Movie Sentiment Analysis using Python

1. Introduction

1.1 Overview

1.2 Purpose

2. Literature Survey

2.1 Existing Problem

2.2 Proposed Solution

3. Theoretical Analysis

3.1 Block Diagram

3.2 Hardware/Software Requirement

4. Result

5. Advantages and Disadvantages

6. Application

7. Conclusion

8.Reference

1 Introduction:

1.1 Overview

NLP stands for natural language processing which is basically used to understand and interpret human language to the machine. In short it is the automatic way to manipulate the natural language, like speech and text, by software for further analysis to get the required information from them

1.2 Purpose

Examples of Natural language processing Proprietary content.

♣ Email filters

♣ Data analysis

♣ Language translation

♣ Smart assistants

2 Literature Survey

2.1 Existing Problem

2.2 Proposed Problem

Source Code:

import numpy as np

import pandas as pd

data=pd.read\_csv("/content/IMDB Dataset.csv")

data.head()

data.columns

data.isnull().any()

data.isnull().sum()

data.describe()

data['sentiment'].value\_counts()

data.shape

Text normalization

## **tokenization**

import seaborn as sns

import matplotlib.pyplot as plt

import nltk

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.preprocessing import LabelBinarizer

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

from wordcloud import WordCloud,STOPWORDS

from nltk.stem import WordNetLemmatizer

from nltk.tokenize import word\_tokenize,sent\_tokenize

import spacy

import re,string,unicodedata

from nltk.tokenize.toktok import ToktokTokenizer

from nltk.stem import LancasterStemmer,WordNetLemmatizer

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

import nltk

nltk.download('stopwords')

#Tokenization of text

tokenizers=ToktokTokenizer()

#Setting English stopwords

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

#Removing the noisy text

def noiseremoval\_text(text):

  soup = BeautifulSoup(text, "html.parser")

  text = soup.get\_text()

  text = re.sub('\[[^]]\*\]', '', text)

  return text

#Apply function on review column

data['review']=data['review'].apply(noiseremoval\_text)

data.head()

## **Stemming**

#Stemming the text

def stemmer(text):

    ps=nltk.porter.PorterStemmer()

    text= ' '.join([ps.stem(word) for word in text.split()])

    return text

#Apply function on review column

data['review']=data['review'].apply(stemmer)

data.head()

## **Removing stop words**

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

#set stopwords to english

stop\_wr=set(stopwords.words('english'))

print(stop\_wr)

#removing the stopwords

def removing\_stopwords(text, is\_lower\_case=False):

    #Tokenization of text

    tokenizers=ToktokTokenizer()

    #Setting English stopwords

    tokens = tokenizers.tokenize(text)

    tokens = [token.strip() for token in tokens]

    if is\_lower\_case:

        filter\_tokens = [token for token in tokens if token not in stop\_wr]

    else:

        filter\_tokens = [token for token in tokens if token.lower() not in stop\_wr]

    filtered\_text = ' '.join(filter\_tokens)

    return filtered\_text

#Apply function on review column

data['review']=data['review'].apply(removing\_stopwords)

data.head()

## **Train test split**

#split the dataset

#train dataset

train\_reviews\_data=data.review[:30000]

#test dataset

test\_reviews\_data=data.review[30000:]

## **Bag of words**

#Count vectorizer for bag of words

cv=CountVectorizer(min\_df=0,max\_df=1,binary=False,ngram\_range=(1,3))

#transformed train reviews

cv\_train=cv.fit\_transform(train\_reviews\_data)

#transformed test reviews

cv\_test=cv.transform(test\_reviews\_data)

print('BOW\_cv\_train:',cv\_train.shape)

print('BOW\_cv\_test:',cv\_test.shape)

#vocab=cv.get\_feature\_names()-toget feature names

## **TF\_IDF**

#Tfidf vectorizer

tf=TfidfVectorizer(min\_df=0,max\_df=1,use\_idf=True,ngram\_range=(1,3))

#transformed train reviews

tf\_train=tf.fit\_transform(train\_reviews\_data)

#transformed test reviews

tf\_test=tf.transform(test\_reviews\_data)

print('Tfidf\_train:',tf\_train.shape)

print('Tfidf\_test:',tf\_test.shape)

## **Lable encoding**

#labeling the sentient data

label=LabelBinarizer()

#transformed sentiment data

sentiment\_data=label.fit\_transform(data['sentiment'])

print(sentiment\_data.shape)

train\_data=data.sentiment[:30000]

test\_data=data.sentiment[30000:]

#training the model

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\_data)

print(lr\_bow)

#Fitting the model for tfidf features

lr\_tfidf=logistic.fit(tf\_train,train\_data)

print(lr\_tfidf)

#Predicting the model for bag of words

lr\_bow\_predict=logistic.predict(cv\_test)

print(lr\_bow\_predict)

##Predicting the model for tfidf features

lr\_tfidf\_predict=logistic.predict(tf\_test)

print(lr\_tfidf\_predict)

#Accuracy score for bag of words

lr\_bow\_score=accuracy\_score(test\_data,lr\_bow\_predict)

print("lr\_bow\_score :",lr\_bow\_score)

#Accuracy score for tfidf features

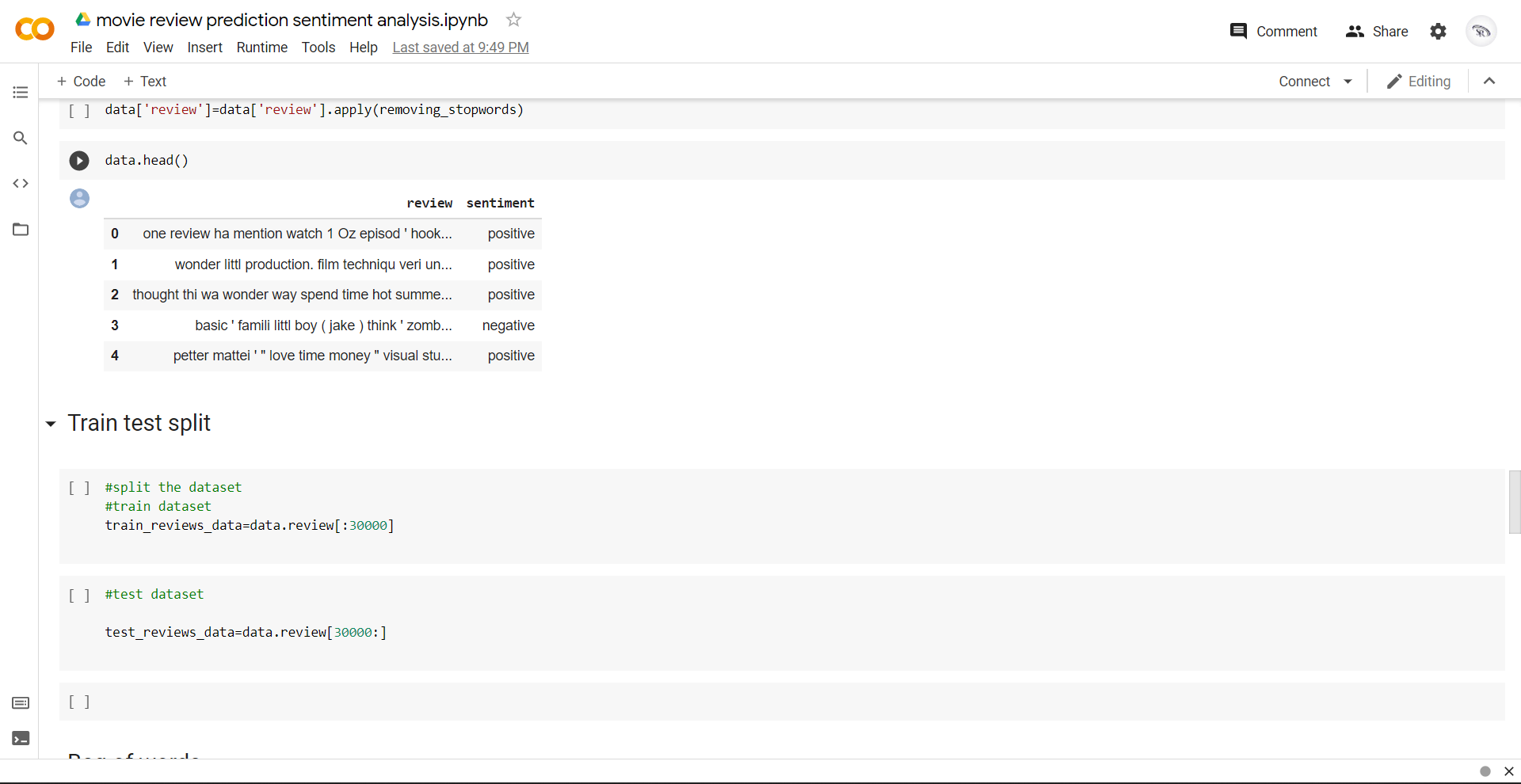
lr\_tfidf\_score=accuracy\_score(test\_data,lr\_tfidf\_predict)

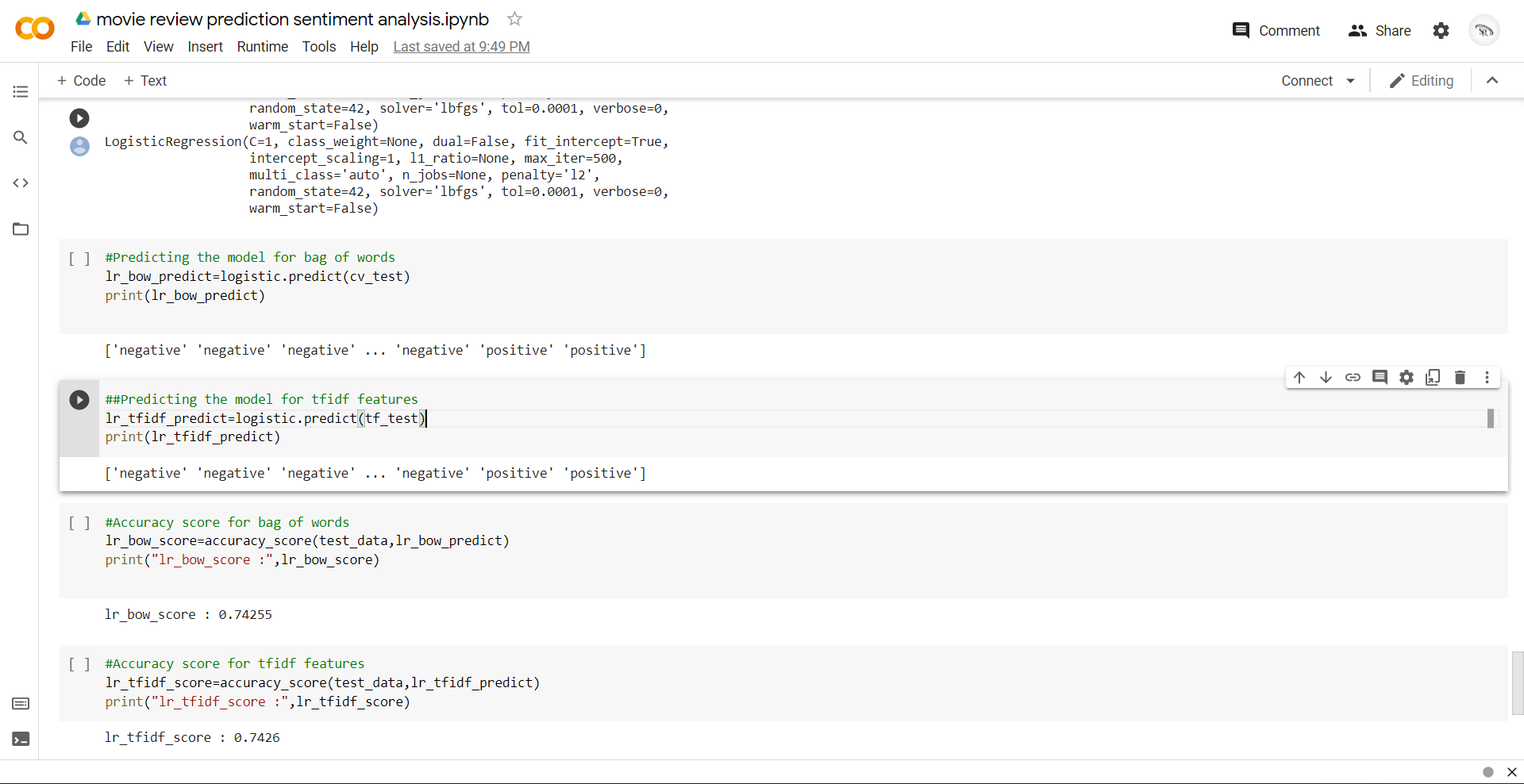
print("lr\_tfidf\_score :",lr\_tfidf\_score)

3. Theoretical Analysis:

Block Diagram

4. Result





Software requirements:

* Google -colab,
* python

hardware requirements:

* 4 GB ram
* Windows 10
* Internet access

5. Advantages and Disadvantages

ADVANTAGES:

* We can analysis the movie easily using by google colab
* It reduces the time
* It executes fast

Disadvantages:

* There is a lack of labelled data
* The range is about scalar type

6. Application

Social media monitoring

Customer support

Customer feedback

Brand monitoring and reputation management

Voice of customer (VoC)

Voice of employee

Product analysis

Market research and competitive research

7.Conclusion

8. Reference

[1] Sentiment Analysis – Wikipedia – <https://en.wikipedia.org/wiki/Sentiment_analysis>

[2] Large Movie Review Dataset– <http://ai.stanford.edu/~amaas/data/sentiment/>

[3] Andrew L Mass, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng and Christopher Potts (2011). Learning Word Vectors for Sentiment Analysis

[4] Internet Movie Database– <http://www.imdb.com/>

[5] Cross Validation– Wikipedia <https://en.wikipedia.org/wiki/Crossvalidation_%28statistics%29>

[6] NLTK Stopwords Corpus: <http://www.nltk.org/book/ch02.html>