In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [2]:
```
df=pd.read_csv('C:/Users/md mejbah uddin/Downloads/CypherByte/archive/spam mail.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	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

In [4]:
```
#Drop all columns except spam and text
df=df.iloc[:,:2]
```

In [5]:
```
df.head()
```

Out[5]:

	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 [6]:
```
df.info()
```

```
<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
```

In [7]:
```
df.describe()
```

Out[7]:

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

In [8]:
```
df.shape
```

Out[8]:
```
(5572, 2)
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
v1    0
v2    0
dtype: int64
```

In [10]:
```
df.columns
```

Out[10]:
```
Index(['v1', 'v2'], dtype='object')
```

In [11]:
```
# label spam mail as 0;  ham mail as 1;

df.loc[df['v1'] == 'spam', 'v1',] = 0
df.loc[df['v1'] == 'ham', 'v1',] = 1
```

In [12]:
```
# separating the data as texts and label

X = df['v2']

Y = df['v1']
```

In [13]:
```
X
```

Out[13]:
```
0       Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
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...
                              ...
5567    This is the 2nd time we have tried 2 contact u...
5568                Will Ì_ b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
```

```
5571                          Rofl. Its true to its name
Name: v2, Length: 5572, dtype: object
```

In [14]:
```
Y
```

Out[14]:
```
0       1
1       1
2       0
3       1
4       1
       ..
5567    0
5568    1
5569    1
5570    1
5571    1
Name: v1, Length: 5572, dtype: object
```

In [15]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=
```

In [16]:
```
print(X.shape)
```

```
(5572,)
```

In [17]:
```
print(X_train.shape)
```

```
(4179,)
```

In [18]:
```
print(X_test.shape)
```

```
(1393,)
```

In [19]:
```
# transform the text data to feature vectors that can be used as input

feature_extraction =TfidfVectorizer(min_df = 1, stop_words='english', lowercase='True')
```

In [20]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

# Create a TfidfVectorizer object with the lowercase parameter set to True
vectorizer = TfidfVectorizer(lowercase=True)

# Fit the TfidfVectorizer object to the training data
vectorizer.fit(X_train)

# Transform the training data into a sparse matrix of feature vectors
X_train_features = vectorizer.transform(X_train)
X_test_features = vectorizer.transform(X_test)
# Print the shape of the feature matrix
print(X_train_features.shape)
```

```
(4179, 7480)
```

In [21]:
```
#X_train_features = feature_extraction.fit_transform(X_train)
```

In [22]:
```
Y_train = Y_train.astype('int')
Y_test = Y_test.astype('int')
```

In [23]:
```
#naive_bayes
gnb=MultinomialNB()
gnb.fit(X_train_features,Y_train)
MultinomialNB()
gnb.score(X_test_features,Y_test)
```

Out[23]:
```
0.9562096195262024
```

In [24]:
```
bnb=BernoulliNB()
bnb.fit(X_train_features,Y_train)
bnb.score(X_test_features,Y_test)
```

Out[24]:
```
0.9734386216798278
```

In [25]:
```
#LogisticRegression
model = LogisticRegression()
model.fit(X_train_features, Y_train)
prediction_on_training_data = model.predict(X_train_features)
accuracy_on_training_data = accuracy_score(Y_train, prediction_on_training_data)
print('Accuracy on training data : ', accuracy_on_training_data)
```

```
Accuracy on training data :  0.9741564967695621
```

SUBMITTED BY-

MD.MEJBAH UDDIN