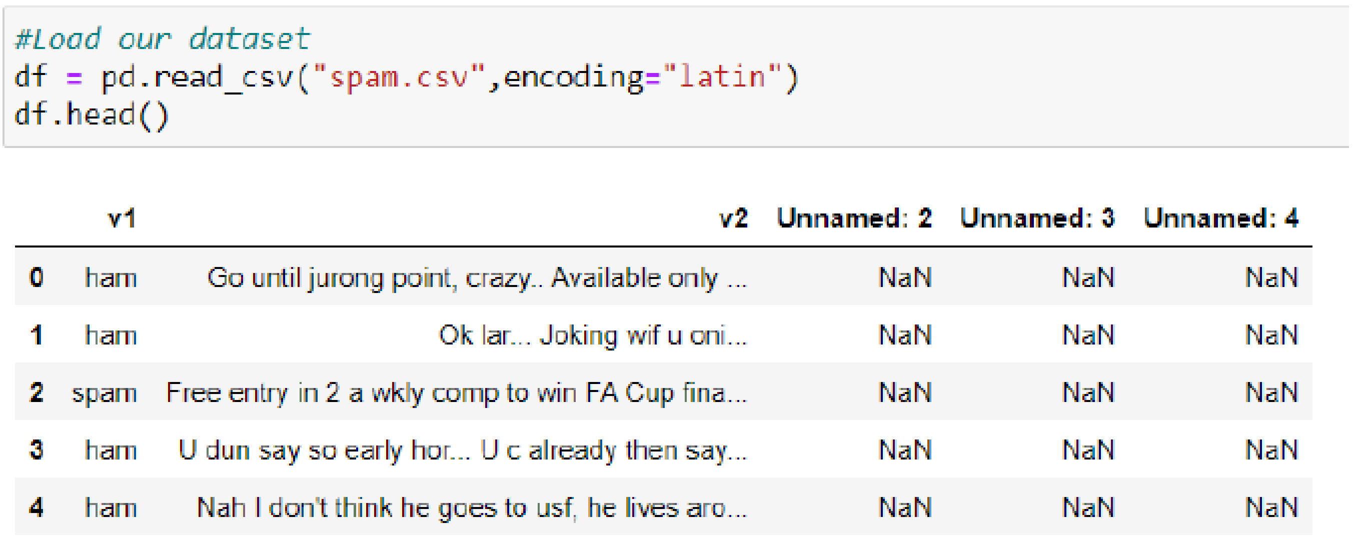
Appendix

Importing the libraries



```
import numpy as np # scientific computation
import pandas as pd # loading dataset file
import matplotlib.pyplot as plt # Visulization
import nltk  # Preprocessing our text
from nltk.corpus import stopwords # removing all the stop words
from nltk.stem.porter import PorterStemmer # stemming of words
```

Read the Dataset

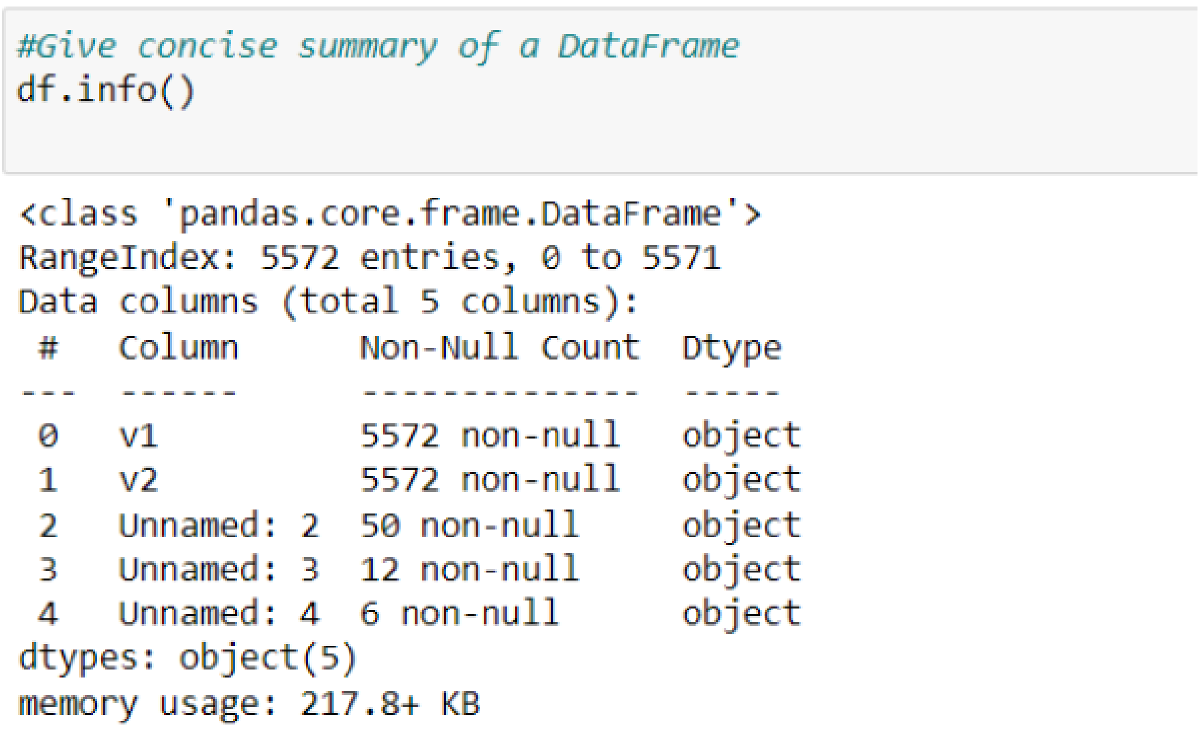


```python
#Load our dataset
df = pd.read_csv("spam.csv",encoding="latin")
df.head()
```

	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

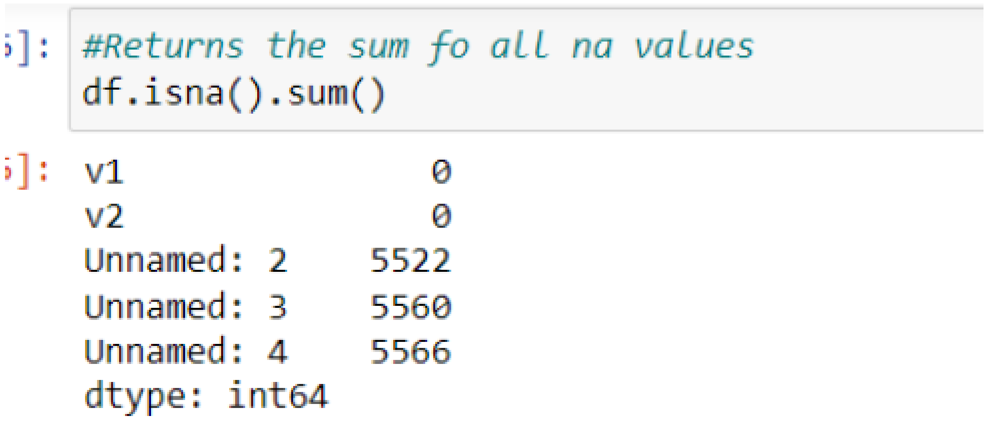
Data Preparation

● Handling missing values



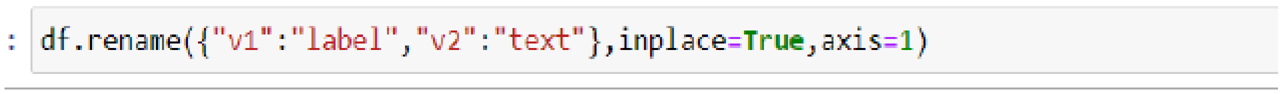
```python
#Give concise summary of a DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

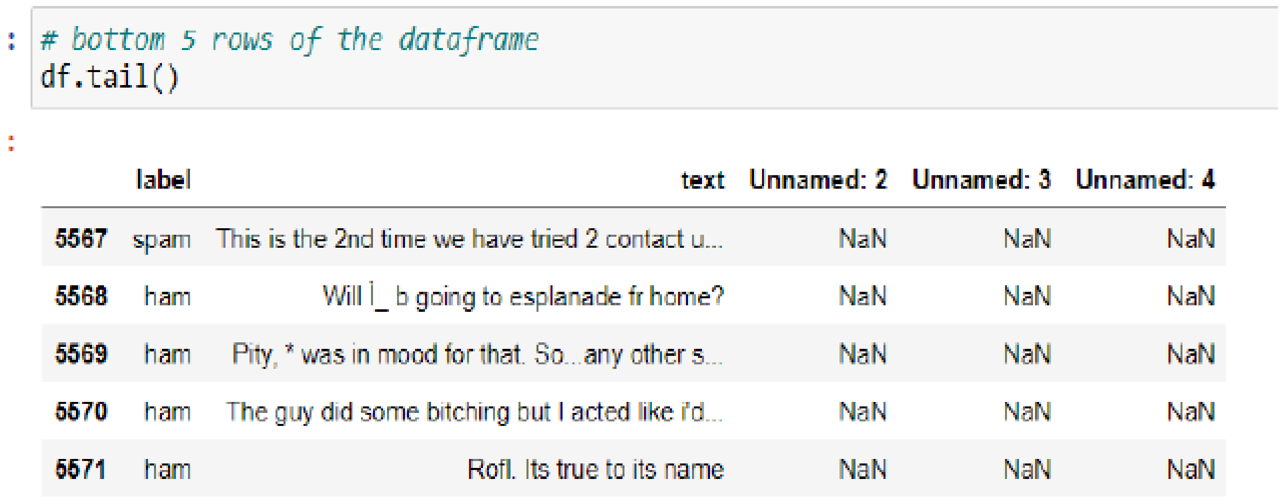


```
#Returns the sum fo all na values
df.isna().sum()
```

```
v1              0
v2              0
Unnamed: 2   5522
Unnamed: 3   5560
Unnamed: 4   5566
dtype: int64
```



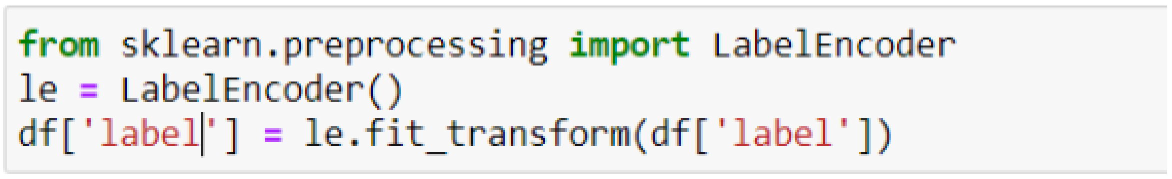
```
df.rename({"v1":"label","v2":"text"},inplace=True,axis=1)
```



```
# bottom 5 rows of the dataframe
df.tail()
```

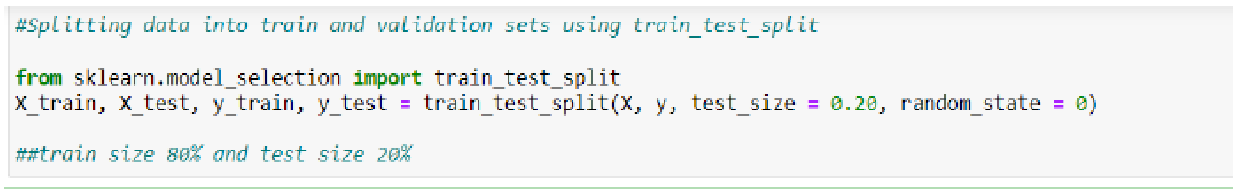
	label	text	Unnamed: 2	Unnamed: 3	Unnamed: 4
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will Ì_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

● **Handling categorical data**



```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['label'] = le.fit_transform(df['label'])
```

● **Handling Imbalance Data**



```
#Splitting data into train and validation sets using train_test_split

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

##train size 80% and test size 20%
```

```
print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train == 0)))

# import SMOTE module from imblearn library
# pip install imblearn (if you don't have imblearn in your system)
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())

print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res == 0)))
```

Cleaning the text data

```
:  nltk.download("stopwords")
```

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

```
:  True
```

```
:  import nltk
   from nltk.corpus import stopwords
   from nltk.stem import PorterStemmer
```

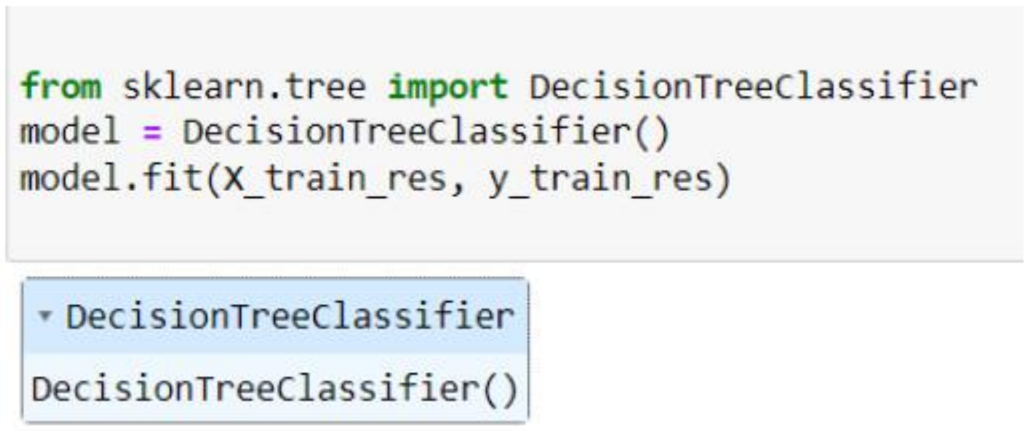
```
:  import re
   corpus = []
   length = len(df)
```

```
:  for i in range(0,length):
       text = re.sub("[^a-zA-Z0-9]"," ",df["text"][i])
       text = text.lower()
       text = text.split()
       pe = PorterStemmer()
       stopword = stopwords.words("english")
       text = [pe.stem(word) for word in text if not word in set(stopword)]
       text = " ".join(text)
       corpus.append(text)
```

Univariate analysis

```
df["label"].value_counts().plot(kind="bar",figsize=(12,6))
plt.xticks(np.arange(2), ('Non spam', 'spam'),rotation=0);
```

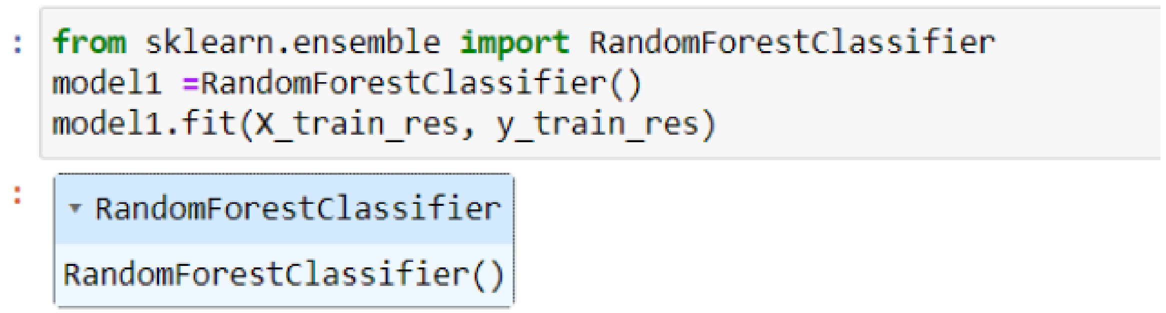
Model Building



```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train_res, y_train_res)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

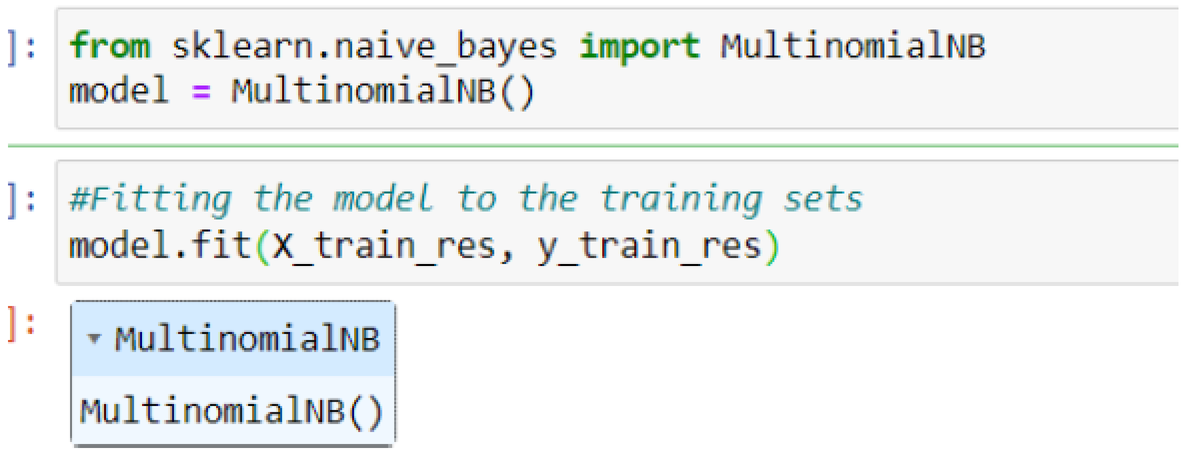
Random forest model



```
: from sklearn.ensemble import RandomForestClassifier
  model1 =RandomForestClassifier()
  model1.fit(X_train_res, y_train_res)
```

```
: ▾ RandomForestClassifier
  RandomForestClassifier()
```

Naïve Bayes model

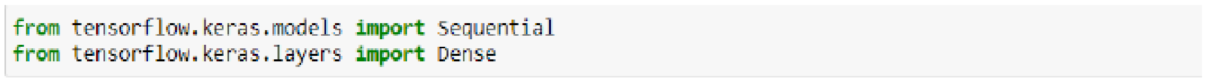


```
]: from sklearn.naive_bayes import MultinomialNB
   model = MultinomialNB()
```

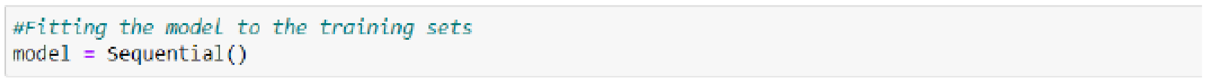
```
]: #Fitting the model to the training sets
   model.fit(X_train_res, y_train_res)
```

```
]: ▾ MultinomialNB
   MultinomialNB()
```

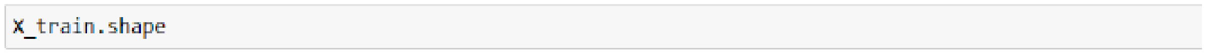
ANN model



```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

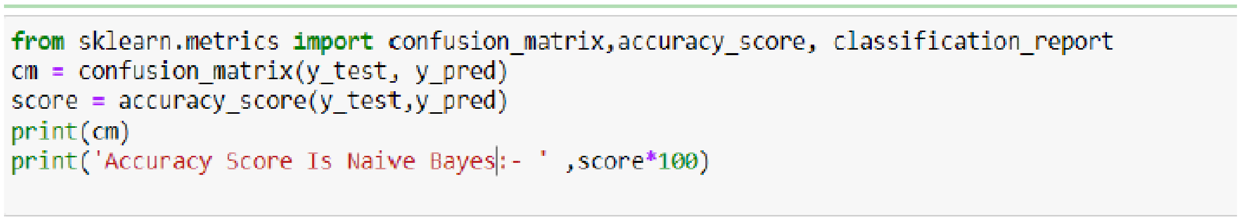


```
#Fitting the model to the training sets
model = Sequential()
```



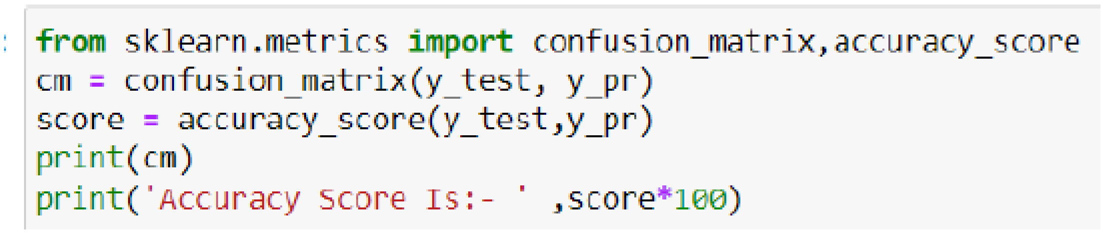
```
x_train.shape
```

Testing model with multiple evaluation metrics



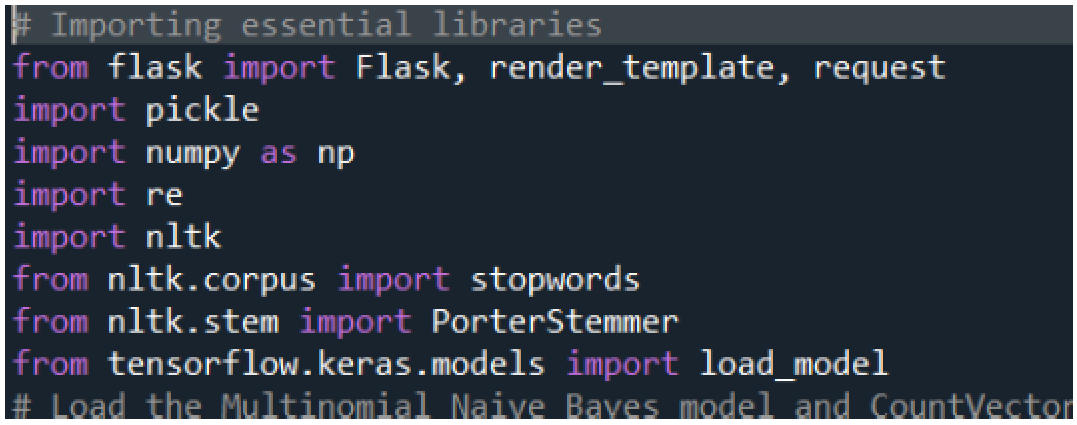
```
from sklearn.metrics import confusion_matrix,accuracy_score, classification_report
cm = confusion_matrix(y_test, y_pred)
score = accuracy_score(y_test,y_pred)
print(cm)
print('Accuracy Score Is Naive Bayes:- ' ,score*100)
```

Comparing model accuracy before & after applying hyperparameter tuning



```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pr)
score = accuracy_score(y_test,y_pr)
print(cm)
print('Accuracy Score Is:- ' ,score*100)
```

Build Python code



```
# Importing essential libraries
from flask import Flask, render_template, request
import pickle
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from tensorflow.keras.models import load_model
# Load the Multinomial Naive Bayes model and CountVector
```