Email Spam classification

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
import string
```

Read Data

In [2]:
```
df = pd.read_csv("emails.csv")
df.head()
df.shape
df.columns
df.drop_duplicates(inplace=True)
print(df.shape)
```

```
(5695, 2)
```

Data Preprocessing

In [3]:
```
print(df.isnull().sum())
nltk.download("stopwords")
def process(text):
    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)

    clean = [word for word in nopunc.split() if word.lower() not in stopwor
    return clean

df['text'].head().apply(process)
```

```
text    0
spam    0
dtype: int64

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

Out[3]:
```
0    [Subject, naturally, irresistible, corporate, ...
1    [Subject, stock, trading, gunslinger, fanny, m...
2    [Subject, unbelievable, new, homes, made, easy...
3    [Subject, 4, color, printing, special, request...
4    [Subject, money, get, software, cds, software,...
Name: text, dtype: object
```

Feature Engineering

In [4]:
```
from sklearn.feature_extraction.text import CountVectorizer
message = CountVectorizer(analyzer=process).fit_transform(df['text'])
```

Split Data

In [5]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(message, df['spam'], test_s
print(message.shape)
```

```
(5695, 37229)
```

Model Training

In [6]:
```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB().fit(xtrain, ytrain)
print(classifier.predict(xtrain))
print(ytrain.values)
```

```
[0 0 0 ... 0 0 0]
[0 0 0 ... 0 0 0]
```

Model Evaluation

In [7]:
```
from sklearn.metrics import classification_report, confusion_matrix, accura
pred = classifier.predict(xtrain)
print(classification_report(ytrain, pred))
print()
print("Confusion Matrix: \n", confusion_matrix(ytrain, pred))
print("Accuracy: \n", accuracy_score(ytrain, pred))

print(classifier.predict(xtest))
print(ytest.values)

pred = classifier.predict(xtest)
print(classification_report(ytest, pred))
print()
print("Confusion Matrix: \n", confusion_matrix(ytest, pred))
print("Accuracy: \n", accuracy_score(ytest, pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3457
           1       0.99      1.00      0.99      1099

    accuracy                           1.00      4556
   macro avg       0.99      1.00      1.00      4556
weighted avg       1.00      1.00      1.00      4556


Confusion Matrix:
 [[3445   12]
 [   1 1098]]
Accuracy:
 0.9971466198419666
[1 0 0 ... 0 0 0]
[1 0 0 ... 0 0 0]
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       870
           1       0.97      1.00      0.98       269

    accuracy                           0.99      1139
   macro avg       0.98      0.99      0.99      1139
weighted avg       0.99      0.99      0.99      1139


Confusion Matrix:
 [[862   8]
 [  1 268]]
Accuracy:
 0.9920983318700615
```

Accuracy

In [8]:
```
b=accuracy_score(ytest, pred)*100
print(b)
```

```
99.20983318700614
```

In [ ]: