# Sentiment Analysis Using Logistic Regression

In [1]:

```
# Import Library and Load Dataset

import pandas as pd
df = pd.read_csv('IMDB Dataset.csv')
```

## Dataset Preview

In [2]:

```
#After Loading, Introspect this dataset
df.head(10)
df['review'][0]
```

Out[2]:

```
"One of the other reviewers has mentioned that after watching just 1 Oz epis
ode you'll be hooked. They are right, as this is exactly what happened 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 the word GO. Trust m
e, this is not a show for the faint hearted or timid. This show pulls no pun
ches with regards to drugs, sex or violence. Its 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 Penitentary. It focuses mainly on Emerald
City, an experimental section of the prison where all the cells have glass f
ronts and face inwards, so privacy is not high on the agenda. Em City is hom
e to many..Aryans, Muslims, gangstas, Latinos, Christians, Italians, Irish a
nd more....so scuffles, death stares, dodgy dealings and shady agreements ar
e never far away.<br /><br />I would say the main appeal of the show is due
to the fact that it goes where other shows wouldn't dare. Forget pretty pict
ures painted for mainstream audiences, forget charm, forget romance...OZ doe
sn't mess around. The first episode I ever saw struck me as so nasty it was
surreal, I couldn't say I was ready for it, but as I watched more, I develop
ed a taste for Oz, and got accustomed to the high levels of graphic violenc
e. Not just violence, but injustice (crooked guards who'll be sold out for a
nickel, inmates who'll kill on order and get away with it, well mannered, mi
ddle class inmates being turned into prison bitches due to their lack of str
eet skills or prison experience) Watching Oz, you may become comfortable wit
h what is uncomfortable viewing....thats if you can get in touch with your d
arker side."
```

## Transforming Document and Vectorize

In [3]:

```
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_)
print(bag.toarray())
```

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

## Feature Extraction

In [4]:

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf = TfidfTransformer(use_idf=True, norm='l2', smooth_idf=True)

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

In [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[5]:

```
['coder', 'like', 'code', 'and', 'thu', 'they', 'code']
```

In [6]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(strip_accents = None,
                       lowercase = False,
                        tokenizer = stemmer_tokenize,
                        use_idf = True,
                        norm='l2',
                        smooth_idf=True)

Y = df.sentiment.values
X = tfidf.fit_transform(df.review)
```

## Sentiment Classication using Logistic Regression

In [7]:

```
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,shu

import pickle
from sklearn.linear_model import LogisticRegressionCV

#Model
clf = LogisticRegressionCV(cv = 5,
                            scoring = 'accuracy',
                            random_state = 3,
                            n_jobs = -3,
                            verbose = -3,
                            max_iter = 300).fit(X_train, Y_train)

saved_model = open('saved_model.sav', 'wb')

pickle.dump(clf, saved_model)
saved_model.close()
```

```
[Parallel(n_jobs=-3)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-3)]: Done   3 out of   5 | elapsed:  5.0min remaining:  3.
3min
[Parallel(n_jobs=-3)]: Done   5 out of   5 | elapsed:  7.4min remaining:
0.0s
[Parallel(n_jobs=-3)]: Done   5 out of   5 | elapsed:  7.4min finished
```

# Model Evaluation

In [8]:

```
filename = 'saved_model.sav'
saved_clf = pickle.load(open(filename, 'rb'))

saved_clf.score(X_test, Y_test)
```

```
c:\users\harshal patel\appdata\local\programs\python\python37\lib\site-packa
ges\sklearn\linear_model\logistic.py:2260: ChangedBehaviorWarning: The long-
standing behavior to use the accuracy score has changed. The scoring paramet
er is now used. This warning will disappear in version 0.22.
  ChangedBehaviorWarning)
```

Out[8]:

```
0.8898
```