** 19MAI0017

TASK1 - LOADING THE DATASET**

In [ ]:
```
import pandas as pd
df=pd.read_csv('IMDB Dataset.csv')
```

In [8]:
```
#after loading,lets introspect this dataset
df.head(10)
```

Out[8]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. <br /><br />The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
5	Probably my all-time favorite movie, a story o...	positive
6	I sure would like to see a resurrection of a u...	positive
7	This show was an amazing, fresh & innovative i...	negative
8	Encouraged by the positive comments about this...	negative
9	If you like original gut wrenching laughter yo...	positive

In [9]:
```
df['review'][0]
```

Out[9]:
```
"One of the other reviewers has mentioned that after watching just 1 Oz
episode you'll be hooked. They are right, as this is exactly what happe
```

```
ned with me.<br /><br />The first thing that struck me about Oz was its
brutality and unflinching scenes of violence, which set in right from t
he word GO. Trust me, this is not a show for the faint hearted or timi
d. This show pulls no punches with regards to drugs, sex or violence. I
ts is hardcore, in the classic use of the word.<br /><br />It is called
OZ as that is the nickname given to the Oswald Maximum Security State P
enitentary. It focuses mainly on Emerald City, an experimental section
of the prison where all the cells have glass fronts and face inwards, s
o privacy is not high on the agenda. Em City is home to many..Aryans, M
uslims, gangstas, Latinos, Christians, Italians, Irish and more....so s
cuffles, death stares, dodgy dealings and shady agreements are never fa
r away.<br /><br />I would say the main appeal of the show is due to th
e fact that it goes where other shows wouldn't dare. Forget pretty pict
ures painted for mainstream audiences, forget charm, forget romance...O
Z doesn't mess around. The first episode I ever saw struck me as so nas
ty it was surreal, I couldn't say I was ready for it, but as I watched
more, I developed a taste for Oz, and got accustomed to the high levels
of graphic violence. Not just violence, but injustice (crooked guards w
ho'll be sold out for a nickel, inmates who'll kill on order and get aw
ay with it, well mannered, middle class inmates being turned into priso
n bitches due to their lack of street skills or prison experience) Watc
hing Oz, you may become comfortable with what is uncomfortable viewin
g....thats if you can get in touch with your darker side."
```

Transforming documents into feature vectors

In [6]:
```python
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
count=CountVectorizer()
docs = np.array(['The sun is shining',
                 'The weather is sweet',
                 'The sun is shining, the weather is sweet, and one and
 one is two'])
bag=count.fit_transform(docs)
print(count.vocabulary_)
```

```
{'the': 6, 'sun': 4, 'is': 1, 'shining': 3, 'weather': 8, 'sweet': 5,
'and': 0, 'one': 2, 'two': 7}
```

In [7]:
```python
print(bag.toarray())
```

```
[[0 1 0 1 1 0 1 0 0]
 [0 1 0 0 0 1 1 0 1]
 [2 3 2 1 1 1 2 1 1]]
```

Term Frequency and Inverse Document Frequency

Task 3 Term Frequency and Inverse Document Frequency

Term frequencies alone do not contribute to distinct information

tf-idf (t,d) = tf(t,d) X idf(t,d)

idf(t,d) = log(nd / 1 + df(d,t))

where nd = total number of documents and df(d,t) = number of documents that contain the term t

In [32]:
```python
from sklearn.feature_extraction.text import TfidfTransformer
tfidf=TfidfTransformer(use_idf=True,norm='l2',smooth_idf=True)
#tr = TfidfTransformer(smooth_idf=True, norm='l2')
print(tfidf.fit_transform(count.fit_transform(docs)).toarray())
```

```
np.set_printoptions(precision=2)
print(tfidf.fit_transform(count.fit_transform(docs)).toarray())
```

```
[[0.         0.43370786 0.         0.55847784 0.55847784 0.
  0.43370786 0.         0.        ]
 [0.         0.43370786 0.         0.         0.         0.55847784
  0.43370786 0.         0.55847784]
 [0.50238645 0.44507629 0.50238645 0.19103892 0.19103892 0.19103892
  0.29671753 0.25119322 0.19103892]]
[[0.   0.43 0.   0.56 0.56 0.   0.43 0.   0.  ]
 [0.   0.43 0.   0.   0.   0.56 0.43 0.   0.56]
 [0.5  0.45 0.5  0.19 0.19 0.19 0.3  0.25 0.19]]
```

Tokenization of document

In [35]:
```
#task 5
from nltk.stem.porter import PorterStemmer
porter=PorterStemmer()

def stemmer_tokenize(text):
    return [porter.stem(word) for word in text.split()]
stemmer_tokenize('coders like coding and thus they code')
```

Out[35]:
```
['coder', 'like', 'code', 'and', 'thu', 'they', 'code']
```

vectorization of document

In [38]:
```
#task 6 vectorizig dataset(x)
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf=TfidfVectorizer(strip_accents=None,lowercase=False,tokenizer=stem
mer_tokenize,use_idf=True,norm='l2',smooth_idf=True)
y=df.sentiment.values
x=tfidf.fit_transform(df.review)
```

Document classification using LogisticRegression

In [40]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=1,test_
size=0.5,shuffle=False)

import pickle
from sklearn.linear_model import LogisticRegressionCV

#model
clf=LogisticRegressionCV(cv=5,scoring='accuracy',random_state=0,n_jobs=
3,verbose=3,max_iter=300).fit(x_train,y_train)
#saving model
saved_model=open('saved_model.sav','wb')

#using the pickle library's dump function to write the trained classifi
er
pickle.dump(clf,saved_model)
saved_model.close()
```

```
[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent worke
rs.
[Parallel(n_jobs=3)]: Done   2 out of   5 | elapsed: 10.1min remaining:
15.1min
[Parallel(n_jobs=3)]: Done   5 out of   5 | elapsed: 17.0min finished
```

Model evaluation

In [41]:
```
#model evaluation
filename='saved_model.sav'
saved_clf=pickle.load(open(filename,'rb'))

#test the saved model on the test data
saved_clf.score(x_test,y_test)
#saved_clf.close()
```

Out[41]:
```
0.8898
```

```
In [ ]:
```