The purpose of this project is to demonstrate natural language processing using bag of words methodology. Thanks to the Kaggle website for providing this great database for comment sentiment

Importing libraries

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

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

In [2]:
```python
# Reading the dataset
dataset=pd.read_csv('IMDB Dataset.csv', engine=None)
dataset
```

	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
...	...	...
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

```
50000 rows × 2 columns
```

In [3]:
```python
#reviweing the data
dataset.info()
dataset.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

	review	sentiment
count	50000	50000
unique	49582	2
top	Loved today's show!!! It was a variety and not...	negative
freq	5	25000

Cleaning the texts

The objectives in this section are as following: 1-clean the dataset from any non-alphabetic characters and make everything lowercase. 2- remove non-english words 3- stem the words for example 'liked' and 'like' become 'like' 4- tokenize the dataset

In [4]:
```python
# I am going to use the stopwords function to remove unnecessary words. However, words that make a sente
nce positive or negative are important and should not be removed.
# so printed the stopwords and picked the words that I needed. ode

from nltk.corpus import stopwords
_stopwords=stopwords.words('english')

i_need_them=['not','ain','aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't",
'hadn',
 "hadn't",'hasn',"hasn't",'haven',"haven't",'isn',"isn't",'ma','mightn',"mightn't",'mustn',"mustn't",'ne
edn',
 "needn't",'shan',"shan't",'shouldn',"shouldn't",'wasn',"wasn't",'weren',"weren't",'won',"won't",'would
n',"wouldn't"]

for x in i_need_them:
   _stopwords.remove(x)

# Also here I download and create a set of all english words which will be helpful in the next cell.
nltk.download('words')
english_words = set(nltk.corpus.words.words())
```

```
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\arzar\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\words.zip.
```

In [5]:
```python
# initilizing a list
main_list=[]

# removing non-alphabetic characters using re library, making lower case and split by space
for i in range(0, dataset.shape[0]):
  comment = re.sub('[^a-zA-Z]', ' ', dataset['review'][i]) #here I am exluding the a to z and A to Z cha
recters
  comment=comment.lower()    # lower case
  comment=comment.split()    # split by spacce
  ps=PorterStemmer()         #Getting the class for stem

  # here I perform the stemming while checking if the word is stopper and ensuring is an English word
  comment= [ps.stem(i) for i in comment if i not in _stopwords and i in english_words]

  # appending them
  comment = ' '.join(comment)
  # finilizing the main_list
  main_list.append(comment) # append to the main_list
```

Creating Bag of Words

In [14]:
```python
from sklearn.feature_extraction.text import CountVectorizer
countvector = CountVectorizer()
# getting my variables
X = countvector.fit_transform(main_list).toarray()
y = dataset.iloc[:, -1].values
```

In [15]:
```python
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit_transform(y)
```

```
array([1, 1, 1, ..., 0, 0, 0])
```

In [16]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

In [20]:
```python
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
D:\Anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed to 'lbf
gs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=0, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [21]:
```python
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[['positive' 'positive']
 ['negative' 'negative']
 ['positive' 'negative']
 ...
 ['positive' 'positive']
 ['positive' 'positive']
 ['negative' 'negative']]
```

In [26]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[4347  688]
 [ 631 4334]]
```

```
0.8681
```

In [ ]: