In [20]:

```
import pandas as pd
import re
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier
```

In [16]:

```
df=pd.read_csv("f:/dataset/sentiment/Restaurant_Reviews.txt",sep="\t")
```

In [17]:

```
doc1='f5o@od is # good & good!'
doc2='& Food # is * tasty'
doc3='Quality is Good'
doc4='food is not good'
doc5='servi89ce is Poor poor means very poor'
doc6='it is to_o costly'
doc7='che^ap quality'

corpus=[doc1,doc2,doc3,doc4,doc5,doc6,doc7]
target=['pos','pos','pos','neg','neg','neg','neg']
```

In [21]:

```
spwords=list(ENGLISH_STOP_WORDS)
spwords.remove('not')

def cleaning(doc):
    doc=doc.lower()
    doc=re.sub("[^a-z ]","",doc)
    wordslist=doc.split()
    newdoc=""
    for word in wordslist:
        if word not in spwords:
            newdoc=newdoc+word+" "
    return newdoc.strip()

corpus_new=list(map(cleaning,corpus))
cv=CountVectorizer()
X=cv.fit_transform(corpus_new).toarray()
model=RandomForestClassifier()
model.fit(X,target)
```

Out[21]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [23]:

```
sample1='Food quality is not good$'
sample2='awesome food'
corpus_test=[sample1,sample2]
corpus_test_new=list(map(cleaning,corpus_test))
X_test=cv.transform(corpus_test_new)
print(model.predict(X_test))
print(model.predict_proba(X_test))
```

```
['neg' 'pos']
[[0.6  0.4 ]
 [0.43 0.57]]
```

In [33]:

```
corpus_new=list(map(cleaning,corpus))
cv=CountVectorizer(max_features=None,min_df=1,max_df=2)
X=cv.fit_transform(corpus_new).toarray()
print(cv.get_feature_names_out())
```

```
['cheap' 'costly' 'means' 'not' 'poor' 'quality' 'service' 'tasty']
```

In [30]:

```
print(corpus_new)
```

```
['food good good', 'food tasty', 'quality good', 'food not good', 'service poor poor mea
ns poor', 'costly', 'cheap quality']
```

In [37]:

```
corpus_new=list(map(cleaning,corpus))
cv=CountVectorizer(ngram_range=(1,1))
X=cv.fit_transform(corpus_new).toarray()
print(cv.get_feature_names_out())
```

```
['cheap' 'costly' 'food' 'good' 'means' 'not' 'poor' 'quality' 'service'
 'tasty']
```

In [39]:

```
corpus_new=list(map(cleaning,corpus))
cv=CountVectorizer(binary=True)
X=cv.fit_transform(corpus_new).toarray()
print(X)
```

```
[[0 0 1 1 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 1]
 [0 0 0 1 0 0 0 1 0 0]
 [0 0 1 1 0 1 0 0 0 0]
 [0 0 0 0 1 0 1 0 1 0]
 [0 1 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 1 0 0]]
```

In [40]:

```
corpus
```

Out[40]:

```
['f5o@od is # good & good!',
 '& Food # is * tasty',
 'Quality is Good',
 'food is not good',
 'servi89ce is Poor poor means very poor',
 'it is to_o costly',
 'che^ap quality']
```

In [41]:

```
cv=CountVectorizer(lowercase=True,stop_words=spwords)
cv.fit_transform(corpus)
print(cv.get_feature_names_out())
```

```
['ap' 'che' 'costly' 'f5o' 'food' 'good' 'means' 'not' 'od' 'poor'
 'quality' 'servi89ce' 'tasty' 'to_o']
```

In [42]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [44]:

```
tv=TfidfVectorizer()
X=tv.fit_transform(corpus_new).toarray()
print(tv.get_feature_names_out())
```

```
['cheap' 'costly' 'food' 'good' 'means' 'not' 'poor' 'quality' 'service'
 'tasty']
```

In [45]:

```
X
```

Out[45]:

```
array([[0.        , 0.        , 0.4472136 , 0.89442719, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.57866699, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.81556393],
       [0.        , 0.        , 0.        , 0.64974959, 0.        ,
        0.        , 0.        , 0.76014832, 0.        , 0.        ],
       [0.        , 0.        , 0.5008545 , 0.5008545 , 0.        ,
        0.70589627, 0.        , 0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.30151134,
        0.        , 0.90453403, 0.        , 0.30151134, 0.        ],
       [0.        , 1.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ],
       [0.76944876, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.63870855, 0.        , 0.        ]])
```

In [48]:

```
corpus_new=list(map(cleaning,corpus))
cv=CountVectorizer()
X=cv.fit_transform(corpus_new).toarray()
print(cv.get_feature_names_out())
print(X)
```

```
['cheap' 'costly' 'food' 'good' 'means' 'not' 'poor' 'quality' 'service'
 'tasty']
[[0 0 1 2 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 1]
 [0 0 0 1 0 0 0 1 0 0]
 [0 0 1 1 0 1 0 0 0 0]
 [0 0 0 0 1 0 3 0 1 0]
 [0 1 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 1 0 0]]
```

In [47]:

```
corpus_new
```

Out[47]:

```
['food good good',
 'food tasty',
 'quality good',
 'food not good',
 'service poor poor means poor',
 'costly',
 'cheap quality']
```

In [49]:

```
import math
```

In [50]:

```
math.log(8/4)+1
```

Out[50]:

```
1.6931471805599454
```

In [52]:

```
1.6931471805599454*2
```

Out[52]:

```
3.386294361119891
```

In [53]:

```
1.6931471805599454**2
```

Out[53]:

```
2.8667473750380923
```

In [54]:

```
3.386294361119891**2
```

Out[54]:

```
11.46698950015237
```

In [55]:

```
2.8667473750380923+11.46698950015237
```

Out[55]:

```
14.33373687519046
```

In [56]:

```
math.sqrt(14.33373687519046)
```

Out[56]:

```
3.7859921916441484
```

In [57]:

```
1.6931471805599454/3.7859921916441484
```

Out[57]:

```
0.4472135954999579
```

In [58]:

```
3.386294361119891/3.7859921916441484
```

```
0.8944271909999159
```

Out[58]:

In [59]:
```
tv=TfidfVectorizer(binary=True)
X=tv.fit_transform(corpus_new).toarray()
print(tv.get_feature_names_out())
```

```
['cheap' 'costly' 'food' 'good' 'means' 'not' 'poor' 'quality' 'service'
 'tasty']
```

In [60]:
```
X
```

Out[60]:
```
array([[0.        , 0.        , 0.70710678, 0.70710678, 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.57866699, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.81556393],
       [0.        , 0.        , 0.        , 0.64974959, 0.        ,
        0.        , 0.        , 0.76014832, 0.        , 0.        ],
       [0.        , 0.        , 0.5008545 , 0.5008545 , 0.        ,
        0.70589627, 0.        , 0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.57735027,
        0.        , 0.57735027, 0.        , 0.57735027, 0.        ],
       [0.        , 1.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ],
       [0.76944876, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.63870855, 0.        , 0.        ]])
```

In [61]:
```
df=pd.read_csv("f:/dataset/sentiment/Restaurant_Reviews.txt",sep="\t")
```

In [62]:
```
df
```

Out[62]:

	Review	Liked
0	Wow... Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of...	1
4	The selection on the menu was great and so wer...	1
...	...	...
995	I think food should have flavor and texture an...	0
996	Appetite instantly gone.	0
997	Overall I was not impressed and would not go b...	0
998	The whole experience was underwhelming, and I ...	0
999	Then, as if I hadn't wasted enough of my life ...	0

1000 rows × 2 columns

In [67]:
```
corpus=df.Review
target=df.Liked
```

In [68]:
```
corpus_new=list(map(cleaning,corpus))
cv=CountVectorizer()
X=cv.fit_transform(corpus_new).toarray()
model=RandomForestClassifier()
model.fit(X,target)
```

Out[68]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [69]:

```
sample1='Food quality is not good$'
sample2='awesome food'
corpus_test=[sample1,sample2]
corpus_test_new=list(map(cleaning,corpus_test))
X_test=cv.transform(corpus_test_new)
print(model.predict(X_test))
print(model.predict_proba(X_test))
```

```
[0 1]
[[0.77 0.23]
 [0.04 0.96]]
```

In [76]:

```
sample1=input("enter your Review:")
corpus_test=[sample1]
corpus_test_new=list(map(cleaning,corpus_test))
X_test=cv.transform(corpus_test_new)
if model.predict_proba(X_test)[0][0]<.5:
    print('you liked')
else:
    print('you did not like..')
```

```
you did not like..
```

In [77]:

```
import nltk
```

In [ ]:

```
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```

In [ ]:

```

```