This program detects if an email is spam or not/ham .

Import necessary libraries

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
```

```
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
import string
```

Load the dataset

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving email.csv to email.csv

```
#Read the csv file
df = pd.read_csv("email.csv")
#Print the first 5 rows of data
df.head(5)
```

	Category	Message
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...

```
#Print the shape(Get the number of rows and columns)
df.shape
```

```
(5572, 2)
```

```
#Get the column names
df.columns
```

```
Index(['Category', 'Message'], dtype='object')
```

```
#check for duplicates and remove them
df.drop_duplicates(inplace = True)
```

```
#show the new shape(new number of rows and columns)
df.shape
```

```
(5157, 2)
```

```
#show the number of missing data (like - NAN, Nan, na) data for each column
df.isnull().sum()
```

```
Category    0
Message     0
dtype: int64
```

Download the stopwords package

```
#Download the stopwords package
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
def process_text(text):

  #1.Remove the punctuation
  #2.Remove the stopwords
  #3.Return a list of clean text words

  #1
  nopunc = [char for char in text if char not in string.punctuation]
  nopunc = ''.join(nopunc)

  #2
  clean_words = [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]

  #3
  return clean_words
```

Tokenization

```
#Show the tokenization (a list of tokens also called lemmas)
df['Message'].head().apply(process_text)
```

```
0    [Go, jurong, point, crazy, Available, bugis, n...
1                        [Ok, lar, Joking, wif, u, oni]
2    [Free, entry, 2, wkly, comp, win, FA, Cup, fin...
3        [U, dun, say, early, hor, U, c, already, say]
4    [Nah, dont, think, goes, usf, lives, around, t...
Name: Message, dtype: object
```

```
#Convert a collection of text to a matrix of tokens
from sklearn.feature_extraction.text import CountVectorizer
messages_bow = CountVectorizer(analyzer=process_text).fit_transform(df['Message'])
```

```
#Split the data into 80% training and 20% testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(messages_bow, df['Category'], test_size=0.20, random_state=0)
```

```
#Get the shape of messages_bow
messages_bow.shape
```

```
(5157, 11422)
```

Creating and training the Naive Bayes Classifier

```
#Create and train the Naive Bayes Classifier
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB().fit(X_train, y_train)
```

```
#Print the predictions
print(classifier.predict(X_train))

#Print the actual values
print(y_train.values)
```

```
['ham' 'spam' 'ham' ... 'ham' 'ham' 'ham']
['ham' 'spam' 'ham' ... 'ham' 'ham' 'ham']
```

Model Evaluation on the both training and testing data

Training dataset

```
#Evaluate the model on the training dataset
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(X_train)
print(classification_report(y_train, pred))
print()
print('Confusion Matrix: \n', confusion_matrix(y_train, pred))
print('Accuracy:', accuracy_score(y_train, pred))
```

```
                  precision    recall  f1-score   support

             ham       1.00      1.00      1.00      3619
            spam       0.98      0.97      0.98       506

        accuracy                           0.99      4125
       macro avg       0.99      0.99      0.99      4125
    weighted avg       0.99      0.99      0.99      4125


    Confusion Matrix:
     [[3611    8]
     [  13  493]]
    Accuracy: 0.9949090909090909
```

After evaluating the model om the training data the model got an accuracy of 99.49%

```
#Print the predictions
print(classifier.predict(X_test))

#Print the actual values
print(y_test.values)
```

```
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
```

Testing dataset

```
#Evaluate the model on the testing dataset
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(X_test)
print(classification_report(y_test, pred))
print()
print('Confusion Matrix: \n', confusion_matrix(y_test, pred))
print('Accuracy:', accuracy_score(y_test, pred))
```

```
                  precision    recall  f1-score   support

             ham       0.99      0.97      0.98       897
            spam       0.81      0.93      0.86       135

        accuracy                           0.96      1032
       macro avg       0.90      0.95      0.92      1032
    weighted avg       0.96      0.96      0.96      1032


    Confusion Matrix:
     [[867  30]
     [ 10 125]]
    Accuracy: 0.9612403100775194
```

After evaluating the model om the testing data the model got an accuracy of 96.12%