Date	02-11-2022
Team ID	PNT2022TMID43024
Project Name	Real time communication powered by Ai for specially Abled

ASSIGNMENT-4

Problem Statement:

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollar industry. At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones. Due to Spam SMS, Mobile service providers suffer from some sort of financial problems as well as it reduces calling time for users. Unfortunately, if the user accesses such Spam SMS they may face the problem of virus or malware. When SMS arrives at mobile it will disturb mobile user privacy and concentration. It may lead to frustration for the user. So Spam SMS is one of he major issues in the wireless communication world and it grows day by day.

Solution:

Dataset: Spam Data = 13.21% , ham Data = 86.78%

Import Python libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.naive_bayes import MultinomialNB
```

Import SMS Data

```
In [2]:
df = pd.read_csv('../input/spamraw.csv')
```

```
df.head()
```

Out[2]:

	type	text
0	ham	Hope you are having a good week. Just checking in
1	ham	K..give back my thanks.
2	ham	Am also doing in cbe only. But have to pay.
3	spam	complimentary 4 STAR Ibiza Holiday or £10,000 ...
4	spam	okmail: Dear Dave this is your final notice to...

Exploratory Data Analysis (EDA)

In [3]:

```
#Find count and unique messages count of all the messages
df.describe()
```

Out[3]:

	type	text
count	5559	5559
unique	2	5156
top	ham	Sorry, I'll call later

	type	text
freq	4812	30

```
#Extract SPAM messages
spam_messages = df[df["type"]=="spam"]
spam_messages.head() #Display first 5 rows of SPAM messages
```

Out[4]:

	type	text
3	spam	complimentary 4 STAR Ibiza Holiday or £10,000 ...
4	spam	okmail: Dear Dave this is your final notice to...
8	spam	Marvel Mobile Play the official Ultimate Spide...
19	spam	U can WIN £100 of Music Gift Vouchers every we...
34	spam	U have won a nokia 6230 plus a free digital ca...

```
#Find count and unique messages count of SPAM messages.
spam_messages.describe()
```

Out[5]:

	type	text
count	747	747

	type	text
unique	1	653
top	spam	Please call our customer service representativ...
freq	747	4

In [6]:

linkcode

```
#Plot the counts of HAM (non SPAM) vs SPAM
sns.countplot(data = df, x= df["type"]).set_title("Amount of spam and no-spam
messages")
plt.show()
```

Splitting the SMS data into Test and Train data

In [7]:

```
data_train, data_test, labels_train, labels_test =
train_test_split(df.text,df.type,test_size=0.2,random_state=0)
print("data_train, labels_train : ",data_train.shape, labels_train.shape)
print("data_test, labels_test: ",data_test.shape, labels_test.shape)
```

```
data_train, labels_train :  (4447,) (4447,)
data_test, labels_test:  (1112,) (1112,)
```

Extraction & CountVectorize

The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

In [8]:

```
vectorizer = CountVectorizer()
#fit & transform
# fit: build dict (i.e. word->wordID)
# transform: convert document (i.e. each line in the file) to word vector
data_train_count = vectorizer.fit_transform(data_train)
data_test_count  = vectorizer.transform(data_test)
```

Modelling & training

Multinomial Naive Bayes is a specialized version of Naive Bayes that is designed more for text documents. Whereas simple naive Bayes would model a document as the presence and absence of particular words, multinomial naive bayes explicitly models the word counts and adjusts the underlying calculations to deal with in.

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

```
In [9]:
clf = MultinomialNB()
clf.fit(data_train_count, labels_train)
predictions = clf.predict(data_test_count)
predictions
```

```
Out[9]:
array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'ham'],
dtype='<U4')
```

Results and Accuracy

```
In [10]:
print ("accuracy_score : ", accuracy_score(labels_test, predictions))
```

```
accuracy_score :  0.9901079136690647
```

Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.

```
In [11]:
print ("confusion_matrix : \n", confusion_matrix(labels_test, predictions))
```

```
confusion_matrix :
 [[979   0]
 [ 11 122]]
```

```
In [12]:
print (classification_report(labels_test, predictions))
```

```
              precision    recall  f1-score   support

         ham       0.99      1.00      0.99       979
        spam       1.00      0.92      0.96       133
```

```
   micro avg       0.99      0.99      0.99      1112
   macro avg       0.99      0.96      0.98      1112
weighted avg       0.99      0.99      0.99      1112
```