SENTIMENT ANALYSIS USING USING BI-DIRECTIONAL LSTM

B.Tech in Computer Science and Engineering **(CSE)**, **Winter** Semester **2019-20**

Name:	Harish Bharadwaj
Registration Number:	18BCE0078
Slot:	F1+TF1
Faculty Name:	GOPALAKRISHNAN T

SUBMITTED BY
APOORV SINGH - 18BCE0007
HARISH BHARDWAJ - 18BCE0078

DECLARATION:

I hereby declare that the thesis entitled “SENTIMENT ANALYSIS USING BI-DIRECTIONAL LSTM" submitted by me, for the award of the degree of Bachelor of Technology in Programme to VIT is a record of bonafide work carried out by me under the supervision of GOPALAKRISHNAN T.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

SIGNATURE OF CANDIDATES

PLACE: VELLORE HARISH BHARDWAJ

DATE: 05/06/2020 APOORV SINGH

CERTIFICATE:

This is to certify that the thesis entitled "SENTIMENT ANALYSIS USING BI-DIRECTIONAL LSTM" submitted by APOORV SINGH & 18BCE0007 and HARISH BHARDWAJ & 18BCE0078, SCOPE, VIT, for the award of the degree of Bachelor of Technology in Programme, is a record of bonafide work carried out by him / her under my supervision during the period 01. 12. 2018 to 30.04.2019, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

PLACE: VELLORE
DATE:

SUBMITTED TO
GOPALAKRISHNAN T

HOD
V SANTHI

ACKNOWLEDGEMENTS

A deepest gratitude and sincere thanks to Prof. Gopalakrishnan.T. in helping us complete our Project with several learning outcomes.
We feel deeply obliged to thank the SCOPE (School of Computer Science and Engineering) Department and the VIT University for their services rendered and for giving us an opportunity to carry out our studies at the University.

TEAM MEMBERS
APOORV SINGH - 18BCE0007
HARISH BHARDWAJ - 18BCE0078

CONTENTS PAGE

1. DECLARATION 2
2. CERTIFICATE 3
3. ACKNOWLEDGMENT 4
4. CONTENTS PAGE 5
5. INTRODUCTION 6
6. PRODUCT DESCRIPTION AND GOALS 7
7. TECHNICAL SPECIFICATIONS 8
8. DESIGN APPROACH AND DETAILS 15
9. PROJECT DEMONSTRATION 18
10. TABLES 25
11. GRAPHS 26
12. CONCLUSION 27
13. SUMMARY & RESULT 27
14. REFERENCES 27

INTRODUCTION:

One day, we were surfing the internet for a product. We found that the same product is being manufactured by many different companies and at different price points. Then we applied a price filter and got the price range at which we wanted to buy the product but still there were lots of products to choose from. So, we started to check the ratings after an hour or so we shortlisted some 10 products.

Now from here we decided to read the reviews written by purchasers for each product and we found out that there were too many reviews to be read by us. We were really exhausted and needed some assistance.

At this point we started thinking, "what if there existed some tool that will help us know really fast how much positive and negative reviews are there on a particular product?" And then we will be able to compare the percentage of positive reviews and buy the product easily.

We started surfing the internet for this kind of product but we found nothing similar and got disappointed. We started thinking about all the features through which we can make this possible and started researching these things. In the process of research, we found that this can be done using some machine learning libraries and some algorithms which we will be discussing in coming sections.

We started learning machine learning but still we were skeptical that "we are really going to get something out of this." Initially we thought of doing it using Naive Bayes as we were amateurs but then we learnt about the "bi-directional LSTM" algorithm.

We decided to make this project using the bi-directional LSTM and also compare the Naive Bayes to this bi-directional LSTM so that we can see that what would have happened if we had had made the project using Naive Bayes and not the bi-directional LSTM. Of Course we have provided the difference between the Naive Bayes in the coming sections.

So, this was a small story on how we ended on this project. I hope you like our project.

PRODUCT DESCRIPTION AND GOALS:

Whenever we go to buy a product online, there we search for reviews. Why are these reviews crucial? Well, this review helps us to understand the product better. These reviews help us to know what are the weak points of the product, if the product is suited for you or not and many other crucial things. But today, we are having lots and lots of reviews on a single product itself and it is tiering to sort through all those reviews to find if you are making the right decision or not. Many will say that this product already exists. So, what is the one thing our project does differently? Let’s explore.

Companies like Amazon and flipkart have provided the option of rating a product and you can write a review as well but many times you will come across a negative review in which the person has given 3-4 stars. Now this could be misleading. At this point of time our product comes in handy to help you find the numbers of positive and negative reviews based on the words that have been used in each review.

Initially, we were using Naive Bayes to do our project. Later we implemented bi-directional LSTM and we found out that it is much better. So, now our project also includes a comparison between Naive Bayes and bi-directional LSTM.

TECHNICAL SPECIFICATIONS

We used the following libraries:

- pandas
- numpy
- scikit learn
- nltk
- re(regular expression)
- unicode data
- selenium
- bs4
- urllib
- keras

Now let us talk about each library in detail

PANDAS:

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

NUMPY:

It is a library in python which contains all the algebraic functions along with arrays and data manipulation tools. It also contains functions for fourier series and matrices.

SCIKIT LEARN:

This is the library which contains most of the templates for training machine learning models. It uses numpy for high performance linear algebra and array operations.

NLTK:

This library is used for natural language processing. It is also known as a natural language toolkit. It is used for the english language.

RE(REGULAR EXPRESSION):

It specifies a set of strings that matches it. The function checks that if a string matches the given regular expression. Regular expressions can be concatenated to form new regular expressions.

UNICODE DATA:

It provides access to the unicode character database. It defines the property of all unicode data characters.

unicodedata.lookup(name)
Look up character by name. If a character with the given name is found, return the corresponding character. If not found, KeyError is raised.

Changed in version 3.3: Support for name aliases 1 and named sequences 2 has been added.

unicodedata.name(chr[, default])
Returns the name assigned to the character chr as a string. If no name is defined, default is returned, or, if not given, ValueError is raised.

unicodedata.decimal(chr[, default])
Returns the decimal value assigned to the character chr as integer. If no such value is defined, default is returned, or, if not given, ValueError is raised.

unicodedata.digit(chr[, default])
Returns the digit value assigned to the character chr as integer. If no such value is defined, default is returned, or, if not given, ValueError is raised.

unicodedata.numeric(chr[, default])
Returns the numeric value assigned to the character chr as float. If no such value is defined, default is returned, or, if not given, ValueError is raised.

unicodedata.category(chr)
Returns the general category assigned to the character chr as string.

unicodedata.bidirectional(chr)
Returns the bidirectional class assigned to the character chr as string. If no such value is defined, an empty string is returned.

unicodedata.combining(chr)
Returns the canonical combining class assigned to the character chr as integer. Returns 0 if no combining class is defined.

unicodedata.east_asian_width(chr)
Returns the east asian width assigned to the character chr as string.

unicodedata.mirrored(chr)
Returns the mirrored property assigned to the character chr as integer. Returns 1 if the character has been identified as a “mirrored” character in bidirectional text, 0 otherwise.

unicodedata.decomposition(chr)
Returns the character decomposition mapping assigned to the character chr as string. An empty string is returned in case no such mapping is defined.

unicodedata.normalize(form, unistr)
Return the normal form form for the Unicode string unistr. Valid values for form are ‘NFC’, ‘NFKC’, ‘NFD’, and ‘NFKD’.

SELENIUM:

It is a tool to test your web application.

BS4:

It is used for web scraping in python with beautifulsoup.There are mainly two ways to extract data from a website:

- Use the API of the website (if it exists). For example, Facebook has the Facebook Graph API which allows retrieval of data posted on Facebook.
- Access the HTML of the webpage and extract useful information/data from it. This technique is called web scraping or web harvesting or web data extraction.

We are going to take out the reviews from the sites using bs4.

URLLIB:

urllib is a package that collects several modules for working with URLs:

- urllib.request for opening and reading URLs
- urllib.error containing the exceptions raised by urllib.request
- urllib.parse for parsing URLs
- urllib.robotparser for parsing robots.txt files

KERAS:

It is tensorflow’s high-level API for building and training deep learning models. It is used for fast prototyping, state-of-art research and production with three different advantages:

- User-friendly
  Keras has a simple, consistent interface optimized for common use cases. It provides clear and actionable feedback for user errors.

- Modular and composable
  Keras models are made by connecting configurable building blocks together, with few restrictions.

- Easy to extend
  Write custom building blocks to express new ideas for research. Create new layers, metrics, loss functions, and develop state-of-the-art models.

AI MODELS WE USED :

There are two models widely used for bi-class classification.
The input is classified into two classes like e.g +ve or a -ve one
The two models implemented are:
1.Gaussian Naive Bayes Model (follows bag of word technique)
2.Bi-Directional LSTM (follows deep neural network technique)

Naive Bayes

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

For Sentiment Analysis:

We need to predict whether it is a positive review or negative one.

So we use:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

where y is being a positive review or a negative one and X is :

$$X = (x_1, x_2, x_3, ..., x_n)$$

ADVANTAGES OF NAIVE BAYES:

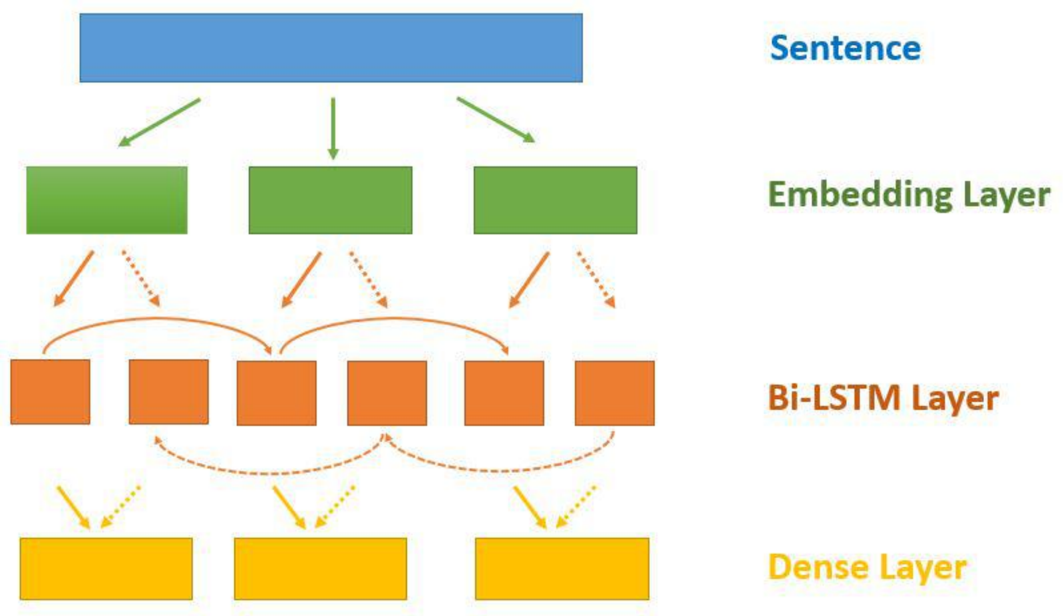
1. The model is simple but effective enough
2. Easy training and to deploy model
3. Handling of both continuous and discrete data can be done using Naive Bayes
4. Numbers of predictors and data points are highly predictable
5. It is not sensitive to irrelevant features.
6. When assumption of independent predictors holds true, a Naive Bayes classifier performs better as compared to other models.
7. Naive Bayes requires a small amount of training data to estimate the test data. So, the training period is less.
8. Easy to implement

DISADVANTAGES OF NAIVE BAYES:

1. The main limitation of Naive Bayes is the assumption of independent predictor features. Naive Bayes implicitly assumes that all the attributes are mutually independent. In real life, it's almost impossible that we get a set of predictors that are completely independent or one another.
2. If a categorical variable has a category in the test dataset, which was not observed in the training dataset, then the model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as Zero Frequency. To solve this, we can use a smoothing technique. Details on additive smoothing or laplace smoothing can be found here.

Bi-Directional LSTM (also known as BRNN) Model:

Our Model Architecture uses a series of :



Recurrent neural networks (which we are about to explore) are a subclass of neural networks, designed to perform a sequence recognition or prediction. They have a flexible number of inputs and they allow cyclical connections between their neurons. This means that they are able to remember previous information and connect it to the current task.

Long Short Term Memory networks (LSTM) are a subclass of RNN, specialized in remembering information for a long period of time. More over the Bidirectional lstms keep the contextual information in both directions.

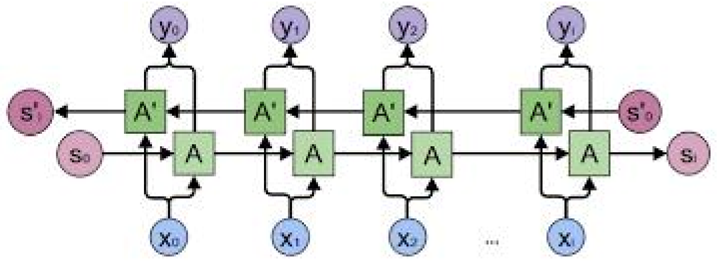
Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems.

In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence.

The principle of BRNN is to split the neurons of a regular RNN into two directions, one for positive time direction (forward states), and another for negative time direction (backward states). Those two states' output are not connected to inputs of the opposite direction states. The general structure of RNN and BRNN can be depicted in the right diagram. By using two time directions, input information from the past and future of the current time frame can be used unlike standard RNN which requires the delays for including future information

BRNNs can be trained using similar algorithms to RNNs, because the two directional neurons do not have any interactions. However, when back-propagation is applied, additional processes are needed because updating input and output layers cannot be done at once. General procedures for training are as follows: For forward pass, forward states and backward states are passed first, then output neurons are passed. For backward pass, output neurons are passed first, then forward states and backward states are passed next. After forward and backward passes are done, the weights are updated.

Architecture of Bi-Directional LSTM :



ADVANTAGES OF BI-DIRECTIONAL LSTM:

1. It analyzes the input sentence in a sequence and that too in both the directions
2. It provides attention to keywords that mainly classify the input sentence
3. This model more accurately predicts the data and classifies them correctly.

DISADVANTAGES OF BI-DIRECTIONAL LSTM:

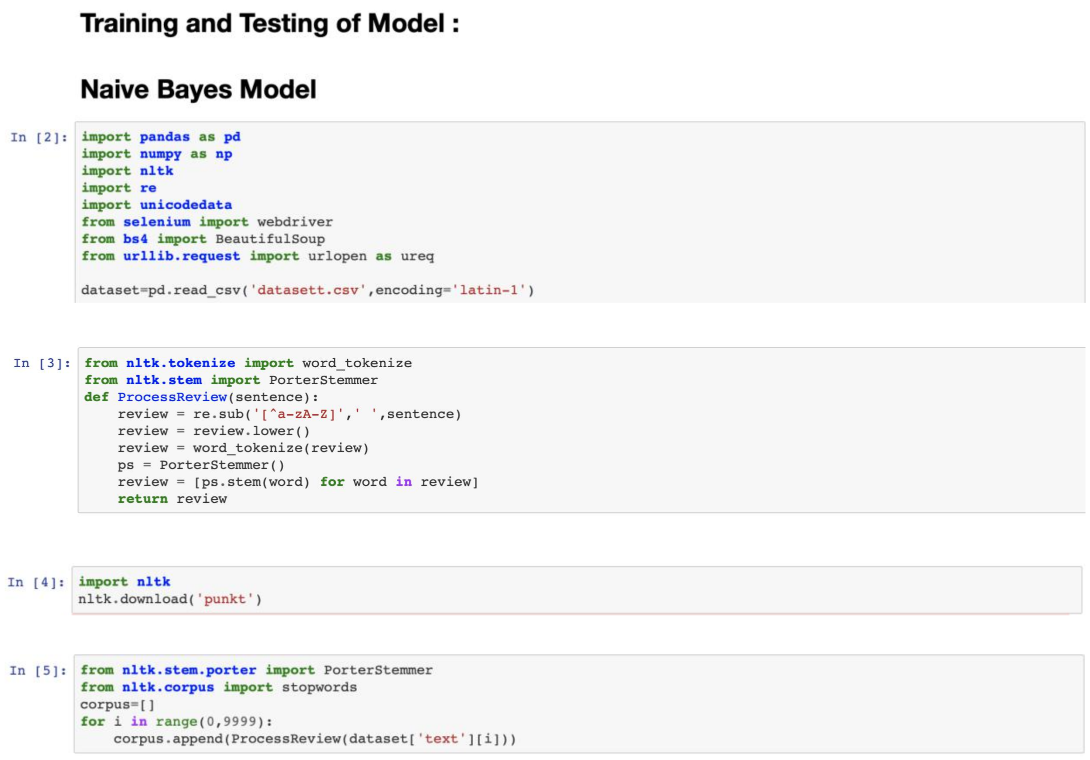
1. Highly computative and requires a lot of RAM or a good GPU.
2. Time consuming and complicated architecture
3. Requires a huge dataset to train the model properly

DESIGN APPROACH AND DETAILS:

Tools required for both models are

1. Python
2. TensorFlow - Google’s open sourced numeric computational library
3. Keras - Neural Network Framework, which can run on top of TensorFlow
4. Numpy - Package for scientific computations
5. Pandas - Package providing easy-to-use data structures and data analysis tools
6. Scikit Learn library
7. NLTK library

OUR CODES:



Training and Testing of Model :

Naive Bayes Model

```
In [2]: import pandas as pd
        import numpy as np
        import nltk
        import re
        import unicodedata
        from selenium import webdriver
        from bs4 import BeautifulSoup
        from urllib.request import urlopen as ureq

        dataset=pd.read_csv('datasett.csv',encoding='latin-1')
```

```
In [3]: from nltk.tokenize import word_tokenize
        from nltk.stem import PorterStemmer
        def ProcessReview(sentence):
            review = re.sub('[^a-zA-Z]',' ',sentence)
            review = review.lower()
            review = word_tokenize(review)
            ps = PorterStemmer()
            review = [ps.stem(word) for word in review]
            return review
```

```
In [4]: import nltk
        nltk.download('punkt')
```

```
In [5]: from nltk.stem.porter import PorterStemmer
        from nltk.corpus import stopwords
        corpus=[]
        for i in range(0,9999):
            corpus.append(ProcessReview(dataset['text'][i]))
```

In [5]:
```
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords
corpus=[]
for i in range(0,9999):
    corpus.append(ProcessReview(dataset['text'][i]))
```

In [6]:
```
corpus1=corpus
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(tokenizer=lambda doc: doc, lowercase=False,max_features=1500)
X=cv.fit_transform(corpus1)
X=X.toarray()
```

In [7]:
```
Y=dataset.iloc[:,0].values
```

In [8]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=0)
```

In [9]:
```
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(X_train,Y_train)
```

In [10]:
```
Y_pred=classifier.predict(X_test)
```

In [11]:
```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(Y_test,Y_pred)
cm
```

In [12]:
```
dataset1=pd.read_csv('products2.csv')
```

In [14]:
```
X_new=cv.transform(corpus2)
X_new=X_new.toarray()
Y_new=classifier.predict(X_new)
Y_new
```

LSTM Model

In [11]:
```
#Data Pre-Processing

import numpy as np
import pandas as pd
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential, load_model
from keras.layers import Dense, Embedding, LSTM, Bidirectional
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
import re

data = pd.read_csv("datasett.csv",encoding='latin-1')

tokenizer = Tokenizer(num_words=2000, split=' ')

tokenizer.fit_on_texts(data['text'])
X = tokenizer.texts_to_sequences(data['text'])
X = pad_sequences(X)
Y = data['sentiment']


X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)
```

In [12]:
```
#Model Creation and Training

model = Sequential()

model.add( Embedding(2000, 128, input_length = X.shape[1], dropout=0.2))
model.add( Bidirectional(LSTM(196, dropout_U = 0.2, dropout_W = 0.2)))
model.add( Dense(2, activation = 'softmax'))

model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.fit(X_train, Y_train, nb_epoch = 7, batch_size = 100, verbose = 2)
```

In [14]:
```
#Model Testing

import numpy as np
y=model.predict(X_test)
Y=np.argmax(y,axis=1)
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test,Y)
```

Deployment of Model :

In [15]:
```
#Data Mining

from selenium import webdriver
from bs4 import BeautifulSoup
from urllib.request import urlopen as ureq
my_url=str(input("enter url"))
uclient=ureq(my_url)
page_html=uclient.read()
uclient.close()
```

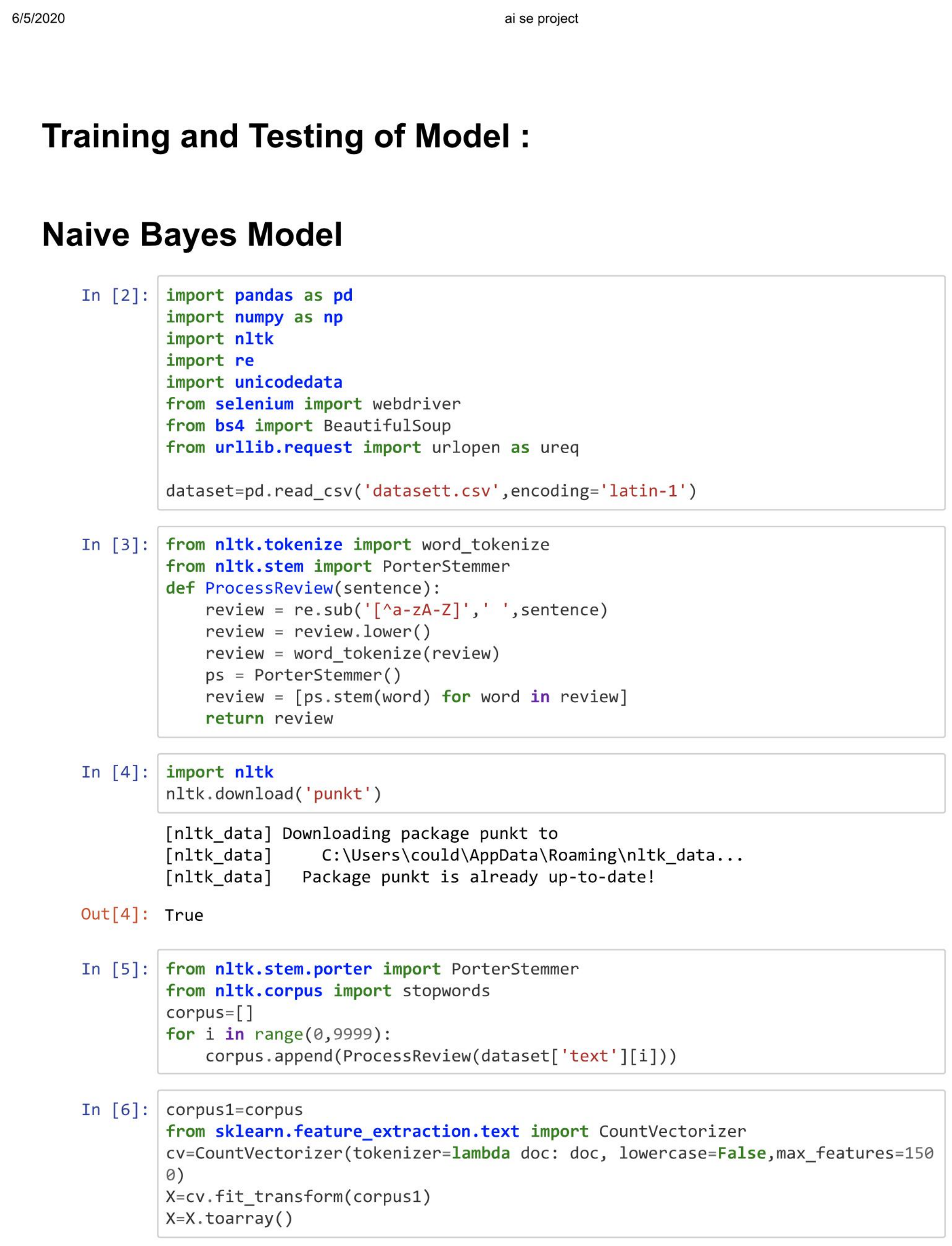
In [18]:

```python
#Data Parsing and Storing Reviews in a csv file

page_soup1=BeautifulSoup(page_html,"html.parser")
totpages=page_soup1.find("div",{"class":"_2zg3yZ _3KSYCY"})
totpages=totpages.span.text.strip()
x=""
while(totpages[-1]!=" "):
    x=x+totpages[-1]
    totpages=totpages[0:len(totpages)-1]
x=int(x[::-1].replace(",",""))
print("pages",x)
filename="products2.csv"
headers="productname,review\n"
f=open(filename,"w",encoding='utf-8')
f.write(headers)
x= 15 if x>15 else x
for z in range(1,x+1):
    my_url=my_url+"&page="+str(z)
    uclient=ureq(my_url)
    page_html=uclient.read()
    uclient.close()
    page_soup=BeautifulSoup(page_html,"html.parser")
    containers=page_soup.findAll("div",{"class":"_1PBCrt"})
    pn=page_soup.find("div",{"class":"_3BTv9X"})
    pn=pn.img["alt"]
    for cont in containers:
        review=cont.findAll("p",{"class":"_2xg6Ul"})
        review=review[0].text
        revwords=review.split()
        if len(revwords)<191:
            f.write(str(pn.replace(",","/"))+","+str(review.replace(",","/"))+"\n")
tp='IT DIDNT WORK WE BUY NEW IT DIE ON ME DON.T BUY IT: THE VHS WE BUY DID NOT WORK THE MATCHS WERE BORING TOO BORING
I KNOW DON.T HOW WWF CAN HAVE A BAD PPV BUT THEY CAN WE BUY NEW IT WAS SO NEW BUT IT DIE ON ME DON.T BUY THIS ON VHS T
HIS VHS WAS NEW BUT DIDNT PLAY I HAVE NOT SEEN ALL OF THE PPV BUT WHAT I HAVE SEEN IT DIDNT PLAY GOOD NOT GOOD AT ALL
IF YOU WANT TO WATCH IT GET ON DVD THAT ALL HAVE TO SAY DON.T BUY THE VHS IF YOU WANT A GOOD PPV BUY HELL IN THE CALL
2012 THAT ONE GOOD i was not happy to have a vhs i buy not work not happy at all it was too old to play good it woods
not work i was sad when it did.nt work i woods not buy a wwf vhs for a 2 time the ppv was poor not that fun to watch i
t the wwf and a bad ppv i want to see a good wwf ppv'
f.write(str(pn.replace(",","/"))+","+tp+"\n")
f.close()
```

In [19]:

```python
#Model Deployment

data2 = pd.read_csv("products2.csv",encoding='latin-1')
tokenizer = Tokenizer(num_words=2000, split=' ')
tokenizer.fit_on_texts(data2['review'])
X2 = tokenizer.texts_to_sequences(data2['review'])
X2 = pad_sequences(X2)
y2=model.predict(X2)
result=np.argmax(y2,axis=1)
posrev=np.count_nonzero(result == 1)
negrev=len(result)-posrev-1
print(f'positive reviews: {posrev}\nnegative reviews: {negrev}')
```

PROJECT DEMONSTRATION:



6/5/2020 ai se project

Training and Testing of Model :

Naive Bayes Model

In [2]:
```
import pandas as pd
import numpy as np
import nltk
import re
import unicodedata
from selenium import webdriver
from bs4 import BeautifulSoup
from urllib.request import urlopen as ureq

dataset=pd.read_csv('datasett.csv',encoding='latin-1')
```

In [3]:
```
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
def ProcessReview(sentence):
    review = re.sub('[^a-zA-Z]',' ',sentence)
    review = review.lower()
    review = word_tokenize(review)
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review]
    return review
```

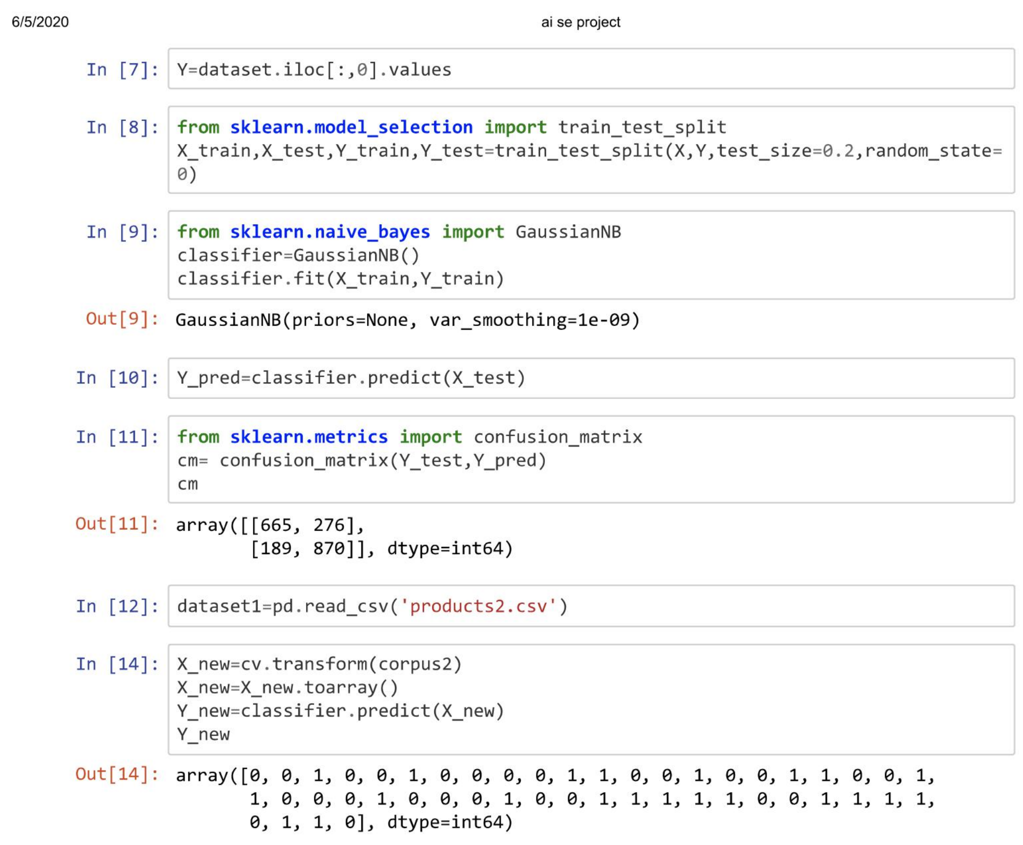
In [4]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\could\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[4]:
```
True
```

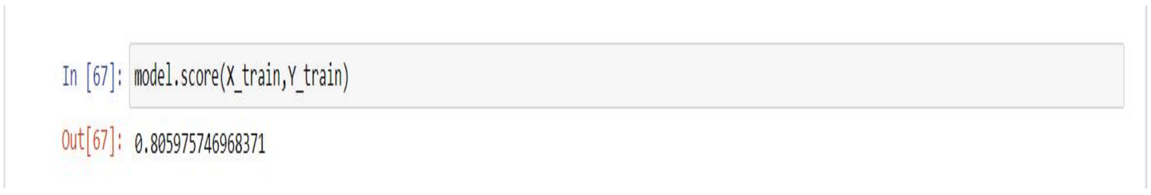
In [5]:
```
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords
corpus=[]
for i in range(0,9999):
    corpus.append(ProcessReview(dataset['text'][i]))
```

In [6]:
```
corpus1=corpus
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(tokenizer=lambda doc: doc, lowercase=False,max_features=150
0)
X=cv.fit_transform(corpus1)
X=X.toarray()
```



6/5/2020 ai se project

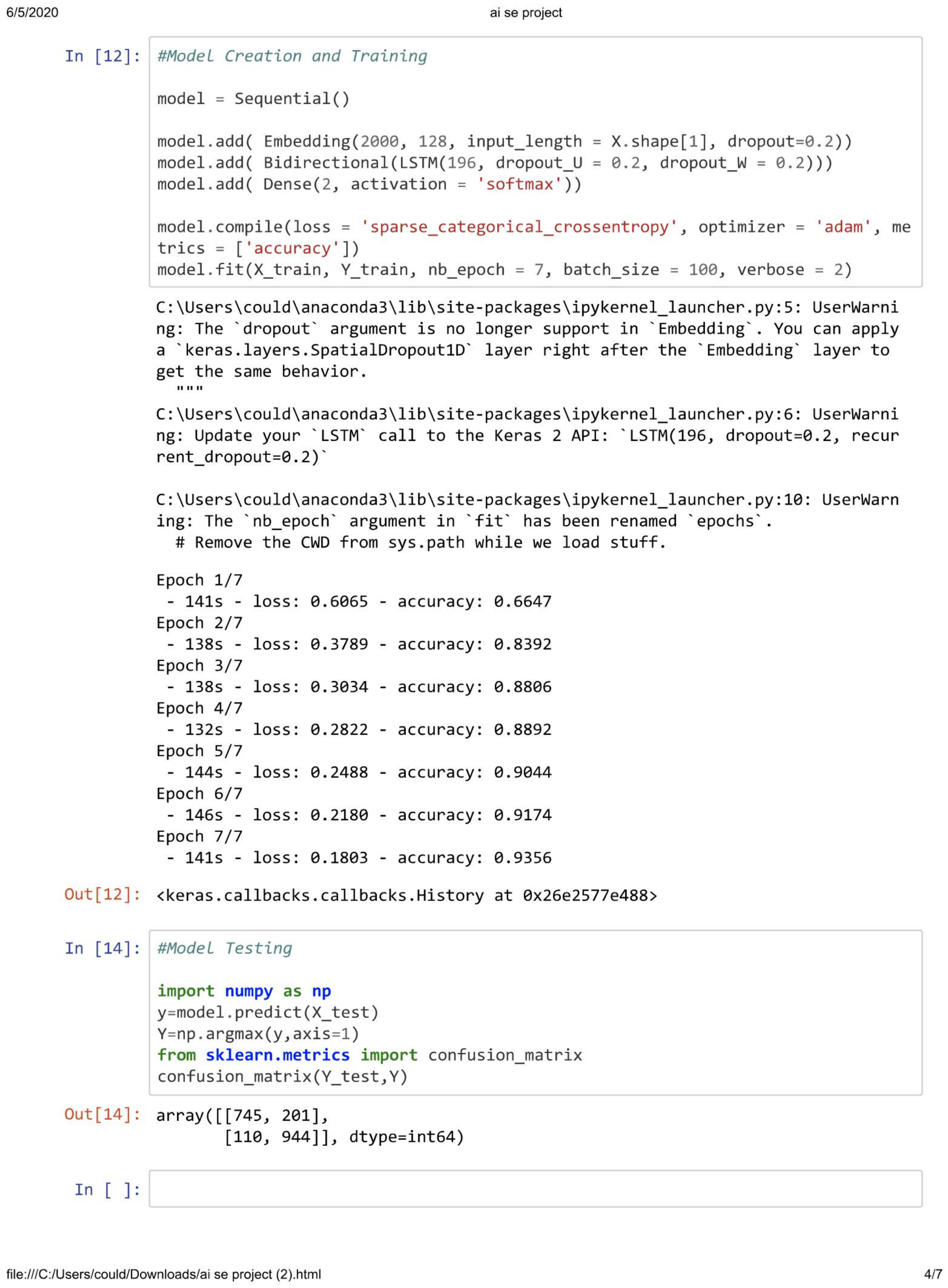
```
In [7]:  Y=dataset.iloc[:,0].values

In [8]:  from sklearn.model_selection import train_test_split
         X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=
         0)

In [9]:  from sklearn.naive_bayes import GaussianNB
         classifier=GaussianNB()
         classifier.fit(X_train,Y_train)

Out[9]:  GaussianNB(priors=None, var_smoothing=1e-09)

In [10]: Y_pred=classifier.predict(X_test)

In [11]: from sklearn.metrics import confusion_matrix
         cm= confusion_matrix(Y_test,Y_pred)
         cm

Out[11]: array([[665, 276],
                [189, 870]], dtype=int64)

In [12]: dataset1=pd.read_csv('products2.csv')

In [14]: X_new=cv.transform(corpus2)
         X_new=X_new.toarray()
         Y_new=classifier.predict(X_new)
         Y_new

Out[14]: array([0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1,
                1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
                0, 1, 1, 0], dtype=int64)
```



```
In [67]: model.score(X_train,Y_train)

Out[67]: 0.805975746968371
```

LSTM Model

In [11]:
```
#Data Pre-Processing

import numpy as np
import pandas as pd
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential, load_model
from keras.layers import Dense, Embedding, LSTM, Bidirectional
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
import re

data = pd.read_csv("datasett.csv",encoding='latin-1')

tokenizer = Tokenizer(num_words=2000, split=' ')

tokenizer.fit_on_texts(data['text'])
X = tokenizer.texts_to_sequences(data['text'])
X = pad_sequences(X)
Y = data['sentiment']


X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, ran
dom_state = 42)
```



6/5/2020 ai se project

In [12]:
```
#Model Creation and Training

model = Sequential()

model.add( Embedding(2000, 128, input_length = X.shape[1], dropout=0.2))
model.add( Bidirectional(LSTM(196, dropout_U = 0.2, dropout_W = 0.2)))
model.add( Dense(2, activation = 'softmax'))

model.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', me
trics = ['accuracy'])
model.fit(X_train, Y_train, nb_epoch = 7, batch_size = 100, verbose = 2)
```

```
C:\Users\could\anaconda3\lib\site-packages\ipykernel_launcher.py:5: UserWarni
ng: The `dropout` argument is no longer support in `Embedding`. You can apply
a `keras.layers.SpatialDropout1D` layer right after the `Embedding` layer to
get the same behavior.
  """
C:\Users\could\anaconda3\lib\site-packages\ipykernel_launcher.py:6: UserWarni
ng: Update your `LSTM` call to the Keras 2 API: `LSTM(196, dropout=0.2, recur
rent_dropout=0.2)`

C:\Users\could\anaconda3\lib\site-packages\ipykernel_launcher.py:10: UserWarn
ing: The `nb_epoch` argument in `fit` has been renamed `epochs`.
  # Remove the CWD from sys.path while we load stuff.

Epoch 1/7
 - 141s - loss: 0.6065 - accuracy: 0.6647
Epoch 2/7
 - 138s - loss: 0.3789 - accuracy: 0.8392
Epoch 3/7
 - 138s - loss: 0.3034 - accuracy: 0.8806
Epoch 4/7
 - 132s - loss: 0.2822 - accuracy: 0.8892
Epoch 5/7
 - 144s - loss: 0.2488 - accuracy: 0.9044
Epoch 6/7
 - 146s - loss: 0.2180 - accuracy: 0.9174
Epoch 7/7
 - 141s - loss: 0.1803 - accuracy: 0.9356
```

Out[12]:
```
<keras.callbacks.callbacks.History at 0x26e2577e488>
```

In [14]:
```
#Model Testing

import numpy as np
y=model.predict(X_test)
Y=np.argmax(y,axis=1)
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test,Y)
```

Out[14]:
```
array([[745, 201],
       [110, 944]], dtype=int64)
```

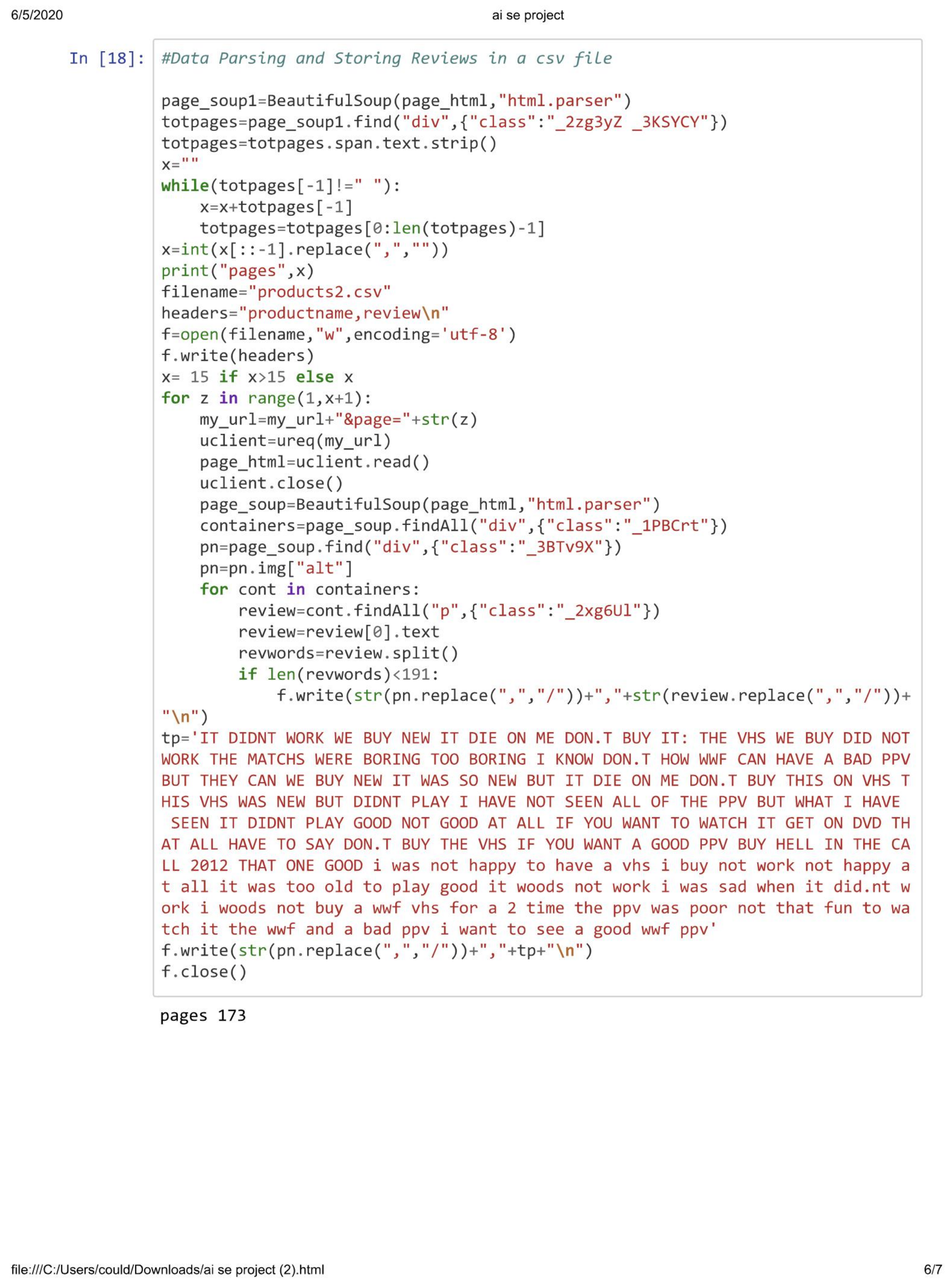
In [ ]:

file:///C:/Users/could/Downloads/ai se project (2).html 4/7

Deployment of Model :

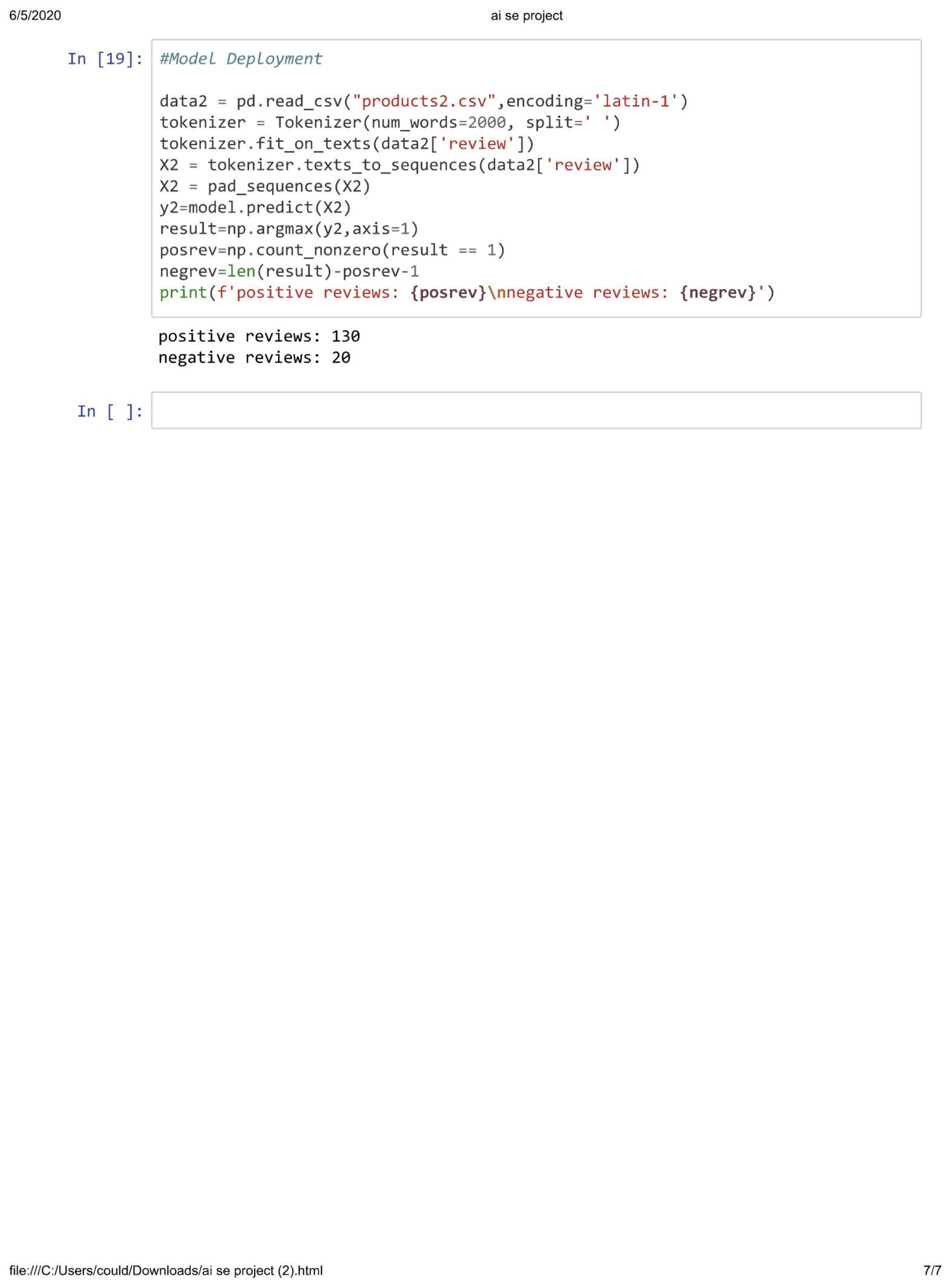
In [15]:
```
#Data Mining

from selenium import webdriver
from bs4 import BeautifulSoup
from urllib.request import urlopen as ureq
my_url=str(input("enter url"))
uclient=ureq(my_url)
page_html=uclient.read()
uclient.close()
```

```
enter urlhttps://www.flipkart.com/samsung-galaxy-s9-midnight-black-64-gb/prod
uct-reviews/itmf33a69rpszgzn?pid=MOBF2VWVBGCT5QQN&lid=LSTMOBF2VWVBGCT5QQN0ZJF
UP&marketplace=FLIPKART
```



In [18]:
```
#Data Parsing and Storing Reviews in a csv file

page_soup1=BeautifulSoup(page_html,"html.parser")
totpages=page_soup1.find("div",{"class":"_2zg3yZ _3KSYCY"})
totpages=totpages.span.text.strip()
x=""
while(totpages[-1]!=" "):
    x=x+totpages[-1]
    totpages=totpages[0:len(totpages)-1]
x=int(x[::-1].replace(",",""))
print("pages",x)
filename="products2.csv"
headers="productname,review\n"
f=open(filename,"w",encoding='utf-8')
f.write(headers)
x= 15 if x>15 else x
for z in range(1,x+1):
    my_url=my_url+"&page="+str(z)
    uclient=ureq(my_url)
    page_html=uclient.read()
    uclient.close()
    page_soup=BeautifulSoup(page_html,"html.parser")
    containers=page_soup.findAll("div",{"class":"_1PBCrt"})
    pn=page_soup.find("div",{"class":"_3BTv9X"})
    pn=pn.img["alt"]
    for cont in containers:
        review=cont.findAll("p",{"class":"_2xg6Ul"})
        review=review[0].text
        revwords=review.split()
        if len(revwords)<191:
            f.write(str(pn.replace(",","/"))+","+str(review.replace(",","/"))+
"\n")
tp='IT DIDNT WORK WE BUY NEW IT DIE ON ME DON.T BUY IT: THE VHS WE BUY DID NOT
WORK THE MATCHS WERE BORING TOO BORING I KNOW DON.T HOW WWF CAN HAVE A BAD PPV
BUT THEY CAN WE BUY NEW IT WAS SO NEW BUT IT DIE ON ME DON.T BUY THIS ON VHS T
HIS VHS WAS NEW BUT DIDNT PLAY I HAVE NOT SEEN ALL OF THE PPV BUT WHAT I HAVE
 SEEN IT DIDNT PLAY GOOD NOT GOOD AT ALL IF YOU WANT TO WATCH IT GET ON DVD TH
AT ALL HAVE TO SAY DON.T BUY THE VHS IF YOU WANT A GOOD PPV BUY HELL IN THE CA
LL 2012 THAT ONE GOOD i was not happy to have a vhs i buy not work not happy a
t all it was too old to play good it woods not work i was sad when it did.nt w
ork i woods not buy a wwf vhs for a 2 time the ppv was poor not that fun to wa
tch it the wwf and a bad ppv i want to see a good wwf ppv'
f.write(str(pn.replace(",","/"))+","+tp+"\n")
f.close()
```

```
pages 173
```



6/5/2020 ai se project

```
In [19]: #Model Deployment

         data2 = pd.read_csv("products2.csv",encoding='latin-1')
         tokenizer = Tokenizer(num_words=2000, split=' ')
         tokenizer.fit_on_texts(data2['review'])
         X2 = tokenizer.texts_to_sequences(data2['review'])
         X2 = pad_sequences(X2)
         y2=model.predict(X2)
         result=np.argmax(y2,axis=1)
         posrev=np.count_nonzero(result == 1)
         negrev=len(result)-posrev-1
         print(f'positive reviews: {posrev}\nnegative reviews: {negrev}')
```

```
positive reviews: 130
negative reviews: 20
```

```
In [ ]:
```

file:///C:/Users/could/Downloads/ai se project (2).html 7/7

TABLES:

TRAINING EPOCH OF THE BI-DIRECTONAL LSTM ALGORITHM:

EPOCH 1/7	0.6647
EPOCH 2/7	0.8392
EPOCH 3/7	0.8806
EPOCH 4/7	0.8892
EPOCH 5/7	0.9044
EPOCH 6/7	0.9174
EPOCH 7/7	0.9356

COMPARISON TABLE OF THE ACCURACY RATE OF BOTH ALGORITHM ON TRAINING:

NAIVE BAYES	BI-DIRECTIONAL LSTM
80.59%	93.56%

COMPARISON TABLE OF THE ACCURACY RATE OF BOTH ALGORITHM ON TESTING:

NAIVE BAYES

	positive review (predicted)	negative review (predicted)
positive review	665	276
negative review	189	870

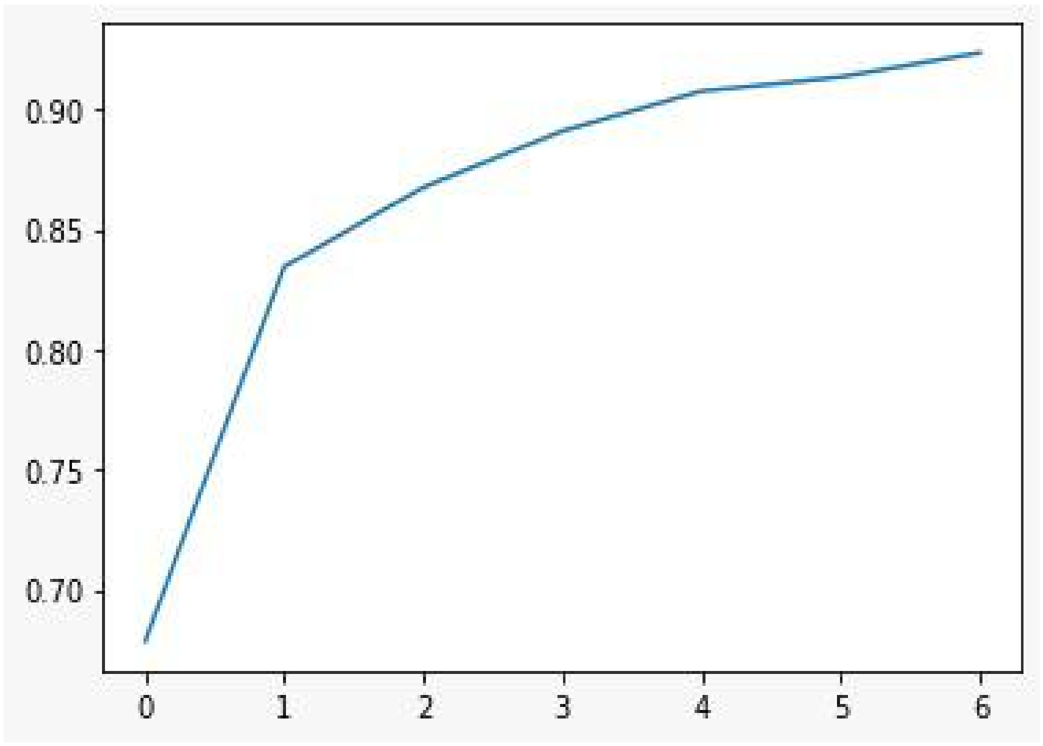
ACCURACY ON TEST SET: 76.75%

BI-DIRECTONAL LSTM

	positive review (predicted)	negative review (predicted)
positive review	745	201
negative review	110	944

ACCURACY ON TEST SET: 84.45%

GRAPH OF Bi-Directional LSTM Accuracy vs Epochs:



CONCLUSION:

The Bi-Directional LSTM model outperforms the Naive Bayes model. The Bi-Directional LSTM model taken into account the sequence of words in the sentences in both directions and also gives attention (more weight) to words that are helpful in classification.Whereas in Naive Bayes model probability is used to classify based on normalized comparison of frequency of a word in a positive review and a negative review.Thus we conclude that Bi-Directional LSTM is much better than Naive Bayes model for the classification of product sentiments.

SUMMARY:

So, summarising the project we can see that after taking out the reviews from flipkart for the samsung galaxy s9 midnight black color we got 173 pages of reviews out of those top 150 reviews were scrapped.The model predicted that 130 were positive and 20 were negative reviews. Also, we were able to see how the bi-directional LSTM is better than the Naive Bayes algorithm.

RESULT:

NUMBER OF REVIEWS SCRAPPED= 150
NUMBERS OF POSITIVE REVIEW = 130
NUMBERS OF NEGATIVE REVIEW = 20

REFERENCES:

- https://numpy.org/doc/
- https://pandas.pydata.org/docs/
- https://scikit-learn.org/stable/_downloads/scikit-learn-docs.pdf
- https://keras.io/documentation/
- https://www.nltk.org
- https://docs.python.org/3/library/urllib.html
- https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- https://docs.python.org/3/howto/unicode.html
- https://www.tensorflow.org/guide
- https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
- https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0