In [7]:
```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [10]:
```
df=pd.read_csv("spam.csv",encoding="latin-1")
```

In [11]:
```
df.head()
```

Out[11]:

	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 [13]:
```
df=df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)
```

In [14]:
```
df.head()
```

Out[14]:

	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 [53]:
```
df.columns =['label','text']
```

In [54]:
```
df.head()
```

Out[54]:

	label	text
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 [55]:
```
df.colums
```

Out[55]:
```
['class', 'text']
```

In [35]:
```
nltk.download('stopwords')
```

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

Out[35]:
```
True
```

In [36]:
```
stop_words=set(stopwords.words('english'))
stemmer=PorterStemmer()
```

In [37]:
```
stop_words
```

Out[37]:
```
{'a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'ain',
 'all',
 'am',
 'an',
 'and',
 'any',
 'are',
 'aren',
 "aren't",
 'as',
 'at',
 'be',
 'because',
 'been',
 'before',
 'being',
 'below',
 'between',
 'both',
 'but',
 'by',
 'can',
 'couldn',
 "couldn't",
 'd',
 'did',
 'didn',
 "didn't",
 'do',
 'does',
 'doesn',
 "doesn't",
 'doing',
 'don',
 "don't",
 'down',
 'during',
 'each',
 'few',
 'for',
 'from',
 'further',
 'had',
 'hadn',
 "hadn't",
 'has',
 'hasn',
 "hasn't",
 'have',
 'haven',
```

```
"haven't",
'having',
'he',
"he'd",
"he'll",
"he's",
'her',
'here',
'hers',
'herself',
'him',
'himself',
'his',
'how',
'i',
"i'd",
"i'll",
"i'm",
"i've",
'if',
'in',
'into',
'is',
'isn',
"isn't",
'it',
"it'd",
"it'll",
"it's",
'its',
'itself',
'just',
'll',
'm',
'ma',
'me',
'mightn',
"mightn't",
'more',
'most',
'mustn',
"mustn't",
'my',
'myself',
'needn',
"needn't",
'no',
'nor',
'not',
'now',
'o',
'of',
'off',
'on',
'once',
'only',
```

```
'or',
'other',
'our',
'ours',
'ourselves',
'out',
'over',
'own',
're',
's',
'same',
'shan',
"shan't",
'she',
"she'd",
"she'll",
"she's",
'should',
"should've",
'shouldn',
"shouldn't",
'so',
'some',
'such',
't',
'than',
'that',
"that'll",
'the',
'their',
'theirs',
'them',
'themselves',
'then',
'there',
'these',
'they',
"they'd",
"they'll",
"they're",
"they've",
'this',
'those',
'through',
'to',
'too',
'under',
'until',
'up',
've',
'very',
'was',
'wasn',
"wasn't",
'we',
"we'd",
```

```
 "we'll",
 "we're",
 "we've",
 'were',
 'weren',
 "weren't",
 'what',
 'when',
 'where',
 'which',
 'while',
 'who',
 'whom',
 'why',
 'will',
 'with',
 'won',
 "won't",
 'wouldn',
 "wouldn't",
 'y',
 'you',
 "you'd",
 "you'll",
 "you're",
 "you've",
 'your',
 'yours',
 'yourself',
 'yourselves'}
```

In [56]:
```
def clean_text(text):
    words=text.lower().split()
    clean_words=[]
    for word in words:
       if word not in stop_words:
          clean_words.append(stemmer.stem(word))
    return ' '.join(clean_words)
```

In [57]:
```
df['text'] = df['text'].apply(clean_text)
```

In [59]:
```
df.head()
```

Out[59]:

	label	text
0	ham	go jurong point, crazy.. avail bugi n great wo...
1	ham	ok lar... joke wif u oni...
2	spam	free entri 2 wkli comp win fa cup final tkt 21...
3	ham	u dun say earli hor... u c alreadi say...
4	ham	nah think goe usf, live around though

In [62]:
```
from sklearn.preprocessing import StandardScaler,LabelEncoder
```

In [63]:
```
le=LabelEncoder()
df['class']=le.fit_transform(df['label'])
```

In [64]:
```
df.head()
```

Out[64]:

	label	text	class
0	ham	go jurong point, crazy.. avail bugi n great wo...	0
1	ham	ok lar... joke wif u oni...	0
2	spam	free entri 2 wkli comp win fa cup final tkt 21...	1
3	ham	u dun say earli hor... u c alreadi say...	0
4	ham	nah think goe usf, live around though	0

In [65]:
```
ve=TfidfVectorizer(stop_words='english')
x=ve.fit_transform(df['text'])
y=df['class']
```

In [68]:
```
print(x[5])
```

```
  (0, 5774)     0.3468599318063216
  (0, 609)      0.22568478152650728
  (0, 6190)     0.17462717493654115
  (0, 1894)     0.3468599318063216
  (0, 7850)     0.24810526628029425
  (0, 6926)     0.32608083582339936
  (0, 3180)     0.250571322752491
  (0, 4282)     0.1689505569623337
  (0, 7777)     0.23004725878935042
  (0, 7635)     0.1961332638083403
  (0, 2305)     0.3123939951970498
  (0, 3567)     0.19961208978703432
  (0, 3137)     0.2813920631147106
  (0, 6651)     0.29046897910131386
  (0, 5088)     0.1623396974479272
```

In [72]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
```

In [73]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_sta

# Train
model = MultinomialNB()
model.fit(x_train, y_train)

# Predict
y_pred = model.predict(x_test)
```

```
# Evaluate
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       949
           1       1.00      0.75      0.86       166

    accuracy                           0.96      1115
   macro avg       0.98      0.87      0.92      1115
weighted avg       0.96      0.96      0.96      1115
```

In [ ]: