Movie_IMDB_review_sentiment_analysis

Kumar Gaurav , *M.sc(Data Science) CHRIST UNIVERSITY

This dataset is availble on kaggle. I downloaded from that. I used NLP method and ML technique for predict sentiment analysis purpose

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
import nltk
import spacy
import re,string,unicodedata
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelBinarizer
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud,STOPWORDS
from nltk.tokenize import word_tokenize,sent_tokenize
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.stem import LancasterStemmer,WordNetLemmatizer
```

In [3]:

```
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from textblob import TextBlob
from textblob import Word
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
from bs4 import BeautifulSoup
```

In [4]:

```
movie = pd.read_csv('Movie_review.csv') # dataset from kaggle
```

In [5]:

```
movie.head()
```

Out[5]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. 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 [6]:

```
movie.tail()
```

Out[6]:

	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 [7]:

```
movie.isnull().any()  # there is no any null values
```

Out[7]:

```
review       False
sentiment    False
dtype: bool
```

In [8]:

```
movie["sentiment"].count()
```

Out[8]:

```
50000
```

In [11]:

```
movie.groupby('sentiment').count()
```

Out[11]:

sentiment	review
negative	25000
positive	25000

In [12]:

```
movie.shape
```

Out[12]:

```
(50000, 2)
```

In [14]:

```
movie.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 [15]:

```
movie.describe()
```

Out[15]:

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

In [16]:

```
nltk.download('stopwords')
```

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

Out[16]:

```
True
```

Text Tokenization

In [18]:

```
token_ = ToktokTokenizer()
```

In [19]:

```
stopwords = nltk.corpus.stopwords.words('english')
```

In [20]:

```
stopwords
```

```
 'she',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
 'who',
 'whom',
 'this',
 'that',
```

In [25]:

```
# Noisy text removing
def noiseremoval_text(text):
    soup = BeautifulSoup(text,"html.parser")
    text = soup.get_text()
    text = re.sub('\[[^]]*\]','',text)
    return text
```

In [26]:

```
movie['review'] = movie['review'].apply(noiseremoval_text)
```

stemming text

In [31]:

```
# text semming
def stemmer(text):

    ps = nltk.porter.PorterStemmer()
    text = ' '.join([ps.stem(word) for word in text.split()])
    return text
```

In [32]:

```
# apply function on review column
movie['review'] = movie['review'].apply(stemmer)
```

In [33]:

```
movie.head()
```

Out[33]:

	review	sentiment
0	one of the other review ha mention that after ...	positive
1	a wonder littl production. the film techniqu i...	positive
2	i thought thi wa a wonder way to spend time on...	positive
3	basic there' a famili where a littl boy (jake)...	negative
4	petter mattei' "love in the time of money" is ...	positive

In [34]:

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

In [35]:

```
# set stopwords to english
stop_word = set(stopwords.words('english'))
print(stop_word)
```

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

In [46]:

```
# removing the stopwords
def removing_stopwords(text, is_lower_case = False):
    # Tokenization of text
    tokenizers = ToktokTokenizer()
    tokens = tokenizers.tokenize(text)
    tokens = [token.strip() for token in tokens]
    if is_lower_case:
        filtered_tokens = [token for token in tokens if token not in stop_word]
    else:
        filtered_tokens = [token for token in tokens if token.lower() not in stop_word]
    filtered_text = ' '.join(filtered_tokens)
    return filtered_text
```

In [47]:

```
movie['review'] = movie['review'].apply(removing_stopwords)
```

In [48]:

```
movie.head()
```

Out[48]:

	review	sentiment
0	one review ha mention watch 1 oz episod ' hook...	positive
1	wonder littl production. film techniqu veri un...	positive
2	thought thi wa wonder way spend time hot summe...	positive
3	basic ' famili littl boy (jake) think ' zomb...	negative
4	petter mattei ' " love time money " visual stu...	positive

Train,test,split the datset

In [49]:

```
train_reviews_movie = movie.review[:30000] # for train dataset
```

In [50]:

```
test_review_movie = movie.review[30000:] # for test dataset
```

Now, we use NLP model technique

1. bag of words

2. TF - IDF

3. Label encoding

In [51]:

```
## BAG of WORDS
cv = CountVectorizer(min_df = 0 , max_df = 1, binary = False , ngram_range = (1,3))
```

In [52]:

```
cv_train = cv.fit_transform(train_reviews_movie)
cv_test = cv.transform(test_review_movie)
```

In [53]:

```
cv_train.shape
```

Out[53]:

```
(30000, 4954557)
```

In [54]:

```
cv_test.shape
```

Out[54]:

```
(20000, 4954557)
```

now TF-IDF

In [55]:

```
#TF-IDF
tf = TfidfVectorizer(min_df = 0 , max_df = 1, use_idf = True,ngram_range = (1,3))
```

In [56]:

```
tf_train = tf.fit_transform(train_reviews_movie)
```

In [57]:

```
tf_test = tf.transform(test_review_movie)
```

In [58]:

```
tf_train.shape
```

Out[58]:

```
(30000, 4954557)
```

In [59]:

```
tf_test.shape
```

Out[59]:

```
(20000, 4954557)
```

Label_encoding

In [60]:

```
label  = LabelBinarizer()
sentimentOfmovie = label.fit_transform(movie['sentiment'])
```

In [61]:

```
sentimentOfmovie.shape
```

Out[61]:

```
(50000, 1)
```

In [62]:

```
train_movie_sentiment = movie.sentiment[:30000]
```

In [63]:

```
test_movie_sentiment = movie.sentiment[30000:]
```

Logistic model

In [64]:

```
logistic=LogisticRegression(penalty='l2',max_iter=500,C=1,random_state=42)
#Fitting the model for Bag of words
lr_bow=logistic.fit(cv_train,train_movie_sentiment)
print(lr_bow)
```

```
LogisticRegression(C=1, max_iter=500, random_state=42)
```

In [65]:

```
#Fitting the model for tfidf features
lr_tfidf=logistic.fit(tf_train,train_movie_sentiment)
print(lr_tfidf)
```

```
LogisticRegression(C=1, max_iter=500, random_state=42)
```

Predict _ model

In [66]:

```
#Predicting the model for bag of words
lr_bow_predict=logistic.predict(cv_test)
print(lr_bow_predict)
```

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

In [67]:

```
#Predicting the model for tfidf features
lr_tfidf_predict=logistic.predict(tf_test)
print(lr_tfidf_predict)
```

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

Model Accuracy

In [68]:

```
#Accuracy score for bag of words
lr_bow_score=accuracy_score(test_movie_sentiment,lr_bow_predict)
print("lr_bow_score :",lr_bow_score)
```

```
lr_bow_score : 0.74255
```

In [69]:

```
#Accuracy score for tfidf features
lr_tfidf_score=accuracy_score(test_movie_sentiment,lr_tfidf_predict)
print("lr_tfidf_score :",lr_tfidf_score)
```

```
lr_tfidf_score : 0.7426
```

Classification_report

In [70]:

```
lr_class_score=classification_report(test_movie_sentiment,lr_bow_predict)
print("lr_bow_score :",lr_class_score)
```

```
lr_bow_score :               precision    recall  f1-score   support

    negative       0.75      0.73      0.74     10015
    positive       0.74      0.75      0.74      9985

    accuracy                           0.74     20000
   macro avg       0.74      0.74      0.74     20000
weighted avg       0.74      0.74      0.74     20000
```

In [71]:

```
lr_tfidf_class_report=classification_report(test_movie_sentiment,lr_tfidf_predict)
print("lr_tfidf_score :",lr_tfidf_class_report)
```

```
lr_tfidf_score :               precision    recall  f1-score   support

    negative       0.75      0.74      0.74     10015
    positive       0.74      0.75      0.74      9985

    accuracy                           0.74     20000
   macro avg       0.74      0.74      0.74     20000
weighted avg       0.74      0.74      0.74     20000
```

Conclusion - I tried many method in background like naive,svm but logistic give me high accuracy , It may give us more high accuracy but will be more work on that.