Open in Colab

In [1]:

```
!curl https://raw.githubusercontent.com/HeptaDecane/LP2_SEM7/main/A04/ImdbDataset.csv --
output ImdbDataset.csv
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 63.1M  100 63.1M    0     0   137M      0 --:--:-- --:--:-- --:--:--  137M
```

In [2]:

```
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt

from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
```

In [3]:

```
df = pd.read_csv('ImdbDataset.csv')
df
```

Out[3]:

| | 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 [4]:

```
df.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 [5]:

```
df['sentiment'] = df['sentiment'].map({'positive':1, 'negative':0})
df.head()
```

Out[5]:

| | 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 [6]:

```
import re
import nltk
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[6]:

```
True
```

In [7]:

```
from nltk.corpus import stopwords

stop_words = stopwords.words('english')
stop_words.remove('not')
print(stop_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
"you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himsel
f', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'the
m', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "tha
t'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if
', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'ag
ainst', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to'
, 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 't
hen', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each
', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'only', 'own', 'same', 's
o', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "sh
ould've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn',
"couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'ha
ven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn',
"needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't"
, 'won', "won't", 'wouldn', "wouldn't"]
```

In [8]:

```
from nltk.stem.porter import PorterStemmer

stemmer = PorterStemmer()

def clean_text(text):
    text = text.lower()
    text = text.replace('<br />','')
    text = text.replace('<br/>', '')
    text = re.sub('[^a-z]',' ',text)
    text = text.split()
    words = [stemmer.stem(word) for word in text if not word in set(stop_words)]
    text = ' '.join(words)
    return text
```

In [9]:

```
df['review'] = df['review'].apply(lambda review: clean_text(review))
df['review'].head()
```

Out[9]:

```
0    one review mention watch oz episod hook right ...
1    wonder littl product film techniqu unassum old...
2    thought wonder way spend time hot summer weeke...
3    basic famili littl boy jake think zombi closet...
4    petter mattei love time money visual stun film...
Name: review, dtype: object
```

In [10]:

```
x = df['review']
y = df['sentiment']
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25,random_state=73)
```

In [11]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(max_features=10000)

vectorizer.fit(df['review'])
x_train = vectorizer.transform(x_train)
x_test = vectorizer.transform(x_test)
```

In [12]:

```
# vectorizer.vocabulary_
```

In [13]:

```
print(x_train)
```

```
  (0, 9959) 0.09290163810807579
  (0, 9735) 0.06279746064472047
  (0, 9729) 0.13586214000968874
  (0, 9684) 0.057332319381930265
  (0, 9659) 0.17649828984372273
  (0, 9617) 0.2268857508002915
  (0, 9305) 0.16381075700560466
  (0, 9025) 0.10868926551224542
  (0, 8961) 0.25403355317527276
  (0, 8577) 0.10872685942617832
  (0, 8279) 0.10319423028751934
  (0, 8076) 0.11642874290269481
  (0, 7848) 0.12326446133371552
  (0, 7793) 0.11321576448710348
  (0, 7697) 0.129200749148155
  (0, 7501) 0.08814746944190625
  (0, 7441) 0.2950046418304996
  (0, 7435) 0.09006602284731824
  (0, 7153) 0.08333424783973568
  (0, 7146) 0.17710721415333278
  (0, 7123) 0.10306275804745381
  (0, 6813) 0.19985680576542647
  (0, 6794) 0.13509574474476888
  (0, 6605) 0.30441401882448244
  (0, 6545) 0.08151147488589215
  : :
  (37499, 1595) 0.0492005871556192
  (37499, 1524) 0.05853609645328196
  (37499, 1499) 0.047294513642451935
  (37499, 1469) 0.048147246504269145
  (37499, 1439) 0.07023895683582054
  (37499, 1170) 0.10227520706592036
  (37499, 1099) 0.09080359856572698
  (37499, 1001) 0.08273480394869831
  (37499, 929) 0.04709402163659443
  (37499, 855) 0.07132531147800265
  (37499, 828) 0.05513674102137520
```

```
  (37499, 828) 0.05513674105137520 4
  (37499, 760) 0.06742276282242218
  (37499, 615) 0.07510310414435552
  (37499, 602) 0.06333860631213609
  (37499, 568) 0.041977000217115874
  (37499, 518) 0.18993662804250824
  (37499, 509) 0.0889785913502585
  (37499, 438) 0.034731507780166634
  (37499, 366) 0.036575139873938775
  (37499, 329) 0.04452575374375134
  (37499, 306) 0.06511395729610119
  (37499, 195) 0.06601235199691813
  (37499, 164) 0.05803835077129399
  (37499, 109) 0.10240632586789034
  (37499, 40) 0.06208616437141953
```

In [14]:

```
print(x_test)
```

```
  (0, 9892) 0.20383555058926328
  (0, 9661) 0.08301880666613018
  (0, 9172) 0.1921095775138967
  (0, 9039) 0.11343868871294127
  (0, 8950) 0.09757714550398244
  (0, 8932) 0.0755915723732292
  (0, 8492) 0.17498302339706603
  (0, 8473) 0.17787379754070887
  (0, 8323) 0.14249422510198512
  (0, 8244) 0.08958965389088576
  (0, 8221) 0.2537525569880813
  (0, 7013) 0.10269356786742348
  (0, 6898) 0.11956086784193688
  (0, 6856) 0.18024438505032694
  (0, 6609) 0.16357954702111532
  (0, 6442) 0.09079373592478315
  (0, 5801) 0.11432888025188787
  (0, 5686) 0.1587883334785956
  (0, 5538) 0.22556187385485577
  (0, 5404) 0.06202424061711628
  (0, 5403) 0.13364835758233448
  (0, 5225) 0.08514122050826281
  (0, 5139) 0.10643831801061217
  (0, 4559) 0.1942020831844242
  (0, 4394) 0.14416822290095765
  : :
  (12499, 1001) 0.03801276765647634
  (12499, 998) 0.06888355656169609
  (12499, 990) 0.05163585692101016
  (12499, 916) 0.078169066073777
  (12499, 913) 0.060265591483204276
  (12499, 912) 0.10315368436545098
  (12499, 760) 0.030977601874969684
  (12499, 745) 0.03409170230439834
  (12499, 619) 0.060074353385686945
  (12499, 613) 0.0997517391328271
  (12499, 602) 0.029101123233714218
  (12499, 514) 0.047546103895694955
  (12499, 474) 0.0557494218587292 7
  (12499, 446) 0.073821462898850 2
  (12499, 438) 0.06383000491184926
  (12499, 383) 0.10652027200841874
  (12499, 342) 0.0300532348293937
  (12499, 240) 0.04829557262219624
  (12499, 238) 0.041081931208132305
  (12499, 232) 0.037696897853684165
  (12499, 210) 0.100487606647925 79
  (12499, 173) 0.04621735102780964
  (12499, 80) 0.029099822829735424
  (12499, 74) 0.02350573019306764
  (12499, 14) 0.04957430966513072
```

In [15]:

```
model = MultinomialNB()
model.fit(x_train,y_train)
```

Out[15]:

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

In [16]:

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

Out[16]:

```
0.87256
```

In [17]:

```
test_set_prediction = model.predict(x_test)
```

In [18]:

```
matrix = metrics.confusion_matrix(y_test,test_set_prediction)
matrix_df = pd.DataFrame(data=matrix,index=['-ve','+ve'],columns=['predicted -ve','predicted +ve'])
matrix_df
```

Out[18]:

| | predicted -ve | predicted +ve |
|---|---|---|
| **-ve** | **5292** | **878** |
| **+ve** | **911** | **5419** |

In [19]:

```
print(metrics.classification_report(y_test,test_set_prediction))
```

```
              precision    recall  f1-score   support

           0       0.85      0.86      0.86      6170
           1       0.86      0.86      0.86      6330

    accuracy                           0.86     12500
   macro avg       0.86      0.86      0.86     12500
weighted avg       0.86      0.86      0.86     12500
```