INTERNSHIP REPORT

Summer Internship Report Submitted in partial fulfillment Of the requirement for the undergraduate degree of

Bachelor of Technology

In

Computer Science Engineering

Submitted by

Kosti Gowri Sai Madhurya

121710313029

Department of Computer Science Engineering

GITAM Institute of Technology

GITAM Deemed to be University

Visakhapatnam Campus -530045

Acknowledgement

I wish to thank prof. K. Thammi Reddy, HOD, Dept. of CSE and also our University Management who gave us an opportunity to improve our knowledge base by providing us with smart bridge for us to explore. I would like to thank my professor V Sunilkumar Assistant Professor who encouraged me to work on my own. I want to thank each one of them for their material, and for the time that they devoted for developing the course.

Smart Bridge has contributed so much in designing the course. So I want to thank smart bridge and my instructor Prathyusha .

Thanks to everyone who has contributed either directly or indirectly in designing this course.

Abstract

The course „Artificial Intelligence“ started with the basics of python for the first week to make sure everyone was aware with the language and then they started of with the basics of machine learning as it is a part of artificial intelligence where all the important libraries like pandas, matplotlib, keras etc..

Then we were taught accordingly by slowly teaching us Artificial intelligence as well as to write the code. We then finally were assigned with teams of 5 members where everyone was assigned with particular roles and with the help of our project guide Prathyusha we completed our project.

Artificial intelligence (AI) is wide-ranging branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence. AI is an interdisciplinary science with multiple approaches, but advancements in machine learning and deep learning are creating a paradigm shift in virtually every sector of the tech industry.

Table of contents:

Profile of the company:

SMARTBRIDGE is an edTech organization with a vision to bridge the gap between academia & industry. Our outcome-based experiential learning programs on emerging technologies (Internet of Things, Machine Learning, Data Science, Artificial Intelligence,Robotics) are building skilled entry - level engineers, for the corporate world. SmartBridge is in mission to build technology communities in academia to encourage students towards innovation & entrepreneurship. Since inception, we have trained thousands of students, faculty and working professionals on emerging technologies via technical bootcamps, hackathons, Summer & Winter Internship Program.

TWITTER SENTIMENT ANALYSIS USING DEEP LEARNING

1. INTRODUCTION:

1.1 Overview

The analysis of sentiment on social networks, such as Twitter or Facebook, has become a powerful means of learning about the users‟ opinions and has a wide range of applications. Twitter has been growing in popularity and nowadays, it is used every day by people to express opinions about different topics, such as products, movies, music, politicians, events, social events, among others. Twitter sentiment classification aims to classify the sentiment polarity of a tweet as positive, negative or neutral. In this project, we develop a deep learning system for

Twitter sentiment classification.

1.2 Purpose

The study of public opinion can provide us with valuable information. Sentiment Analysis, also called Opinion Mining, is an useful tool within natural language

processing that allow us to identify, quantify, and study subjective information. Due to the fact that quintillion of bytes of data is produced every day, this technique gives us the possibility to extract attributes of this data such as negative or positive opinion about a subject, also information about which subject is being talked about and what characteristics hold the persons or entities expressing that opinion.

2. Literature Survey:

2.1 Existing problem:

One of the most visited social networking sites by millions of users is twitter where they share their opinion about various domains like politics,

brands, products, celebrities etc. and it become important for some companies and government to know the feelings of the people about their company reputation and actions made by government so it can make their work.

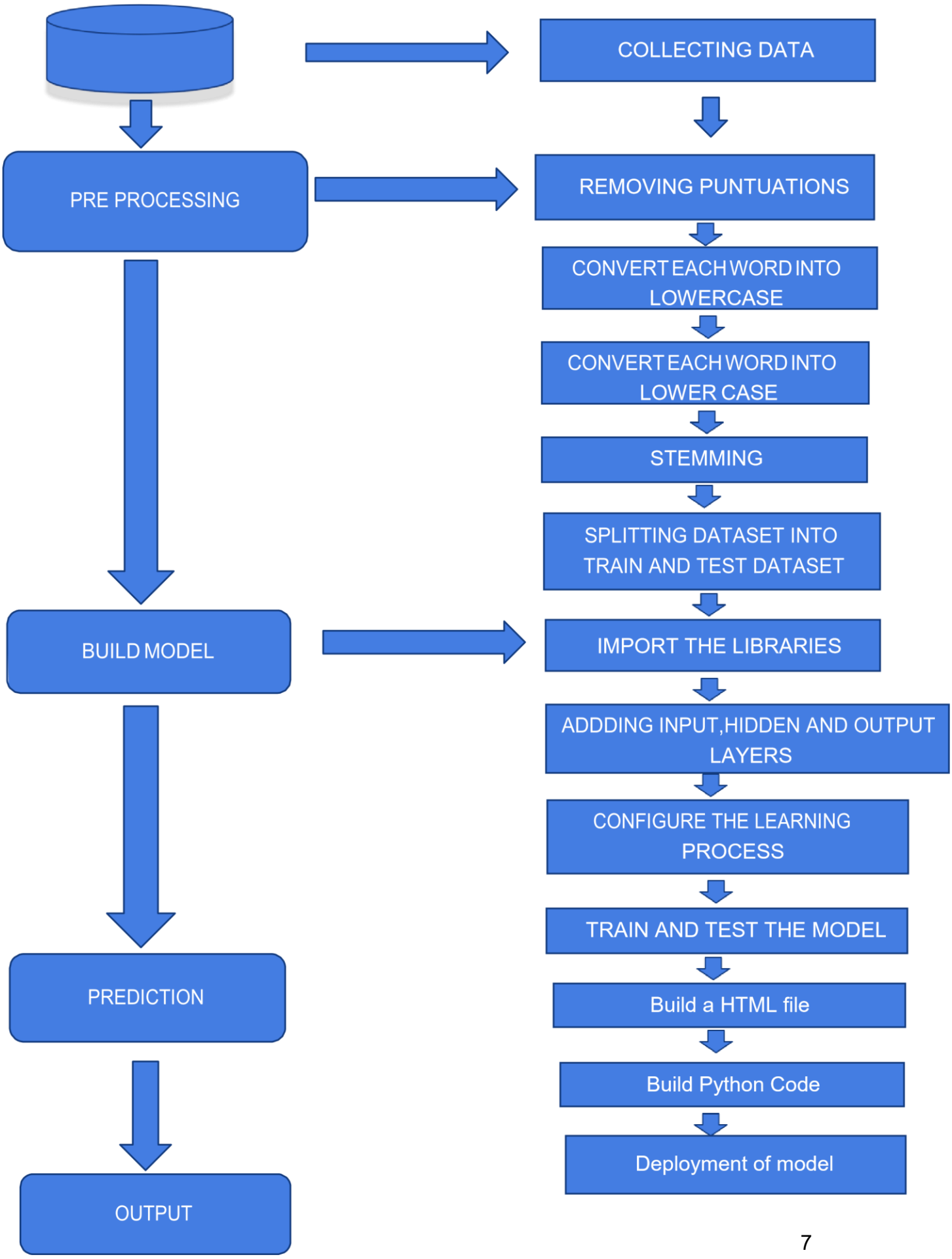
2.2 Proposed solution:

Sentiment analysis is the solution for this Sentiment Analysis is the automated process of

analyzing text data and sorting it into sentiments positive, negative. Performing

Sentiment Analysis on data from Twitter using machine learning can help companies understand how people are talking about their brand. With more than 321 million active users, sending a daily average of 500 million Tweets, Twitter allows businesses to reach a broad audience and connect with customers without intermediaries. On the downside, it‟s harder for

brands to quickly detect negative content, and if it goes viral you might end up with an unexpected PR crisis on your hands. This is one of the reasons why social listening — monitoring conversation and feedback in social media — has become a crucial process in social media marketing.


COLLECTING DATA
PRE PROCESSING
REMOVING PUNTUATIONS
CONVERT EACH WORD INTO LOWERCASE
CONVERT EACH WORD INTO LOWER CASE
STEMMING
SPLITTING DATASET INTO TRAIN AND TEST DATASET
BUILD MODEL
IMPORT THE LIBRARIES
ADDDING INPUT,HIDDEN AND OUTPUT LAYERS
CONFIGURE THE LEARNING PROCESS
TRAIN AND TEST THE MODEL
Build a HTML file
Build Python Code
Deployment of model
PREDICTION
OUTPUT

3. Theoritical Analysis:

3.1 Software Analysis:

There are a high number of frameworks that can be used for machine learning tasks, however, we are going to use Keras because it offers consistent and simple APIs, minimizes the number of user actions required and more importantly, it is easy to learn and use. We will also make use of the Natural Language Toolkit (NLTK), that provides many corpora and lexical resources that will come in handy for tagging, parsing, and semantic reasoning, and Scikit-learn, that provides useful tools for data mining and data analysis.

Text Processing Steps :

1 . Gathering data

2 . Import the Dataset

3 . Text Cleaning or Pre-processing

- •Remove Punctuations, Numbers
- •Convert each word into its lower case
- •Stemming
- •Splitting Data into Training and Test set

4.Model building

1. Gathering data

The data set which we have taken is Twitter Sentiment Analysis. The train set consists of three columns namely ID, Tweet Statement and category.

```
In [2]: dataset= pd.read_csv('train.csv',encoding = "ISO-8859-1",nrows=7000)
        #dataset.drop(["date","day","topic","name"], axis = 1,inplace=True)
```

2. Import the libraries

The dataset train.csv is imported using Pandas Library.Various libraries such as numpy and matplotlib.pyplot

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        import pickle
        #cleaning text data
        import nltk
        import numpy as np
        import re #regular expressions(rmeoving fullstops etc)
        nltk.download('stopwords')
        from nltk.corpus import stopwords
        from nltk.stem.porter import PorterStemmer

        [nltk_data] Downloading package stopwords to
        [nltk_data]     C:\Users\91738\AppData\Roaming\nltk_data...
        [nltk_data]   Package stopwords is already up-to-date!
```

Import the dataset:

```
In [2]: dataset= pd.read_csv('train.csv',encoding = "ISO-8859-1",nrows=9000)
        dataset
```

Out[2]:

	sentiment	text
0	0	is so sad for my APL frie...
1	0	I missed the New Moon trail ...
2	1	omg its already 7:30 :O
3	0	... Omgaga. Im sooo im gunna CRy. I'...
4	0	i think mi bf is cheating on me!!! ...
...	...	...
8995	0	"#WikiLeaks is overloaded by global inter...
8996	0	"< monica> tenk pÃ¬Â¥ Neophos, da &...
8997	0	""Doing some work"" Facebo...
8998	1	""I'm really glad you're in my life....
8999	1	"...begging, begging you ooh ooh ooh...&q...

9000 rows × 2 columns

3. Text Cleaning or Pre-processing:

"Re" is the library which is used to replace the selected special characters with desired parameter. "NLTK" – Natural language Tool Kit is the library used for stemming using a special class in the library.

3.1 Remove Punctuations, Numbers:

Punctuations, Numbers doesn"t help much in processing the given text, so we will be using re library to replace all the punctuations numbers with a space while excluding alphabets. As in the dataset the reviews are present in Tweet column, we are declaring a variable called tweet and assigning the second row of the column to declared variable.

Then using re library we are substituting all the other special characters with a space excluding alphabets.

removing punctuation and number

```
In [5]: def preprocess_tweet(tweet):
            #Preprocess the text in a single tweet
            #arguments: tweet = a single tweet in form of string
            #convert the tweet to lower case
            tweet.lower()
            #convert all urls to sting "URL"
            tweet= re.sub('[^a-zA-Z]',' ',tweet)
            tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)
            #convert all @username to "AT_USER"
            tweet = re.sub('@[^\s]+','AT_USER', tweet)
            #correct all multiple white spaces to a single white space
            tweet = re.sub('[\s]+', ' ', tweet)
            #convert "#topic" to just "topic"
            tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
            return tweet
        dataset['text'] = dataset['text'].apply(preprocess_tweet)
```

3.2 Convert each word into its lower case:

Every word in the taken tweet should be lower cased, because if we have a word in different cases the machine will think both are different words.

3.3 Stemming:

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers.

```
In [6]: c=[]
for i in range(0,7000):
    review1= re.sub('[^a-zA-Z]',' ',dataset['text'].iloc[i])
    #cover it as lower case
    review1=review1.split()#splits words and forms as list
    ps= PorterStemmer()
    review1=[ps.stem(word) for word in review1 if not word in set(stopwords.words('english'))]
    review1=' '.join(review1)
    c.append(review1)
#print(c)
```

3.4 Splitting Data into Training and Test set:

For this, we need class train_test_split from sklearn.cross_validation. Split can be made 70/30 or 80/20 or 85/15 or 75/25, here we choose 80/20 via “test_size”. X is the bag of words; y is 0 or 1 (positive or negative).

```
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=0.3,random_state=0)
pickle.dump(cv.vocabulary_,open("cf.pkl","wb"))
```

3.5 Model Building Importing Libraries:

The first step is to define the functions and classes we intend to use in this. We will use two classes from the Keras library to define our model. **Initializing the model :**

Keras has 2 ways to define a neural network:

- Sequential
- Function API

We will use the Sequential constructor to create a model, which will then have layers added to it using the add() method.

Adding Input layer:

This step is to add a dense layer (input layer) where you will be specifying the number of inputs to the neural network, activation function and weights initializer and number of connection to the hidden layer as the arguments. We use add() method to add dense layer.

Adding Hidden layer:

This step is to add a dense layer (Hidden layer) where you will be specifying the number neurons to the next layer, activation function and weight initializer as the arguments.

Adding an Output Layer :

This step is to add a dense layer (output layer) where you will be specifying the number of classes your dependent variable has, activation function and weight initializer as the arguments.

importing libraries for the model

```
In [9]: from keras.models import Sequential
        from keras.layers import Dense
        model=Sequential()
```

Adding Input layer

```
In [10]: model.add(Dense(input_dim=x.shape[1],init="random_uniform",activation="sigmoid",output_dim=100))
```

Adding hidden layer

```
In [11]: model.add(Dense(init="random_uniform",activation="sigmoid",output_dim=20))
```

Adding output layer

```
In [12]: model.add(Dense(output_dim=1,init='random_uniform',activation='sigmoid'))
```

Configuring the learning process:

Compilation requires 3 arguments: an optimizer, a loss function, and a list of metrics.

configure the learning process

```
In [13]: model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
         #optimizing model

WARNING:tensorflow:From C:\Users\91738\anaconda3\lib\site-packages\tensorflow\python\ops\nn_impl.py:180: add_dispatch_support.<
locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

In [14]:
```
model.fit(x_train,y_train,epochs=50,batch_size=130)
y_pred=model.predict(x_test)
#print(x_test)
y_pred=(y_pred>0.5)
model.save('final1.h5')
```

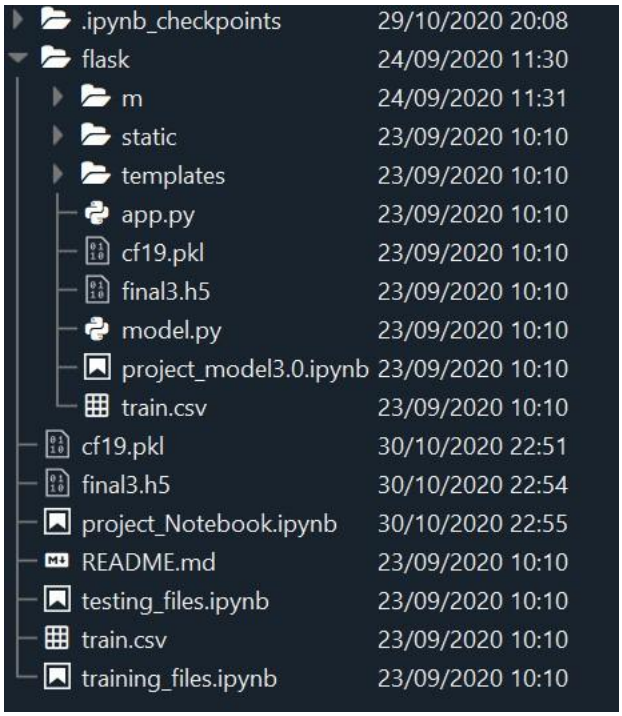
```
WARNING:tensorflow:From C:\Users\91738\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.globa
variables is deprecated. Please use tf.compat.v1.global_variables instead.

Epoch 1/50
4900/4900 [==============================] - 0s 90us/step - loss: 0.6680 - accuracy: 0.6324
Epoch 2/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.6572 - accuracy: 0.6324
Epoch 3/50
4900/4900 [==============================] - 0s 54us/step - loss: 0.6556 - accuracy: 0.6324
Epoch 4/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.6536 - accuracy: 0.6324
Epoch 5/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.6514 - accuracy: 0.6324
Epoch 6/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.6480 - accuracy: 0.6324
Epoch 7/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.6428 - accuracy: 0.6324
Epoch 8/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.6338 - accuracy: 0.6324
Epoch 9/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.6174 - accuracy: 0.6343
Epoch 10/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.5800 - accuracy: 0.6733
Epoch 11/50
4900/4900 [==============================] - 0s 54us/step - loss: 0.5124 - accuracy: 0.7688
Epoch 12/50
4900/4900 [==============================] - 0s 54us/step - loss: 0.4389 - accuracy: 0.8320
Epoch 13/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.3802 - accuracy: 0.8559
Epoch 14/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.3393 - accuracy: 0.8765
Epoch 15/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.3098 - accuracy: 0.8880
Epoch 16/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.2872 - accuracy: 0.8976
Epoch 17/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.2681 - accuracy: 0.9059
Epoch 18/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.2532 - accuracy: 0.9112
Epoch 19/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.2411 - accuracy: 0.9159
Epoch 20/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.2283 - accuracy: 0.9220
Epoch 21/50
```

```
Epoch 22/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.2090 - accuracy: 0.9298
Epoch 23/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.2007 - accuracy: 0.9318
Epoch 24/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1939 - accuracy: 0.9347
Epoch 25/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1870 - accuracy: 0.9361
Epoch 26/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.1803 - accuracy: 0.9386
Epoch 27/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1741 - accuracy: 0.9406
Epoch 28/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.1690 - accuracy: 0.9431
Epoch 29/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1641 - accuracy: 0.9455
Epoch 30/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1601 - accuracy: 0.9471
Epoch 31/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1564 - accuracy: 0.9469
Epoch 32/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1515 - accuracy: 0.9494
Epoch 33/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1479 - accuracy: 0.9502
Epoch 34/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.1445 - accuracy: 0.9529
Epoch 35/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1417 - accuracy: 0.9537
Epoch 36/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1394 - accuracy: 0.9553
Epoch 37/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1357 - accuracy: 0.9557
Epoch 38/50
4900/4900 [==============================] - 0s 50us/step - loss: 0.1332 - accuracy: 0.9573
Epoch 39/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1307 - accuracy: 0.9571
Epoch 40/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1281 - accuracy: 0.9586
Epoch 41/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1264 - accuracy: 0.9594
Epoch 42/50
4900/4900 [==============================] - 0s 57us/step - loss: 0.1241 - accuracy: 0.9594
Epoch 43/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.1224 - accuracy: 0.9616
Epoch 44/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1202 - accuracy: 0.9622
Epoch 45/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1183 - accuracy: 0.9624
Epoch 46/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1173 - accuracy: 0.9633
Epoch 47/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1154 - accuracy: 0.9631
Epoch 48/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1136 - accuracy: 0.9643
Epoch 49/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1122 - accuracy: 0.9637
Epoch 50/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1108 - accuracy: 0.9649
```

Building HTML Page:

This is the basic HTML page for our Project. H1 tag is used to give heading to the project. user has to enter the tweet , so we have to add 1 (one) text input fields in the web page.A button is used to send these values to the model files this functionality will be written in the python file app.py. the model predicts the value and is displayed on the
{{ y_pred }}field and respected emoji will be displaye

FILE STRUCTURE:


.ipynb_checkpoints 29/10/2020 20:08
flask 24/09/2020 11:30
m 24/09/2020 11:31
static 23/09/2020 10:10
templates 23/09/2020 10:10
app.py 23/09/2020 10:10
cf19.pkl 23/09/2020 10:10
final3.h5 23/09/2020 10:10
model.py 23/09/2020 10:10
project_model3.0.ipynb 23/09/2020 10:10
train.csv 23/09/2020 10:10
cf19.pkl 30/10/2020 22:51
final3.h5 30/10/2020 22:54
project_Notebook.ipynb 30/10/2020 22:55
README.md 23/09/2020 10:10
testing_files.ipynb 23/09/2020 10:10
train.csv 23/09/2020 10:10
training_files.ipynb 23/09/2020 10:10

HTML CODE:

```
<html>
<head>
<title> Sentiment analysis on twitter comments</title>
<style>
body
    {
      font-size:20px;
      font-family: latoregular;
      color : black;
      margin: 140px;
      padding:60px;
      background-image:url("https://i.pinimg.com/564x/35/81/e9/3581e9f7e1a8b4fbcfd85028951047b9.jpg");
      background-repeat: no-repeat;
      background-size: cover;
    }
div
{
  border: 5px solid black;
  #background-color: E0E0E0;
  background-color: FFFFCC;
  width: 450px;
  padding: 30px;
  font-family: 'latoregular';
  font-size: 15px;
  margin-top: 50px;
  margin-right: 300px;
  padding-top: 0px;
}
p.ridge
{
  border-style: ridge;
}
</style>
</head>
<body>
<div>
<h2> Sentiment analysis on twitter comments </h2>
<hr>
<form action = "{{ url_for('y_predict')}}" method = "post">
    <br>
    <b> Enter the twitter comment </b>
    <input type = "text" name ="Sentence"/> <br>
    <br>
    <center><button> Predict </button></center>
</form>
<center> <b> </b> </center>
{{ prediction_text }}
</div>
</body>
</html>
```

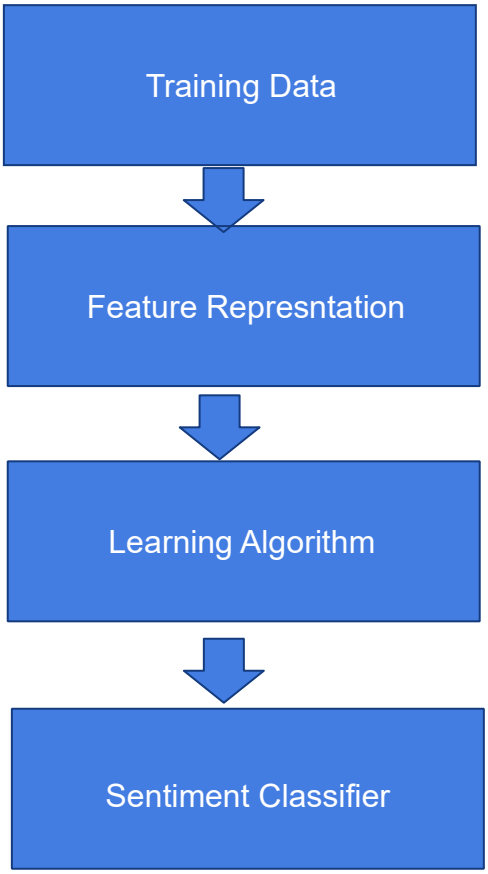
PYTHON CODE:

```
from keras.models import load_model
from flask import Flask,request,jsonify,render_template
from sklearn.feature_extraction.text import CountVectorizer
import pickle
app=Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')
@app.route('/y_predict',methods=['POST'])
def y_predict():
    with open('cf19.pkl','rb') as file:
        cv=pickle.load(file)
        model=load_model('final3.h5')
        prediction = model.predict(cv.transform([request.form['Sentence']]))
        output=prediction[0]
        if(output<0.5):
            return render_template('index.html',prediction_text="The tweet has negative expressions")
        else:
            return render_template('index.html',prediction_text="The tweet has Positive expressions",image=True)
if(__name__=="__main__"):
    app.run(debug=True)
```

4. EXPERIMENAL INVESTIGATION:

During our investigation we got to know all the required parameters to predict the sentiment of the Twitter comments

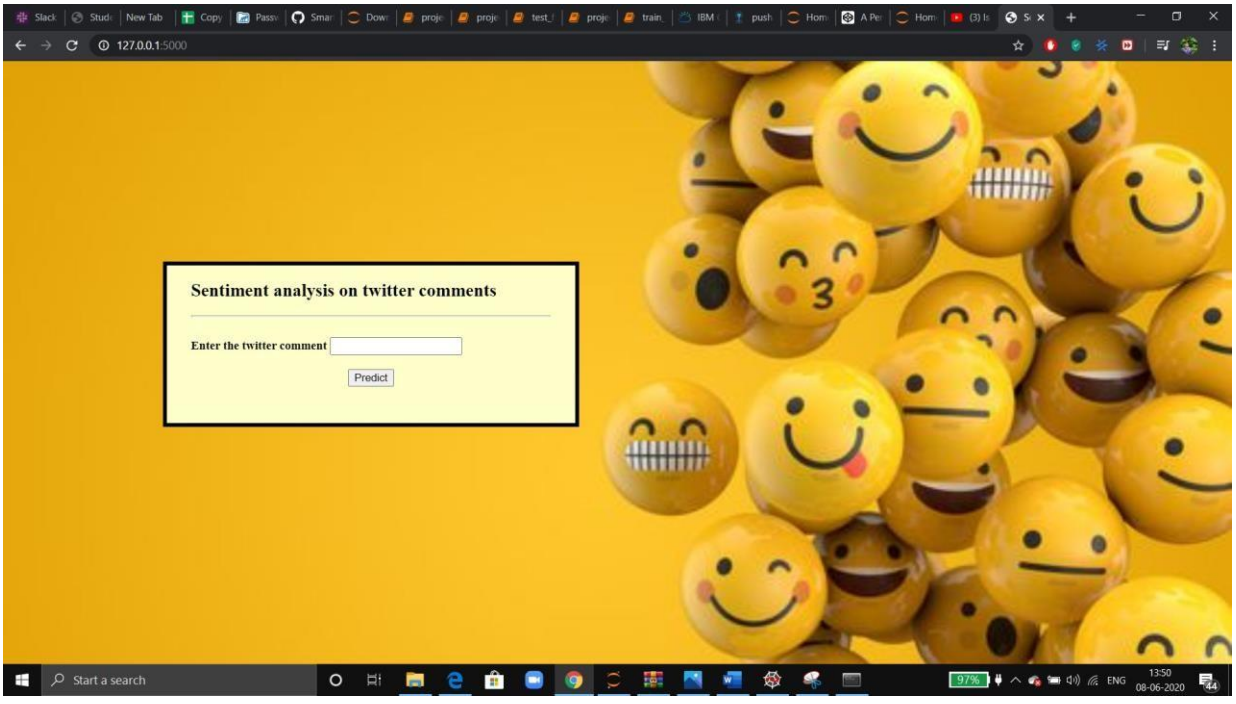
5. FLOWCHART:



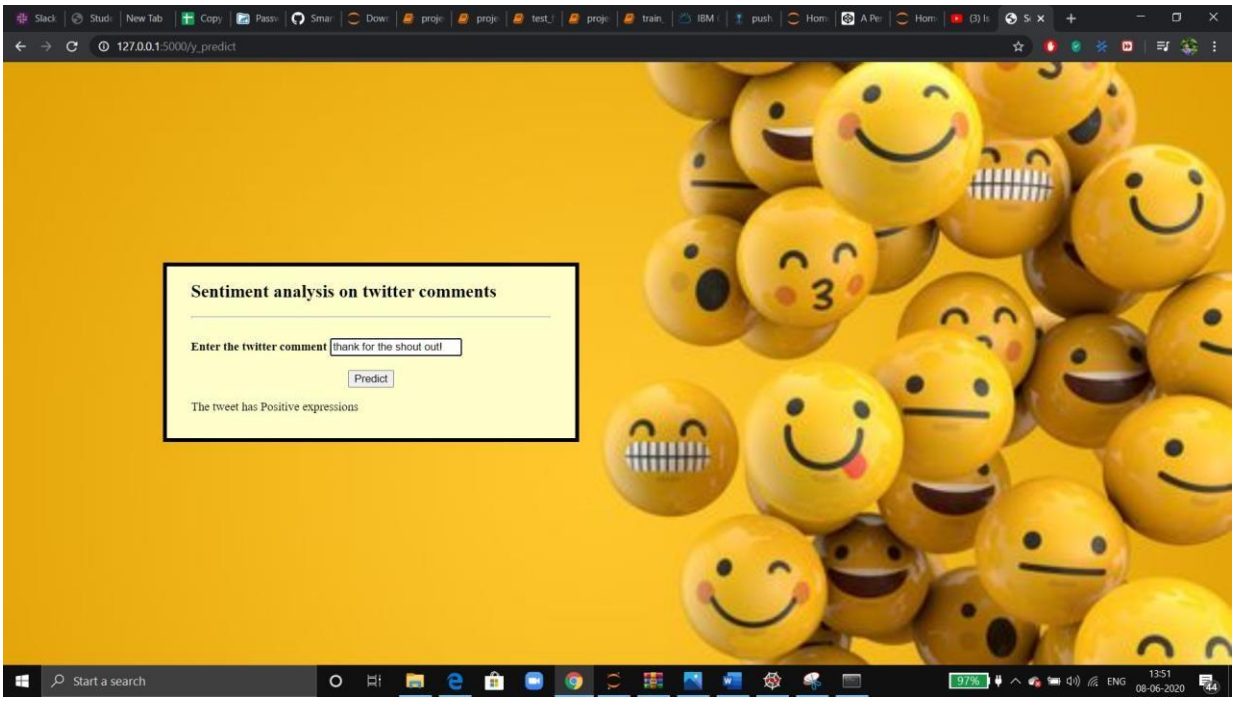
OUTPUT:

```
(base) C:\Users\Mohammed Owais\Desktop\internship>python app1.py
Using TensorFlow backend.
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is depre
cated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is depre
cated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is depre
cated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is depre
cated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is depre
cated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is depre
cated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:541: FutureWarning: Passing (type, 1) or '1type' as a synonym of type i
s deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:542: FutureWarning: Passing (type, 1) or '1type' as a synonym of type i
s deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:543: FutureWarning: Passing (type, 1) or '1type' as a synonym of type i
s deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:544: FutureWarning: Passing (type, 1) or '1type' as a synonym of type i
s deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as a synonym of type i
s deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym of type i
s deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
 * Serving Flask app "app1" (lazy loading)
 * Environment: production
```

PAGE:



OUTPUT:



6. RESULT:

Sentiment Analysis is an interesting way to think about the applicability of Natural Language Processing in making automated conclusions about text. It is being utilized in social media trend analysis and, sometimes, for marketing purposes. The results from sentiment analysis help businesses understand the conversations and discussions taking place about them.They can quickly identify any negative sentiments being expressed and turn poor customer experiences into good ones.

7. ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

Sentiment analysis is a useful tool for any organisation or group for which public sentiment or attitude towards them is important for their success-whichever way that success is defined.

Our social media ,blogs and online forums millions of people are busily discussing and reviewing businesses,companies and organisations .And those opinions are being „listened to“ and analysed.

Those being discussed are making use of this enormous amount data by using computer programs that don“t just locate all mentions of their products,services but also determines the emotions and attitudes behind the words being used.

DISADVANTAGES:

Sentiment analysis tools can identify and analyse many pieces of text automatically and quickly. But computer programs recognizing things –the sort of things a person would have little trouble identifying. And failing to recognize these can skew the results. So sentiment analysis tool do a really great job of analyzing text for opinion and attitude ,but they are not perfect.

8. APPLICATIONS:

Social media monitoring

Customer Experience Management and voice of customer People analytics

9. CONCLUSION:

We develop a deep learning system for message-level Twitter sentiment classification in this project. The effectiveness of this project has been verified in both Positive/negative/neutral classification of tweets. Sentiment analysis allows businesses to identify customer sentiment toward products, brands or services in online Conversations and feedback. Thus this tool can be used in a wide range of applications.

10. FUTURE SCOPE:

Sentiment analysis is already evolving rapidly from a very simple (positive, negative, neutral) to more granular and deep understanding. These new classifier really dimensionalize the nuances of human expression in meaningful ways. There is also a move away from document/record level analysis of the text towards entity/facet level - meaning every expression of opinion is captured so that we can really understand the root cause drivers of opinions. This requires machine learning approaches that are superceding more traditional rules based approaches.

11. Bibliography Appendix :

Model Building:

• dataset

• notebook

Application Building:

• Html 5 and CSS 3 files

• Flask

REFERENCES:

w ww.kaggle.com w ww.github.com

www.geeksforgeeks.org

towardsdatascience.com pythonprogramming.net

Screenshots:

```
In [2]: dataset= pd.read_csv('train.csv',encoding = "ISO-8859-1",nrows=7000)
        #dataset.drop(["date","day","topic","name"], axis = 1,inplace=True)
```

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import pickle
#cleaning text data
import nltk
import numpy as np
import re #regular expressions(rmeoving fullstops etc)
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\91738\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [2]:
```
dataset= pd.read_csv('train.csv',encoding = "ISO-8859-1",nrows=9000)
dataset
```

Out[2]:

	sentiment	text
0	0	is so sad for my APL frie...
1	0	I missed the New Moon trail ...
2	1	omg its already 7:30 :O
3	0	.. Omgaga. Im sooo im gunna CRy. I'...
4	0	i think mi bf is cheating on me!!! ...
...	...	...
8995	0	"#WikiLeaks is overloaded by global inter...
8996	0	"< monica> tenk pÃ¤Ã¥ Neophos, da &...
8997	0	""Doing some work"" Facebo...
8998	1	""I'm really glad you're in my life....
8999	1	"...begging, begging you ooh ooh ooh...&q...

9000 rows × 2 columns

removing punctuation and number

In [5]:
```
def preprocess_tweet(tweet):
    #Preprocess the text in a single tweet
    #arguments: tweet = a single tweet in form of string
    #convert the tweet to lower case
    tweet.lower()
    #convert all urls to sting "URL"
    tweet= re.sub('[^a-zA-Z]',' ',tweet)
    tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)
    #convert all @username to "AT_USER"
    tweet = re.sub('@[^\s]+','AT_USER', tweet)
    #correct all multiple white spaces to a single white space
    tweet = re.sub('[\s]+', ' ', tweet)
    #convert "#topic" to just "topic"
    tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
    return tweet
dataset['text'] = dataset['text'].apply(preprocess_tweet)
```

In [6]:
```
c=[]
for i in range(0,7000):
    review1= re.sub('[^a-zA-Z]',' ',dataset['text'].iloc[i])
    #cover it as lower case
    review1=review1.split()#splits words and forms as list
    ps= PorterStemmer()
    review1=[ps.stem(word) for word in review1 if not word in set(stopwords.words('english'))]
    review1=' '.join(review1)
    c.append(review1)
#print(c)
```

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=0.3,random_state=0)
pickle.dump(cv.vocabulary_,open("cf.pkl","wb"))
```

importing libraries for the model

In [9]:
```
from keras.models import Sequential
from keras.layers import Dense
model=Sequential()
```

Adding Input layer

In [10]:
```
model.add(Dense(input_dim=x.shape[1],init="random_uniform",activation="sigmoid",output_dim=100))
```

Adding hidden layer

In [11]:
```
model.add(Dense(init="random_uniform",activation="sigmoid",output_dim=20))
```

Adding output layer

In [12]:
```
model.add(Dense(output_dim=1,init='random_uniform',activation='sigmoid'))
```

configure the learning process

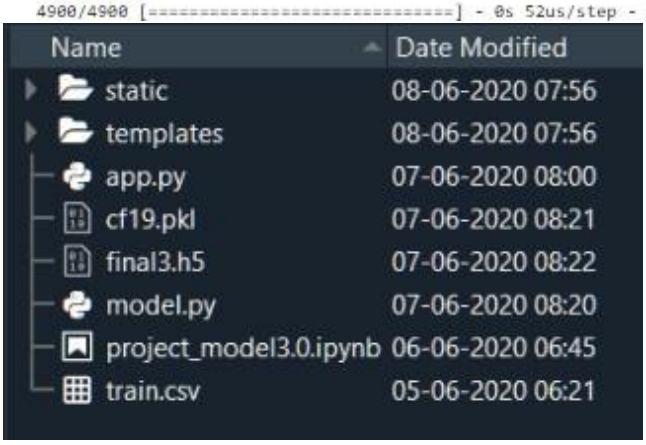
In [13]:
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
#optimizing model
```

```
WARNING:tensorflow:From C:\Users\91738\anaconda3\lib\site-packages\tensorflow\python\ops\nn_impl.py:180: add_dispatch_support.<
locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

In [14]:
```
model.fit(x_train,y_train,epochs=50,batch_size=130)
y_pred=model.predict(x_test)
#print(x_test)
y_pred=(y_pred>0.5)
model.save('final1.h5')
```

```
WARNING:tensorflow:From C:\Users\91738\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.globa
variables is deprecated. Please use tf.compat.v1.global_variables instead.

Epoch 1/50
4900/4900 [==============================] - 0s 90us/step - loss: 0.6680 - accuracy: 0.6324
Epoch 2/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.6572 - accuracy: 0.6324
Epoch 3/50
4900/4900 [==============================] - 0s 54us/step - loss: 0.6556 - accuracy: 0.6324
Epoch 4/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.6536 - accuracy: 0.6324
Epoch 5/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.6514 - accuracy: 0.6324
Epoch 6/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.6480 - accuracy: 0.6324
Epoch 7/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.6428 - accuracy: 0.6324
Epoch 8/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.6338 - accuracy: 0.6324
Epoch 9/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.6174 - accuracy: 0.6343
Epoch 10/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.5800 - accuracy: 0.6733
Epoch 11/50
4900/4900 [==============================] - 0s 54us/step - loss: 0.5124 - accuracy: 0.7688
Epoch 12/50
4900/4900 [==============================] - 0s 54us/step - loss: 0.4389 - accuracy: 0.8320
Epoch 13/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.3802 - accuracy: 0.8559
Epoch 14/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.3393 - accuracy: 0.8765
Epoch 15/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.3098 - accuracy: 0.8880
Epoch 16/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.2872 - accuracy: 0.8976
Epoch 17/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.2681 - accuracy: 0.9059
Epoch 18/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.2532 - accuracy: 0.9112
Epoch 19/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.2411 - accuracy: 0.9159
Epoch 20/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.2283 - accuracy: 0.9220
Epoch 21/50
```

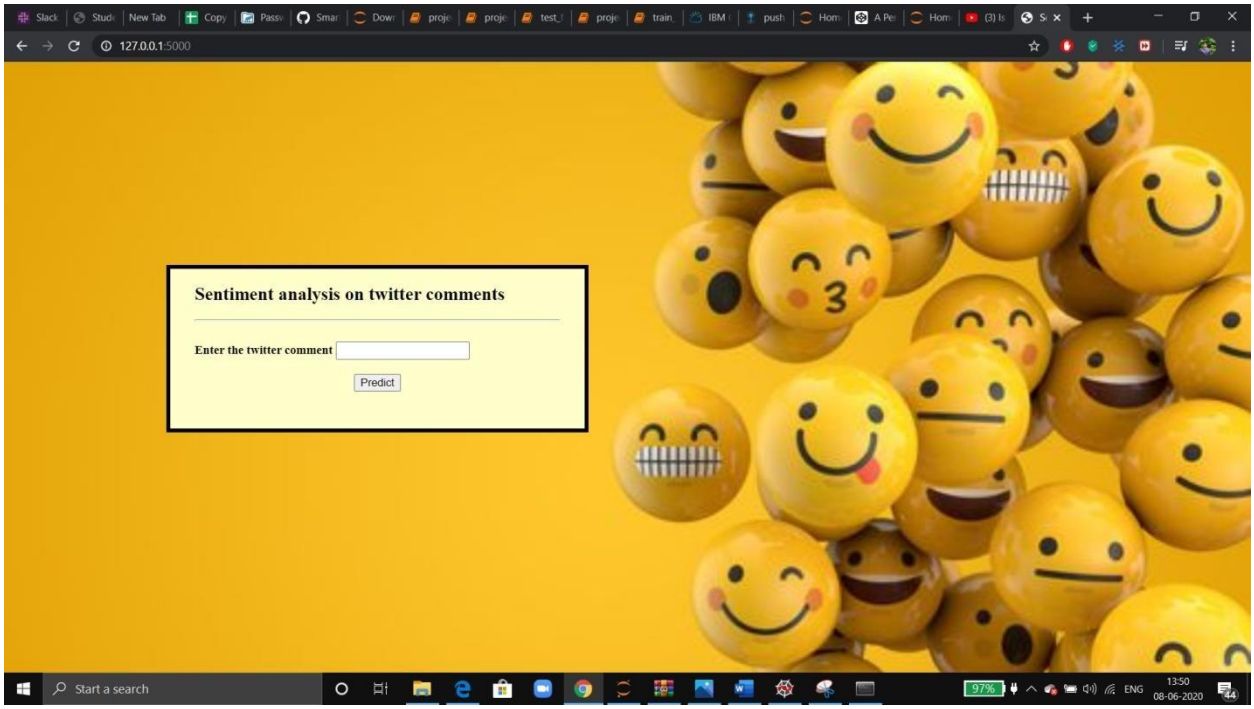
```
Epoch 22/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.2090 - accuracy: 0.9298
Epoch 23/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.2007 - accuracy: 0.9318
Epoch 24/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1939 - accuracy: 0.9347
Epoch 25/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1870 - accuracy: 0.9361
Epoch 26/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.1803 - accuracy: 0.9386
Epoch 27/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1741 - accuracy: 0.9406
Epoch 28/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.1690 - accuracy: 0.9431
Epoch 29/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1641 - accuracy: 0.9455
Epoch 30/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1601 - accuracy: 0.9471
Epoch 31/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1564 - accuracy: 0.9469
Epoch 32/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1515 - accuracy: 0.9494
Epoch 33/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1479 - accuracy: 0.9502
Epoch 34/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.1445 - accuracy: 0.9529
Epoch 35/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1417 - accuracy: 0.9537
Epoch 36/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1394 - accuracy: 0.9553
Epoch 37/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1357 - accuracy: 0.9557
Epoch 38/50
4900/4900 [==============================] - 0s 50us/step - loss: 0.1332 - accuracy: 0.9573
Epoch 39/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1307 - accuracy: 0.9571
Epoch 40/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1281 - accuracy: 0.9586
Epoch 41/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1264 - accuracy: 0.9594
Epoch 42/50
4900/4900 [==============================] - 0s 57us/step - loss: 0.1241 - accuracy: 0.9594
Epoch 43/50
4900/4900 [==============================] - 0s 53us/step - loss: 0.1224 - accuracy: 0.9616
Epoch 44/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1202 - accuracy: 0.9622
Epoch 45/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1183 - accuracy: 0.9624
Epoch 46/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1173 - accuracy: 0.9633
Epoch 47/50
4900/4900 [==============================] - 0s 51us/step - loss: 0.1154 - accuracy: 0.9631
Epoch 48/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1136 - accuracy: 0.9643
Epoch 49/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1122 - accuracy: 0.9637
Epoch 50/50
4900/4900 [==============================] - 0s 52us/step - loss: 0.1108 - accuracy: 0.9649
```

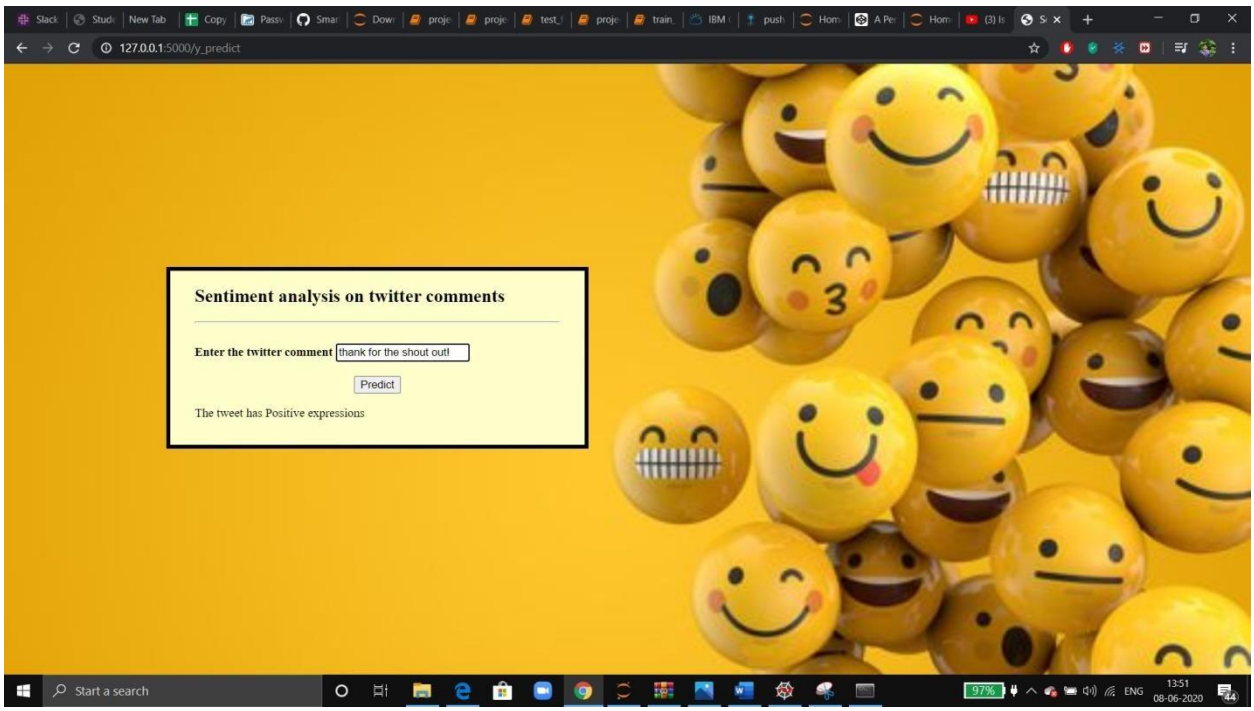


```
<html>
<head>
<title> Sentiment analysis on twitter comments</title>
<style>
body
    {
      font-size:20px;
      font-family: latoregular;
      color : black;
      margin: 140px;
      padding:60px;
      background-image:url("https://i.pinimg.com/564x/35/81/e9/3581e9f7e1a8b4fbcfd85028951047b9.jpg");
      background-repeat: no-repeat;
      background-size: cover;
    }
div
{
  border: 5px solid black;
  #background-color: E0E0E0;
  background-color: FFFFCC;
  width: 450px;
  padding: 30px;
  font-family: 'latoregular';
  font-size: 15px;
  margin-top: 50px;
  margin-right: 300px;
  padding-top: 0px;
}
p.ridge
{
  border-style: ridge;
}
</style>
</head>
<body>
<div>
<h2> Sentiment analysis on twitter comments </h2>
<hr>
<form action = "{{ url_for('y_predict')}}" method = "post">
    <br>
    <b> Enter the twitter comment </b>
    <input type = "text" name ="Sentence"/> <br>
    <br>
    <center><button> Predict </button></center>
</form>
<center> <b> </b> </center>
{{ prediction_text }}
</div>
</body>
</html>
```

```python
from keras.models import load_model
from flask import Flask,request,jsonify,render_template
from sklearn.feature_extraction.text import CountVectorizer
import pickle
app=Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')
@app.route('/y_predict',methods=['POST'])
def y_predict():
    with open('cf19.pkl','rb') as file:
        cv=pickle.load(file)
        model=load_model('final3.h5')
        prediction = model.predict(cv.transform([request.form['Sentence']]))
        output=prediction[0]
        if(output<0.5):
            return render_template('index.html',prediction_text="The tweet has negative expressions")
        else:
            return render_template('index.html',prediction_text="The tweet has Positive expressions",image=True)
if(__name__=="__main__"):
    app.run(debug=True)
```

```
(base) C:\Users\Mohammed Owais\Desktop\internship>python app1.py
Using TensorFlow backend.
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:541: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:542: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:543: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:544: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\Mohammed Owais\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
 * Serving Flask app "app1" (lazy loading)
 * Environment: production
```


127.0.0.1:5000
Sentiment analysis on twitter comments
Enter the twitter comment
Predict


127.0.0.1:5000/y_predict
Sentiment analysis on twitter comments
Enter the twitter comment
thank for the shout out!
Predict
The tweet has Positive expressions

--*-**-*-*-*-*-*-*-*-*-*

THE END