entiment-analysis-and-knn-of-imdb

January 30, 2024

[1]:
```
!pip install beautifulsoup4
!pip install wordcloud
```

```
Requirement already satisfied: beautifulsoup4 in c:\users\hp\anaconda3\lib\site-
packages (4.12.2)
Requirement already satisfied: soupsieve>1.2 in c:\users\hp\anaconda3\lib\site-
packages (from beautifulsoup4) (2.4)
Requirement already satisfied: wordcloud in c:\users\hp\anaconda3\lib\site-
packages (1.9.2)
Requirement already satisfied: numpy>=1.6.1 in c:\users\hp\anaconda3\lib\site-
packages (from wordcloud) (1.24.3)
Requirement already satisfied: pillow in c:\users\hp\anaconda3\lib\site-packages
(from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in c:\users\hp\anaconda3\lib\site-
packages (from wordcloud) (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\hp\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\hp\anaconda3\lib\site-
packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\hp\anaconda3\lib\site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\hp\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in
c:\users\hp\anaconda3\lib\site-packages (from matplotlib->wordcloud) (23.1)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in
c:\users\hp\anaconda3\lib\site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\hp\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\hp\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
```

[2]:
```
import pandas as pd
```

[3]:
```
df=pd.read_csv('IMDB Dataset.csv')
df.head()
```

[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
```

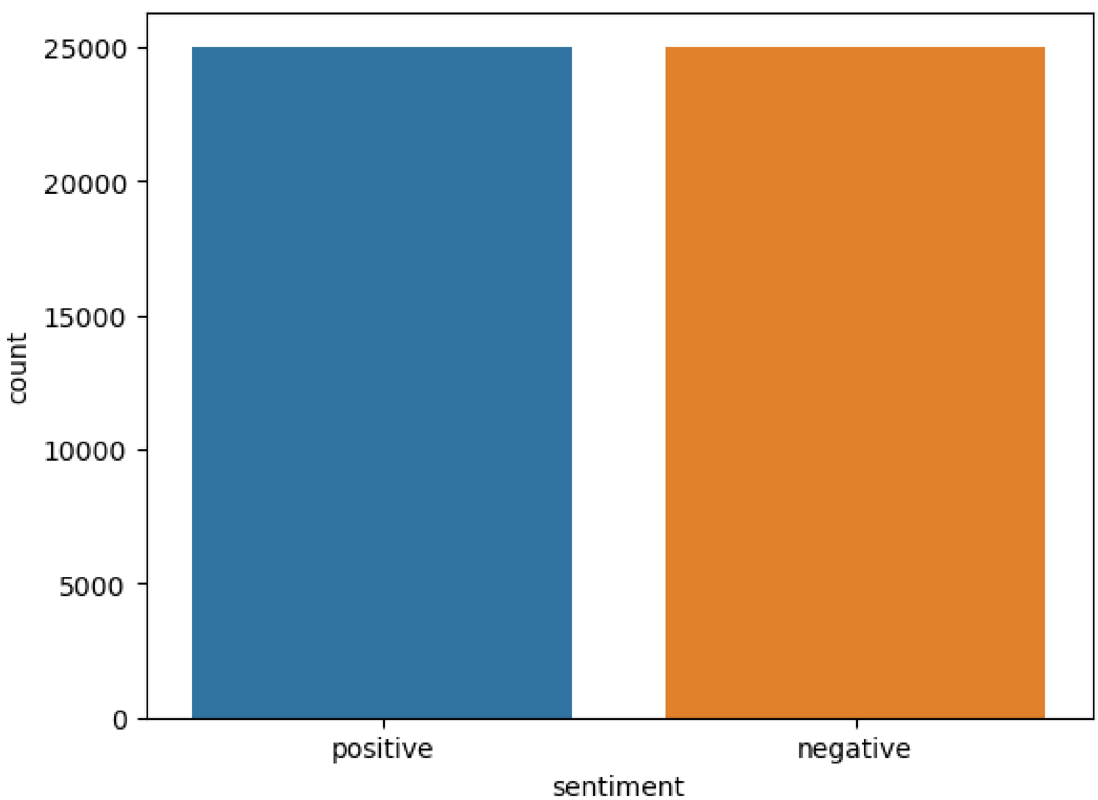
[4]:
```python
df['sentiment'].value_counts()
```

[4]:
```
sentiment
positive    25000
negative    25000
Name: count, dtype: int64
```

[5]:
```python
import seaborn as sns
```

[6]:
```python
sns.countplot(x='sentiment',data=df)
```

[6]:
```
<Axes: xlabel='sentiment', ylabel='count'>
```



[7]:
```
positive_review=list(df[df['sentiment']=='positive']['review'])[:100]
negative_review=list(df[df['sentiment']=='negative']['review'])[:100]
```

[8]:
```
from wordcloud import WordCloud,STOPWORDS
from matplotlib import pyplot as plt
stopwords=set(STOPWORDS)
stopwords
```

[8]:
```
{'a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'all',
 'also',
 'am',
 'an',
 'and',
 'any',
 'are',
 "aren't",
 'as',
 'at',
 'be',
 'because',
 'been',
 'before',
 'being',
 'below',
 'between',
 'both',
 'but',
 'by',
 'can',
 "can't",
 'cannot',
 'com',
 'could',
 "couldn't",
 'did',
 "didn't",
 'do',
 'does',
 "doesn't",
 'doing',
 "don't",
```

```
'down',
'during',
'each',
'else',
'ever',
'few',
'for',
'from',
'further',
'get',
'had',
"hadn't",
'has',
"hasn't",
'have',
"haven't",
'having',
'he',
"he'd",
"he'll",
"he's",
'hence',
'her',
'here',
"here's",
'hers',
'herself',
'him',
'himself',
'his',
'how',
"how's",
'however',
'http',
'i',
"i'd",
"i'll",
"i'm",
"i've",
'if',
'in',
'into',
'is',
"isn't",
'it',
"it's",
'its',
```

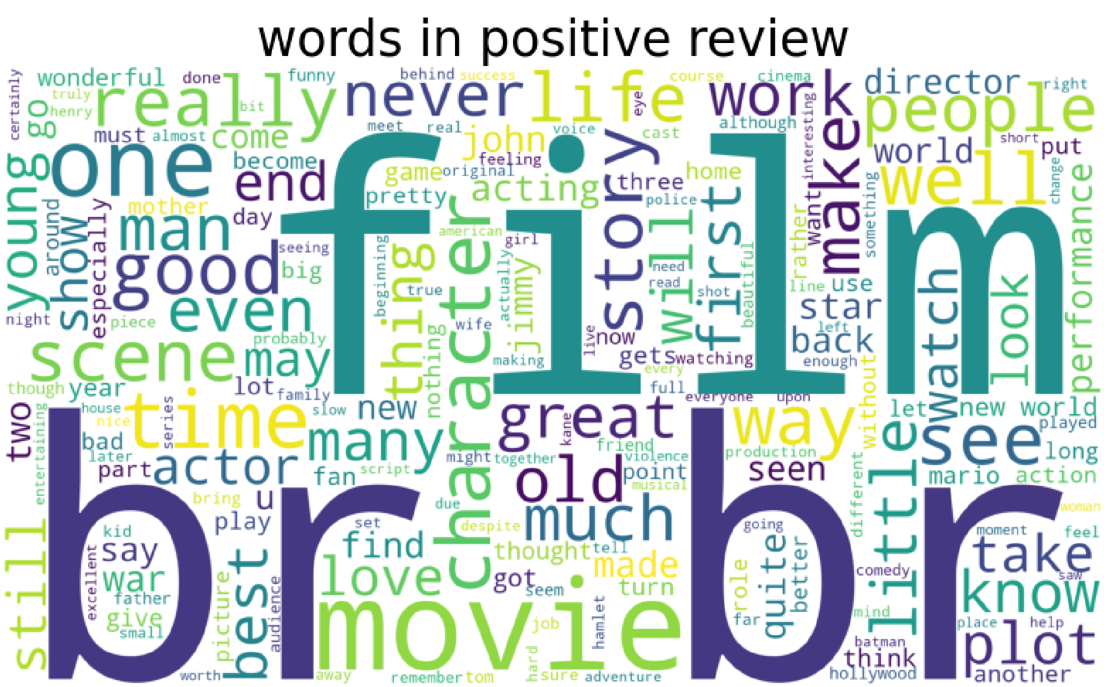
```
'itself',
'just',
'k',
"let's",
'like',
'me',
'more',
'most',
"mustn't",
'my',
'myself',
'no',
'nor',
'not',
'of',
'off',
'on',
'once',
'only',
'or',
'other',
'otherwise',
'ought',
'our',
'ours',
'ourselves',
'out',
'over',
'own',
'r',
'same',
'shall',
"shan't",
'she',
"she'd",
"she'll",
"she's",
'should',
"shouldn't",
'since',
'so',
'some',
'such',
'than',
'that',
"that's",
'the',
```

```
'their',
'theirs',
'them',
'themselves',
'then',
'there',
"there's",
'therefore',
'these',
'they',
"they'd",
"they'll",
"they're",
"they've",
'this',
'those',
'through',
'to',
'too',
'under',
'until',
'up',
'very',
'was',
"wasn't",
'we',
"we'd",
"we'll",
"we're",
"we've",
'were',
"weren't",
'what',
"what's",
'when',
"when's",
'where',
"where's",
'which',
'while',
'who',
"who's",
'whom',
'why',
"why's",
'with',
"won't",
```

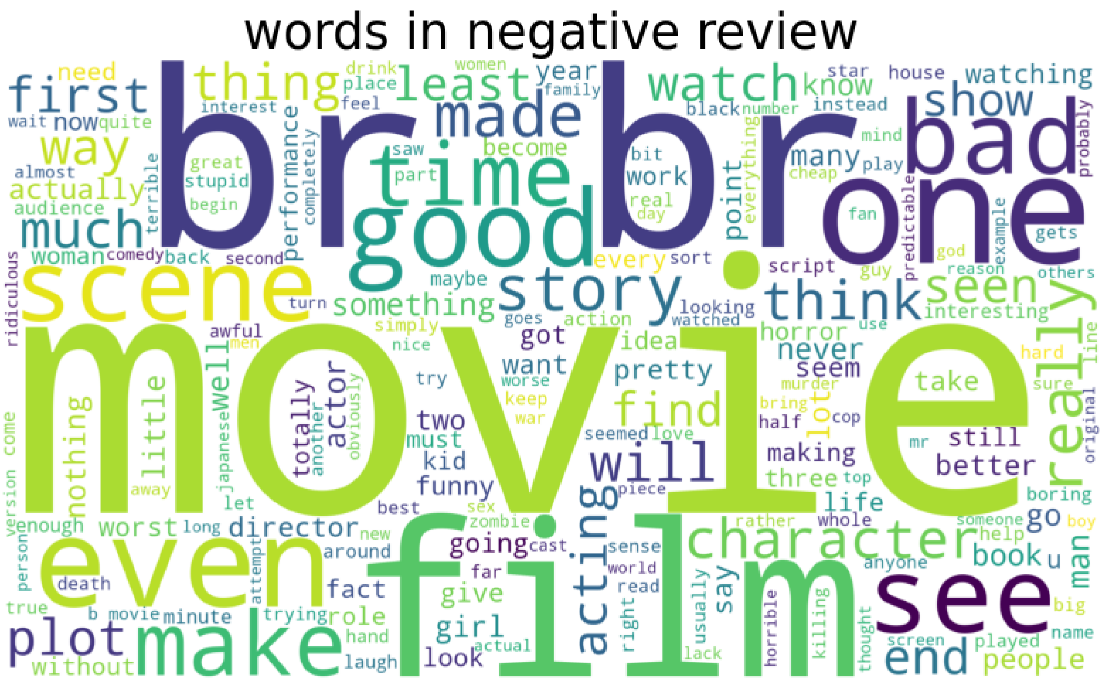
```
 'would',
 "wouldn't",
 'www',
 'you',
 "you'd",
 "you'll",
 "you're",
 "you've",
 'your',
 'yours',
 'yourself',
 'yourselves'}
```

[9]:
```python
def create_cloud(string,title=None):
    cloud=WordCloud(height=1080,
                    width=1920,
                    background_color='white',
                    min_font_size=10,
                    stopwords=STOPWORDS).generate(string)
    plt.figure(figsize=(10,20))
    plt.imshow(cloud)
    plt.axis("off")
    if title:
        plt.title(title,fontdict={'fontsize':25})
    plt.show()
```

[10]:
```python
create_cloud(' '.join(positive_review).lower(),'words in positive review')
```



```
[11]: create_cloud(' '.join(negative_review).lower(),'words in negative review')
```




```
[12]: def text_processing(data):
          from bs4 import BeautifulSoup
          import re
          def decontracted(phrase):
              # specific
              phrase= re.sub(r'<br /><br />',' ',phrase)
              phrase = re.sub(r"won't", "will not", phrase)
              phrase = re.sub(r"can\'t", "can not", phrase)

              # general
              phrase = re.sub(r"n\'t", " not", phrase)
              phrase = re.sub(r"\'re", " are", phrase)
              phrase = re.sub(r"\'s", " is", phrase)
              phrase = re.sub(r"\'d", " would", phrase)
              phrase = re.sub(r"\'ll", " will", phrase)
              phrase = re.sub(r"\'t", " not", phrase)
              phrase = re.sub(r"\'ve", " have", phrase)
              phrase = re.sub(r"\'m", " am", phrase)
              phrase = re.sub(r'"', " ", phrase)
              return phrase
```

```
    stopwords=set(STOPWORDS)

    # Combining all the above sentence
    from tqdm import tqdm
    preprocessed_reviews = []
    # tqdm is for printing the status bar
    for sentance in tqdm(data['review'].values):
        sentance = re.sub(r"http\S+", "", sentance)
        sentance = BeautifulSoup(sentance, 'lxml').get_text()
        sentance = decontracted(sentance)
        sentance = re.sub("\S*\d\S*", "", sentance).strip()
        # https://gist.github.com/sebleier/5542802
        sentance = ' '.join(e.lower() for e in sentance.split() if e not in␣
↪stopwords)
        preprocessed_reviews.append(sentance.strip())

    from nltk.stem import PorterStemmer

    porter = PorterStemmer()
    list_of_sentence=[]
    for  sentence in preprocessed_reviews:
        words_in_sentence=[]
        for words in sentence.split():
            words_in_sentence.append(porter.stem(words))

        list_of_sentence.append(' '.join(words_in_sentence))
    return(list_of_sentence)
```

[13]:
```
X=text_processing(df[:1000])
```

```
 76%|
| 755/1000 [00:00<00:00,
1910.58it/s]C:\Users\HP\AppData\Local\Temp\ipykernel_15932\2228875265.py:29:
MarkupResemblesLocatorWarning: The input looks more like a filename than markup.
You may want to open this file and pass the filehandle into Beautiful Soup.
  sentance = BeautifulSoup(sentance, 'lxml').get_text()
100%|
| 1000/1000 [00:00<00:00, 1919.34it/s]
```

[14]:
```
df=df[:1000]
```

[15]:
```
df.head()
```

[15]:
```
                                              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
```

[16]:

```python
df['cleaned_review']=X
```

[17]:

```python
df.head()
```

[17]:

```
                                              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

                                    cleaned_review
0  one review mention watch oz episod will hooked…
1  a wonder littl production. the film techniqu u…
2  i thought wonder way spend time hot summer wee…
3  basic famili littl boy (jake) think zombi clos…
4  petter mattei love time money visual stun film…
```

[18]:

```python
x=df['cleaned_review']
y=df['sentiment']
```

[19]:

```python
y = list(y)
for i in range(len(y)):
    if y[i]=='positive':
        y[i]=1
    else:
        y[i]=0

df['sentiment_score']=y

y=df['sentiment_score']
```

[20]:

```python
df
```

[20]:

```
                                                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
..                                                 …       …
995  Nothing is sacred. Just ask Ernie Fosselius. T…  positive
996  I hated it. I hate self-aware pretentious inan…  negative
997  I usually try to be professional and construct…  negative
```

```
998  If you like me is going to see this in a film …  negative
999  This is like a zoology textbook, given that it…  negative

                                        cleaned_review  sentiment_score
0    one review mention watch oz episod will hooked…                1
1    a wonder littl production. the film techniqu u…                1
2    i thought wonder way spend time hot summer wee…                1
3    basic famili littl boy (jake) think zombi clos…                0
4    petter mattei love time money visual stun film…                1
..                                                 …              …
995  noth sacred. just ask erni fosselius. these da…                1
996  i hate it. i hate self-awar pretenti inan masq…                0
997  i usual tri profession construct i critic movi…                0
998  if go see film histori class someth school, tr…                0
999  thi zoolog textbook, given depict anim accurat…                0

[1000 rows x 4 columns]
```

[21]:
```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x[:1000],y[:1000],test_size=0.
↪3,random_state=42)
```

[22]:
```python
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

[22]: ((700,), (300,), (700,), (300,))

[23]:
```python
list(y_test).count(0)
```

[23]: 161

[24]:
```python
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
x_train_bow = vectorizer.fit_transform(x_train)
x_test_bow = vectorizer.transform(x_test)
```

[25]:
```python
x_train_bow.shape,x_test_bow.shape
```

[25]: ((700, 13277), (300, 13277))

[26]:
```python
x_train.shape
```

[26]: (700,)

[27]:
```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,f1_score
for i in range(10,30):
```

```python
    print('K',i)

    # initialization
    neigh = KNeighborsClassifier(n_neighbors=i)

    # Training
    neigh.fit(x_train_bow, y_train)

    # Test the training data
    y_pred_train = neigh.predict(x_train_bow)
    accuracy_train = accuracy_score(y_pred_train,y_train)
    f1_train = f1_score(y_pred_train,y_train)

    # Test the test data
    y_pred_test = neigh.predict(x_test_bow)
    accuracy_test = accuracy_score(y_pred_test,y_test)
    f1_test = f1_score(y_pred_test,y_test)

    print('train accuracy : ',accuracy_train,' test accuracy: ',accuracy_test)
    print('f1 train: ',f1_train,' f1 test: ',f1_test)
    print()
```

```
K 10
train accuracy :  0.6785714285714286  test accuracy:  0.5166666666666667
f1 train:  0.734982332155477  f1 test:  0.5938375350140056

K 11
train accuracy :  0.6071428571428571  test accuracy:  0.49666666666666665
f1 train:  0.7077577045696068  f1 test:  0.6157760814249365

K 12
train accuracy :  0.64  test accuracy:  0.54
f1 train:  0.7206208425720619  f1 test:  0.6310160427807485

K 13
train accuracy :  0.6014285714285714  test accuracy:  0.5133333333333333
f1 train:  0.7096774193548386  f1 test:  0.6313131313131313

K 14
train accuracy :  0.6228571428571429  test accuracy:  0.5466666666666666
f1 train:  0.7173447537473233  f1 test:  0.6439790575916231

K 15
train accuracy :  0.5814285714285714  test accuracy:  0.5166666666666667
f1 train:  0.7007150153217568  f1 test:  0.6384039900249378
```

```
K 16
train accuracy :  0.5928571428571429  test accuracy:  0.53
f1 train:  0.7009443861490031  f1 test:  0.6412213740458015

K 17
train accuracy :  0.5771428571428572  test accuracy:  0.52
f1 train:  0.6991869918699186  f1 test:  0.6417910447761195

K 18
train accuracy :  0.5942857142857143  test accuracy:  0.5333333333333333
f1 train:  0.7053941908713693  f1 test:  0.6482412060301507

K 19
train accuracy :  0.5671428571428572  test accuracy:  0.5233333333333333
f1 train:  0.6948640483383686  f1 test:  0.6486486486486487

K 20
train accuracy :  0.58  test accuracy:  0.5466666666666666
f1 train:  0.6975308641975307  f1 test:  0.6565656565656566

K 21
train accuracy :  0.5614285714285714  test accuracy:  0.52
f1 train:  0.6933066933066934  f1 test:  0.6470588235294117

K 22
train accuracy :  0.5714285714285714  test accuracy:  0.5266666666666666
f1 train:  0.6957403651115619  f1 test:  0.6467661691542289

K 23
train accuracy :  0.5614285714285714  test accuracy:  0.5133333333333333
f1 train:  0.6933066933066934  f1 test:  0.6439024390243903

K 24
train accuracy :  0.57  test accuracy:  0.5133333333333333
f1 train:  0.6956521739130435  f1 test:  0.6386138613861385

K 25
train accuracy :  0.5571428571428572  test accuracy:  0.52
f1 train:  0.6906187624750499  f1 test:  0.6487804878048781

K 26
train accuracy :  0.57  test accuracy:  0.5233333333333333
f1 train:  0.696266397578204  f1 test:  0.6486486486486487

K 27
train accuracy :  0.5528571428571428  test accuracy:  0.5166666666666667
f1 train:  0.689175769612711  f1 test:  0.648910411622276
```

```
K 28
train accuracy :  0.5585714285714286  test accuracy:  0.53
f1 train:  0.6894472361809044  f1 test:  0.6552567237163814

K 29
train accuracy :  0.5571428571428572  test accuracy:  0.5133333333333333
f1 train:  0.6912350597609562  f1 test:  0.647342995169082
```

[28]:

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score

#initializing classifier
neigh = KNeighborsClassifier(n_neighbors=12)

#training data
neigh.fit(x_train_bow,y_train)

#test the training data
y_pred_train = neigh.predict(x_train_bow)
accuracy_train = accuracy_score(y_pred_train,y_train)
f1_train = f1_score(y_pred_train,y_train)

#test the testing data
y_pred_test = neigh.predict(x_test_bow)
accuracy_test = accuracy_score(y_pred_test,y_test)
f1_test = f1_score(y_pred_test,y_test)

print('train accuracy : ',accuracy_train,' test accuracy :',accuracy_test)
print('f1 train : ',f1_train,' f1 test: ',f1_test)
```

```
train accuracy :  0.64  test accuracy : 0.54
f1 train :  0.7206208425720619  f1 test:  0.6310160427807485
```

[29]:

```python
from sklearn.metrics import classification_report
target_names = ['Postive', 'Negative']
print(classification_report(y_pred_test, y_test, target_names=target_names))
print(classification_report(y_pred_train, y_train, target_names=target_names))
```

```
              precision    recall  f1-score   support

     Postive       0.27      0.68      0.39        65
    Negative       0.85      0.50      0.63       235

    accuracy                           0.54       300
   macro avg       0.56      0.59      0.51       300
weighted avg       0.72      0.54      0.58       300
```

```
              precision    recall  f1-score   support

     Postive       0.36      0.77      0.49       160
    Negative       0.90      0.60      0.72       540

    accuracy                           0.64       700
   macro avg       0.63      0.69      0.61       700
weighted avg       0.78      0.64      0.67       700
```

[30]:
```python
from sklearn.model_selection import GridSearchCV

parameters = {'n_neighbors':list(range(10,30,2))}
neigh = KNeighborsClassifier()

clf = GridSearchCV(neigh, parameters)
clf.fit(x_train_bow, y_train)
```

[30]:
```
GridSearchCV(estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': [10, 12, 14, 16, 18, 20, 22, 24, 26,
                                         28]})
```

[31]:
```python
clf.best_params_
```

[31]:
```
{'n_neighbors': 12}
```

[32]:
```python
neigh = KNeighborsClassifier(n_neighbors=12, p=2)
neigh.fit(x_train_bow, y_train)

y_pred_train = clf.predict(x_train_bow)
f1_train = f1_score(y_pred_train,y_train)
print(f1_train)
print(classification_report(y_pred_train, y_train, target_names=target_names))
```

```
0.7206208425720619
              precision    recall  f1-score   support

     Postive       0.36      0.77      0.49       160
    Negative       0.90      0.60      0.72       540

    accuracy                           0.64       700
   macro avg       0.63      0.69      0.61       700
weighted avg       0.78      0.64      0.67       700
```

[33]:
```
y_pred_test = clf.predict(x_test_bow)
f1_test = f1_score(y_pred_test,y_test)
print(f1_test)
print(classification_report(y_pred_test, y_test, target_names=target_names))
```

```
0.6310160427807485
              precision    recall  f1-score   support

     Postive       0.27      0.68      0.39        65
    Negative       0.85      0.50      0.63       235

    accuracy                           0.54       300
   macro avg       0.56      0.59      0.51       300
weighted avg       0.72      0.54      0.58       300
```

0.0.1 Decision Tree

[45]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix
```

[41]:
```
dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(x_train_bow, y_train)
```

[41]:
```
DecisionTreeClassifier(random_state=42)
```

[54]:
```
y_pred_dt=deci.predict(x_test_bow)
```

[55]:
```
score=accuracy_score(y_test,y_pred)
score
```

[55]:
```
0.67
```

[56]:
```
cm_dt= confusion_matrix(y_test, y_pred)
cm_dt
```

[56]:
```
array([[108,  53],
       [ 46,  93]], dtype=int64)
```

[57]:
```
classification_report(y_test, y_pred)
```

[57]:
```
'              precision    recall  f1-score   support\n\n           0
0.70      0.67      0.69       161\n           1       0.64      0.67      0.65
139\n\n    accuracy                           0.67       300\n   macro avg
0.67      0.67      0.67       300\nweighted avg       0.67      0.67      0.67
300\n'
```

0.0.2 Random Forest

[58]:
```
from sklearn.ensemble import RandomForestClassifier
```

[59]:
```
rf_classifier = RandomForestClassifier(random_state=42)

rf_classifier.fit(x_train_bow, y_train)
```

[59]:
```
RandomForestClassifier(random_state=42)
```

[60]:
```
y_pred_rf = rf_classifier.predict(x_test_bow)
```

[63]:
```
accuracy_rf = accuracy_score(y_test, y_pred_rf)
accuracy_rf
```

[63]:
```
0.7766666666666666
```

[64]:
```
confusion_matrix(y_test, y_pred_rf)
```

[64]:
```
array([[120,  41],
       [ 26, 113]], dtype=int64)
```

0.0.3 Naive Bayes

[65]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
```

[66]:
```
vectorizer = CountVectorizer()
x_train_bow = vectorizer.fit_transform(x_train)
x_test_bow = vectorizer.transform(x_test)
```

[67]:
```
nb_classifier = MultinomialNB()
```

[68]:
```
nb_classifier.fit(x_train_bow, y_train)
```

[68]:
```
MultinomialNB()
```

[69]:
```
y_pred_nb = nb_classifier.predict(x_test_bow)
```

[70]:
```
accuracy_nb = accuracy_score(y_test, y_pred_nb)
accuracy_nb
```

[70]:
```
0.79
```

[71]:
```
confusion_matrix(y_test, y_pred_nb)
```

[71]: array([[137, 24],
       [39, 100]], dtype=int64)

0.0.4 Logistice Regression

[72]:
```
from sklearn.linear_model import LogisticRegression
```

[73]:
```
x_train_bow = vectorizer.fit_transform(x_train)
x_test_bow = vectorizer.transform(x_test)
```

[75]:
```
logistic_classifier = LogisticRegression(random_state=42)
logistic_classifier.fit(x_train_bow, y_train)
```

[75]: LogisticRegression(random_state=42)

[76]:
```
y_pred_logistic = logistic_classifier.predict(x_test_bow)
```

[78]:
```
accuracy_logistic = accuracy_score(y_test, y_pred_logistic)
accuracy_logistic
```

[78]: 0.8033333333333333

[ ]:
```
```