About Dataset

IMDB dataset having 50K movie reviews for natural language processing or Text analytics.This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training and 25,000 for testing.

Download ¶

https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews (https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews)

In [1]:

```

#import
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import re
import string
import nltk
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
#load dataset
data=pd.read_csv("IMDB Dataset.csv")
data.head()
```

Out[2]:

	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

In [3]:

```
data.tail()
```

Out[3]:

	review	sentiment
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

In [4]:

```
data.shape
```

Out[4]:

```
(50000, 2)
```

In [5]:

```
data.info()
```

```
<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
```

In [6]:

```
#Preprocessing dataset
def remove_pattern(input_txt,pattern):
    r=re.findall(pattern,input_txt)
    for word in r:
        input_txt=re.sub(word,"",input_txt)
    return input_txt
```

In [7]:

```
from sklearn.preprocessing import LabelEncoder
```

In [8]:

```
lb = LabelEncoder()
lb
```

Out[8]:

```
LabelEncoder()
```

In [9]:

```
data.columns
```

Out[9]:

```
Index(['review', 'sentiment'], dtype='object')
```

In [10]:

```
data['sentiment']= lb.fit_transform(data['sentiment'])
```

In [11]:

```
data.head()
```

Out[11]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. <br /><br />The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

In [12]:

```
from sklearn.feature_extraction.text import CountVectorizer
```

In [13]:

```
bagwords_vectorizer=CountVectorizer(max_df=0.90,min_df=2,max_features=1000,stop_words="engl
```

In [14]:

```
bagwords=bagwords_vectorizer.fit_transform(data["review"])
```

In [15]:

```
bagwords[0].toarray()
```

Out[15]:

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
        0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
        0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,
        0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0]], dtype=int64)
```

In [16]:

```
from sklearn.model_selection import train_test_split
```

In [17]:

```
x_train,x_test,y_train,y_test=train_test_split(bagwords,data["sentiment"],test_size=0.3,ran
```

In [18]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
```

In [19]:

```
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

Out[19]:

```
LogisticRegression()
```

In [20]:

```
y_pred=lr.predict(x_test)
y_pred
```

Out[20]:

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

In [21]:

```
f1_score(y_test,y_pred)
```

Out[21]:

```
0.8603698811096433
```

In [22]:

```
accuracy_score(y_test,y_pred)
```

Out[22]:

```
0.8590666666666666
```

```
#Simple linear
```

In [23]:

```
from sklearn.linear_model import LinearRegression
```

In [24]:

```
lr=LinearRegression()
```

In [25]:

```
from sklearn.model_selection import train_test_split
```

In [26]:

```
x_train,x_test,y_train,y_test=train_test_split(bagwords,data["sentiment"],test_size=0.3,ran
```

In [27]:

```
lr.fit(x_train,y_train)
```

Out[27]:

```
LinearRegression()
```

In [28]:

```
y_pred=lr.predict(x_test)
y_pred
```

Out[28]:

```
array([ 0.63232451, -0.22348682,  0.75338716, ...,  0.69911988,
        0.5610131 ,  0.29367937])
```

In [29]:

```
lr.score(x_train,y_train)
```

Out[29]:

```
0.48157347300769826
```

In [30]:

```
print("Coefficient: \n",lr.coef_)
```

```
Coefficient: 
 [ 1.81758304e-02 -1.41418792e-03 -2.05878562e-02  1.19190166e-03
 -3.51725294e-02 -3.63273408e-04  2.80209993e-02 -1.59018151e-02
 -8.82937062e-04  1.61578618e-02 -1.91816622e-02 -4.42629256e-02
 -3.48991579e-03  1.74845213e-02 -1.94874687e-02  1.64473367e-02
 -2.47750194e-02 -1.33916625e-02 -1.63997095e-02  3.84226199e-03
  5.12741204e-04 -2.67016179e-02 -1.33724062e-02  1.68033172e-02
  9.32564968e-03  2.05708823e-02  1.84989777e-02  2.14619800e-03
  3.76864847e-03 -1.08140441e-02  1.09378252e-02  2.65506629e-02
  6.28251924e-02 -4.69743299e-03  8.60192085e-04 -3.28967258e-02
  8.16689214e-03 -6.72525217e-02 -1.10059053e-03 -6.46667858e-03
 -2.59626085e-02 -6.89068965e-04 -2.54389626e-02  3.42310743e-02
 -1.58334606e-02 -8.89442927e-03 -3.97663848e-03 -8.87697425e-03
 -5.48832491e-03  4.43949381e-02 -6.07125719e-02 -2.72285101e-02
  3.20582294e-02 -6.58798394e-03 -1.03604506e-02 -2.27318282e-02
 -8.60555042e-02 -6.58775957e-03  6.92660047e-02 -6.63611833e-02
  2.20856182e-03  8.40686440e-03 -4.66330756e-02 -7.31418654e-02
 -1.10828354e-02 -5.32875440e-02  1.23849862e-03 -3.99586997e-04
 -3.71211744e-02  2.06688785e-02  2.80565803e-02  1.74808922e-02
```

In [31]:

```
print("Intercept :\n",lr.intercept_)
```

```
Intercept :
 0.5007616378862353
```

In [32]:

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [33]:

```
mean_absolute_error(y_test, y_pred)
```

Out[33]:

```
0.31971691504703836
```

In [34]:

```
mean_squared_error(y_test, y_pred)
```

Out[34]:

```
0.1392473435276331
```

In [35]:

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

Out[35]:

```
0.3731586037164802
```

In [36]:

```
r2_score(y_test, y_pred)
```

Out[36]:

```
0.44299478218549404
```

In [37]:

```
a={"Actual value":x_train,"predicted value":y_pred}
a
```

Out[37]:

```
{'Actual value': <35000x1000 sparse matrix of type '<class 'numpy.int64'>'
        with 1621554 stored elements in Compressed Sparse Row format>,
 'predicted value': array([ 0.63232451, -0.22348682,  0.75338716, ...,  0.69
911988,
         0.5610131 ,  0.29367937])}
```

In [ ]: