- Name: Abhishek V Sutrave
- Batch: MARCH-P-3
- Role:Data Science Intern
- Company:OASIS INFOBYTE
- TASK4 :EMAIL SPAM DETECTION WITH MACHINE LEARNING
- Objective : Utilize Machine Learning techniques so that it can accurately categorize emails.
- Develop an Email Spam Detection System.

In [1]:
```
#Importing required libraries
import pandas as pd
import numpy as np
#Importing warning modules
import warnings
warnings.filterwarnings('ignore')
#Importing necessary function to split data into training and testing sets
from sklearn.model_selection import train_test_split
#Bringing in visualization libraries for plotting graphs
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#Reading the CSV file into a DataFrame
import pandas as pd

# Load the dataset
data__1= pd.read_csv("C:/Users/Abhishek/Desktop/OASIS INFOBYTE/spam.csv",encoding='ISO-8859-1')

data__1.head(4)
```

Out[2]:

	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

Conducting a descriptive analysis to understand the variables and their associated values

In [46]:
```
#Examining the dataset's dimensions

data__1.shape
```

Out[46]:
```
(5572, 5)
```

In [14]:
```
#Display dataset information
data__1.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
```

In [15]:
```
#Display column names
print(data__1.columns)
```

```
Index(['Category', 'Message', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')
```

In [17]:
```
#verifying for duplicate entries in dataset
data__1.duplicated().sum()
```

Out[17]:
```
np.int64(403)
```

In [18]:
```
#Checking for missing values in the dataset
data__1.isna().sum()
```

Out[18]:
```
v1              0
v2              0
Unnamed: 2   5522
Unnamed: 3   5560
Unnamed: 4   5566
dtype: int64
```

In [19]:
```
#Dropping duplicate rows
data__1.drop_duplicates(keep='first',inplace=True)
data__1.shape
```

Out[19]:
```
(5169, 5)
```

In [20]:
```
#Dropping irrelevant columns from the Dataframe using their names
data__1.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
#Displaying the first few record after cleaning up the columns
data__1.head()
```

Out[20]:

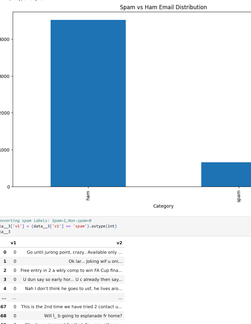
	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

In [27]:
```
#Analyzing the frequency of spam and non spam mails
spam=data__1['v1'].value_counts()
print(spam)
```

```
v1
ham     4516
spam     653
Name: count, dtype: int64
```

In [28]:
```
#Plotting the frequency of spam and ham emails in bar chart
data__1['v1'].value_counts().plot(kind='bar',figsize=(12,7),title="Spam vs Ham Email Distribution",xlabel='Category',ylabel='Frequency')
print(data__1['v1'].head())
```

```
0    ham
1    ham
2   spam
3    ham
4    ham
Name: v1, dtype: object
```



In [29]:
```
#converting spam labels: Spam=1,Non-spam=0
data__1['v1'] = (data__1['v1'] == 'spam').astype(int)
data__1
```

Out[29]:

	v1	v2
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	1	This is the 2nd time we have tried 2 contact u...
5568	0	Will_ b going to esplanade fr home?
5569	0	Pity_ * was in mood for that. So... any other s...
5570	0	The guy did some bitching but I acted like i'd...
5571	0	Rofl. Its true to its name

5169 rows × 2 columns

In [30]:
```
#Displaying sample spam messages
print("Here are some examples of spam messages:")
print(data__1[data__1['v1'] == 1]['v2'].head())
```

```
Here are some examples of spam messages:
2    Free entry in 2 a wkly comp to win FA Cup fina...
5    FreeMsg Hey there darling it's been 3 week's n...
8    WINNER!! As a valued network customer you have...
9    Had your mobile 11 months or more? U R entitle...
11   SIX chances to win CASH from 100 to 20,000 pos...
Name: v2, dtype: object
```

In [31]:
```
#Displaying sample non-spam (ham) messages
print("Here are some examples of ham messages:")
print(data__1[data__1['v1'] == 0]['v2'].head())
```

```
Here are some examples of ham messages:
0    Go until jurong point, crazy.. Available only ...
1                        Ok lar... Joking wif u oni...
3    U dun say so early hor... U c already then say...
4    Nah I don't think he goes to usf, he lives aro...
6    Even my brother is not like to speak with me. ...
Name: v2, dtype: object
```

In [32]:
```
data__1.describe()
```

Out[32]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
count	5572	5572	50	12	6
unique	2	5169	43	10	5
top	ham	Sorry, I'll call later bt not his girlfrnd... G o o d n i g h t . . .@"	MK17 92H. 450Ppw 16"	GNT:-)"	
freq	4825	30	3	2	2

In [33]:
```
data__1.tail(5)#Provides last 6 entries
```

Out[33]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
5566	spam	REMINDER FROM O2: To get 2.50 pounds free call...	NaN	NaN	NaN
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will_ 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

In [52]:
```
#Returns a boolean series
data__1.duplicated()
```

Out[52]:
```
0       False
1       False
2       False
3       False
4       False
        ...  
5567    False
5568    False
5569    False
5570    False
5571    False
Length: 5572, dtype: bool
```

In [7]:
```
data__1.duplicated().sum()
```

Out[7]:
```
np.int64(403)
```

In [9]:
```
#Returns the total number of elements in dataset
data__1.size
```

Out[9]:
```
27860
```

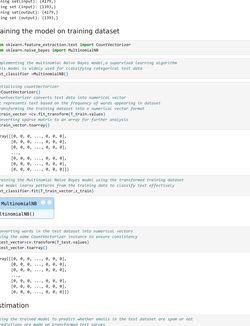
In [10]:
```
data__1.columns=data__1.columns.str.replace("v1","Category").str.replace("v2","Message")
```

In [36]:
```
data__1.head()
```

Out[36]:

	Category	Message	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

In [39]:
```
counts= data__1['Category'].value_counts()
plt.plot(counts, labels=["ham","Not_Safe"],autopct="%0.4f", colors=["blue","red"])
plt.title("Distribution of message categories")
plt.show()
```



Developing the Model

In [7]:
```
from sklearn.model_selection import train_test_split
```

In [15]:
```
#Here the predictors represent mails and response denotes whether email is spam or ham
predictors=data__1['v2']
response=data__1['v1']
```

In [16]:
```
#Dividing the dataset into training and testing sets for model evaluation
#Allocating 75% of the data for training
X_train,X_test,y_train,y_test=train_test_split(predictors,response,train_size=0.75)
#Displaying the dimensions of the training and testing datasets
print('Training set(input):',X_train.shape)
print('Testing set (input):',X_test.shape)
print('Training set(output):',y_train.shape)
print('Testing set (output):',y_test.shape)
```

```
Training set(input): (4179,)
Testing set (input): (1393,)
Training set(output): (4179,)
Testing set (output): (1393,)
```

Training the model on training dataset

In [17]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
```

In [18]:
```
#Implementing the multinomial Naive Bayes model,a supervised learning algorithm
#This model is widely used for classifying categorical text data
text_classifier =MultinomialNB()
```

In [19]:
```
#Initializing countVectorizer
cv=CountVectorizer()
#This vectorizer converts text data into numerical vector
#It represents text based on the frequency of words appearing in dataset
#transforming the training dataset into a numerical vector format
X_train_vector =cv.fit_transform(X_train.values)
#Converting sparse matrix to an array for further analysis
X_train_vector.toarray()
```

Out[19]:
```
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]])
```

In [21]:
```
#Training the Multinomial Naive Bayes model using the transformed training dataset
#The model learns patterns from the training data to classify text effectively
text_classifier.fit(X_train_vector,y_train)
```

Out[21]:
```
▾ MultinomialNB ▸ ▸
MultinomialNB()
```

In [22]:
```
#Converting words in the test dataset into numerical vectors
#Using the same CountVectorizer instance to ensure consistency
X_test_vector=cv.transform(X_test.values)
X_test_vector.toarray()
```

Out[22]:
```
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]])
```

Estimation

In [23]:
```
#Using the trained model to predict whether emails in the test dataset are spam or not
#Predictions are made on transformed test values
y_pred=text_classifier.predict(X_test_vector)
y_pred
```

Out[23]:
```
array(['ham', 'spam', 'ham', ..., 'ham', 'ham', 'ham'], dtype='<U4')
```

Evaluating the model

In [38]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

In [25]:
```
#Generating the confusion matrix
conf_matrix=confusion_matrix(y_test,y_pred)
print("Confusion Matrix:\n",conf_matrix)
```

```
Confusion Matrix:
[[1189   5]
 [  13  186]]
```

In [42]:
```
#Generating a detailed classification report
class_report=classification_report(y_test,y_pred)
print("Classification Report:\n",class_report)
```

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

         ham       0.99      1.00      0.99      1194
        spam       0.97      0.93      0.95       199

    accuracy                           0.99      1393
   macro avg       0.98      0.96      0.97      1393
weighted avg       0.99      0.99      0.99      1393
```

In [28]:
```
#Compute balanced accuracy
from sklearn.metrics import balanced_accuracy_score
balanced_accuracy=balanced_accuracy_score(y_test,y_pred)
print("Balanced Accuracy score",balanced_accuracy)
```

```
Balanced Accuracy score: 0.9627951264436238
```

In [29]:
```
from sklearn.metrics import precision_score
precision=precision_score(y_test,y_pred,pos_label="spam")
print("Precision",precision)
```

```
Precision: 0.9737151248164786
```

In [ ]: