FINAL DOCUMENTATION

SENTIMENTAL ANALYSIS OF TWITTER DATA USING DEEP LEARNING

BY

Naveen Kumar. K

Susitha. A

Varshini. M

Swetha.A

INDEX

TWITTER SENTIMENTAL ANALYSIS USING DEEP LEARNING

INTRODUCTION:

Overview

Sentiment analysis is the automated process of identifying and extracting the subjective information that underlies a text. This can be either an opinion, a judgment, or a feeling about a particular topic or subject. The most common type of sentiment analysis is called 'polarity detection' and involves classifying a statement as 'positive', 'negative', or 'neutral'. Twitter has grown in popularity during the past decades. It is now used by millions of users who share information about their daily life and their feelings. In order to automatically process and analyze these data, applications can rely on analysis methods such as sentiment analysis and topic modeling. Developing a program for sentiment analysis is an approach to be used to computationally measure customers' perceptions.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.

Purpose

Nearly 80% of the world's digital data is unstructured, and a large portion of that includes social media data. Sentiment analysis tools use artificial intelligence and natural language processing (NLP) to organize unstructured text data automatically. Sentiment analysis algorithms are able to learn from data samples to detect the polarity of Tweets in real-time.

- **Business:** In marketing field companies use it to develop their strategies, to understand customers' feelings towards products or brand, how people respond to their campaigns or product launches and why consumers don't buy some products.
- **Politics:** In political field, it is used to keep track of political view, to detect consistency and inconsistency between statements and actions at the government level. It can be used to predict election results as well!
- **Public Actions:** Sentiment analysis also is used to monitor and analyse social phenomena, for the spotting of potentially dangerous situations and determining the general mood of the blogosphere.

LITERATURE SURVEY:

This section summarizes some of the scholarly and the research works in the field of deep learning to analyse sentiments on the Twitter and preparing prediction model for various applications. As the available social platforms are shooting up, the information is becoming vast and can be extracted to turn into business objectives ,social campaigns, marketing and other promotional strategies. The benefit of social media to know public opinions and extract their emotions are considered.

Existing Problem:

Every day massive amount of data is being generated by social media users which can be used to analyze their opinion about any event, movie, product or politics. Thus in order to trace it, an automated process of analysing text data and sorting into positive, negative or neutral is proposed.

Different Classes of Sentiment Analysis

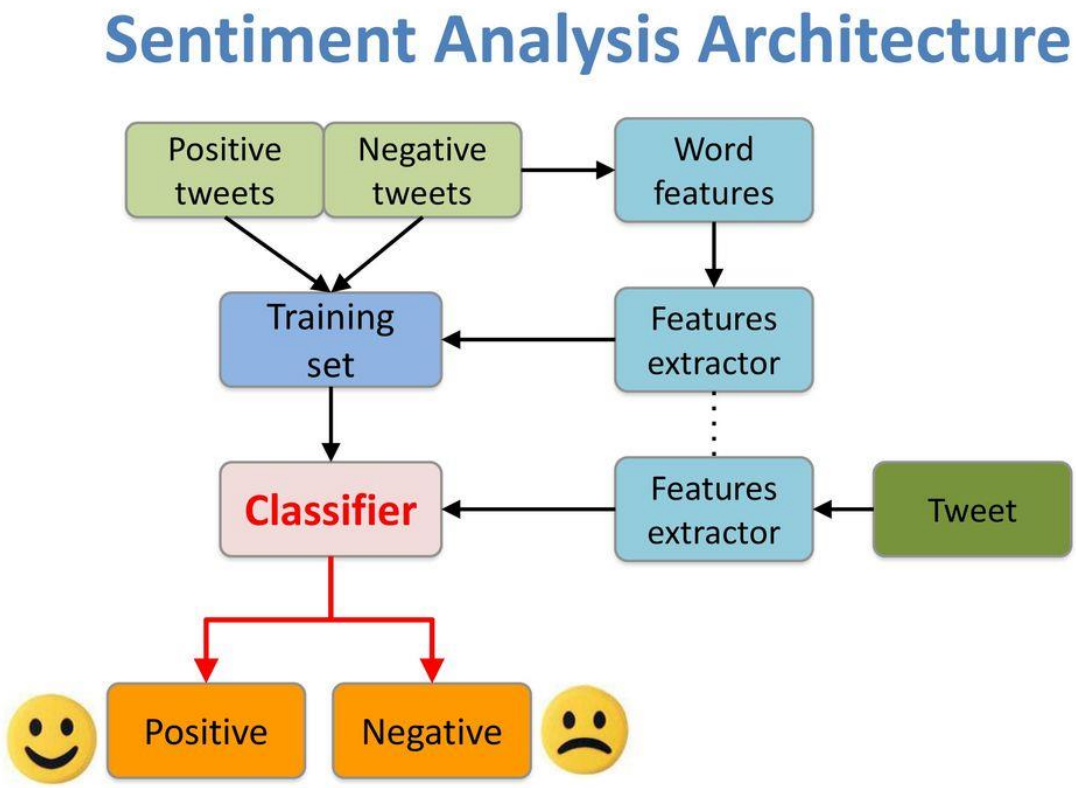
a. Positive Sentiments: These are the good words about the target in consideration. If the positive sentiments are increased, it is referred to be good. In case of product reviews, if the positive reviews about the product are more, it is bought by many customers.

b. Negative Sentiments: These are the bad words about the target in consideration. If the negative sentiments are increased, it is discarded from the preference list. In case of product reviews, if the negative reviews about the product are more, no one intend to buy it.

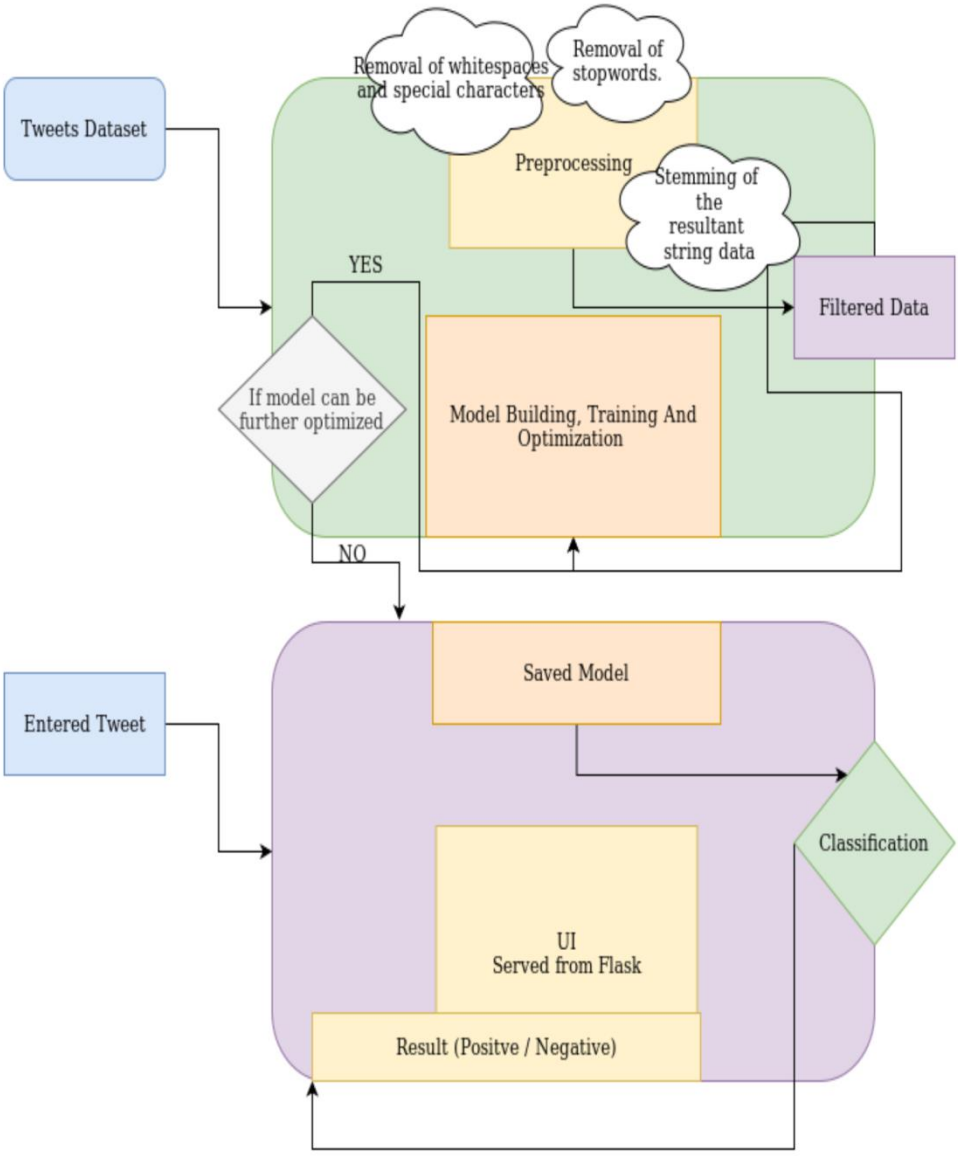
c. Neutral Sentiments: These are neither good nor bad words about the target. Hence it is neither preferred nor neglected.

THEORITICAL ANALYSIS:

Block Diagram:



Flowchart


Tweets Dataset
Removal of whitespaces and special characters
Removal of stopwords.
Preprocessing
Stemming of the resultant string data
Filtered Data
YES
If model can be further optimized
Model Building, Training And Optimization
NO
Saved Model
Entered Tweet
Classification
UI
Served from Flask
Result (Positve / Negative)

EXPERIMENTAL INVESTIGATION:

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

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.

2.Import the libraries

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

Import the dataset

Sentimental Analysis of twitter data using deep learning

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Importing Datasets

In [ ]:
```
train = pd.read_csv('dataset/train.csv', encoding='latin-1')
test = pd.read_csv('dataset/test.csv', encoding='latin-1')
```

In [ ]:
```
train.head()
```

Out[79]:

	ItemID	Sentiment	SentimentText
0	1	0	is so sad for my APL frie...
1	2	0	I missed the New Moon trail...
2	3	1	omg its already 7:30 :O
3	4	0	.. Omgaga. Im sooo im gunna CRy. I'...
4	5	0	i think mi bf is cheating on me!!! ...

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.

In [ ]:
```
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [ ]:
```
sentiment = train["SentimentText"][0]
sentiment = str(re.sub('[^a-zA-Z]', ' ', sentiment)).lower().strip()
```

In [ ]:
```
sentiment
```

Out[86]:
```
'is so sad for my apl friend'
```

In [ ]:
```
ps = PorterStemmer()
sentiment = [ps.stem(word) for word in sentiment if not word in set(stopwords.words('english'))]
```

In [ ]:
```
sentiment
```

Out[88]:
```
[' ', ' ', ' ', 'f', 'r', ' ', ' ', 'p', 'l', ' ', 'f', 'r', 'e', 'n']
```

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.

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.

Stemming

```
In [ ]: sentiment = train["SentimentText"][0]
        sentiment = str(re.sub('[^a-zA-Z]', ' ', sentiment)).lower().strip()

In [ ]: sentiment
Out[86]: 'is so sad for my apl friend'

In [ ]: ps = PorterStemmer()
        sentiment = [ps.stem(word) for word in sentiment if not word in set(stopwords.words('english'))]

In [ ]: sentiment
Out[88]: [' ', ' ', ' ', 'f', 'r', ' ', ' ', 'p', 'l', ' ', 'f', 'r', 'e', 'n']

In [ ]: data = []
        for i in range(train.shape[0]):
          sentiment = train["SentimentText"][i]
          sentiment = str(re.sub('[^a-zA-Z]', ' ', sentiment)).lower().strip().split()
          sentiment = [ps.stem(word) for word in sentiment if not word in set(stopwords.words('english'))]
          sentiment = ' '.join(sentiment)
          data.append(sentiment)
```

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

Importing Count vectorizer:

```
In [ ]: from sklearn.feature_extraction.text import CountVectorizer
        cv = CountVectorizer(max_features = 1000, stop_words='english')
        cv.fit(data)
        X = cv.transform(data).toarray()

In [ ]:

In [ ]: X.shape

Out[93]: (99989, 1000)

In [ ]: cv.vocabulary_

Out[94]: {'sad': 727,
          'friend': 354,
          'miss': 571,
          'new': 597,
          'moon': 580,
          'omg': 620,
          'alreadi': 29,
          'im': 450,
          'sooo': 799,
          'cri': 227,
          'sinc': 776,
          'min': 568,
```

Splitting Data into Training and Test set

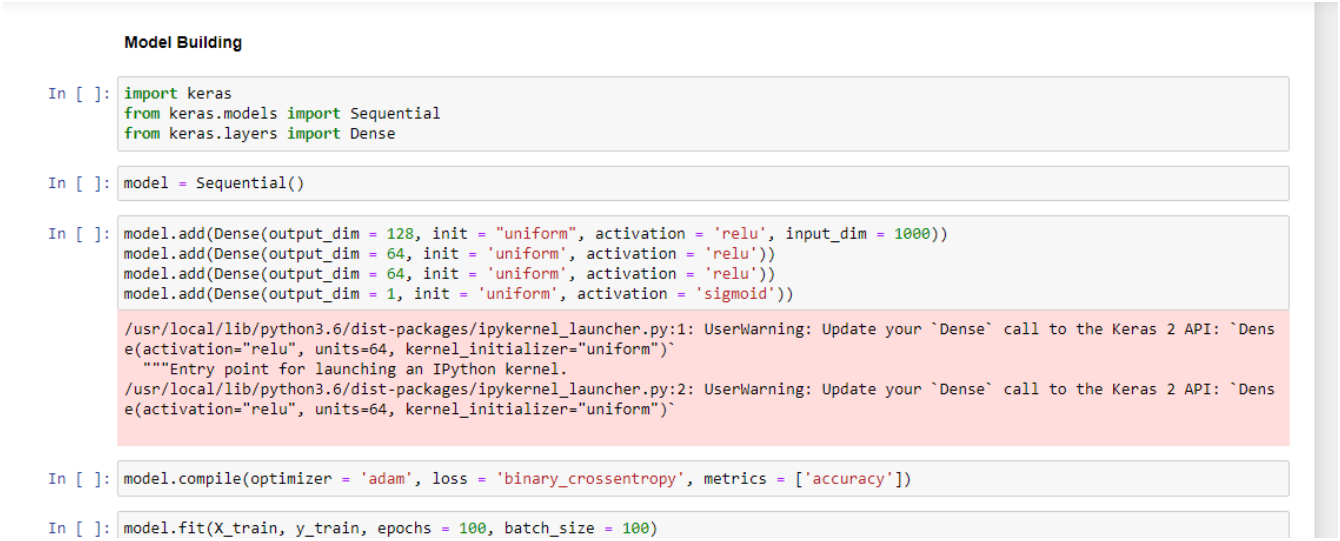
For this, we need class train_test_split from sklearn.cross_validation. Split can be made 70/30 or 80/20 or 85/15 or 75/25, here we choose 80/20 via “test_size”. X is the bag of words; y is 0 or 1 (positive or negative).

```
In [ ]: y.shape

Out[97]: (99989,)

In [ ]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 7)

In [ ]: print("X_train : ", X_train.shape, "\nX_test : ", X_test.shape, "\ny_train : ", y_train.shape, "\ny_test : ", y_test.shape)

        X_train :  (79991, 1000)
        X_test :  (19998, 1000)
        y_train :  (79991,)
        y_test :  (19998,)
```

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.



Model Building

In [ ]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

In [ ]:
```
model = Sequential()
```

In [ ]:
```
model.add(Dense(output_dim = 128, init = "uniform", activation = 'relu', input_dim = 1000))
model.add(Dense(output_dim = 64, init = 'uniform', activation = 'relu'))
model.add(Dense(output_dim = 64, init = 'uniform', activation = 'relu'))
model.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: UserWarning: Update your `Dense` call to the Keras 2 API: `Dens
e(activation="relu", units=64, kernel_initializer="uniform")`
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: UserWarning: Update your `Dense` call to the Keras 2 API: `Dens
e(activation="relu", units=64, kernel_initializer="uniform")`
```

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

In [ ]:
```
model.fit(X_train, y_train, epochs = 100, batch_size = 100)
```

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 layers.

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.

Configuring the learning process:

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

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

In [ ]: model.fit(X_train, y_train, epochs = 100, batch_size = 100)
Epoch 92/100
79991/79991 [==============================] - 5s 57us/step - loss: 0.0945 - accuracy: 0.9482
Epoch 93/100
79991/79991 [==============================] - 4s 56us/step - loss: 0.0943 - accuracy: 0.9483
Epoch 94/100
79991/79991 [==============================] - 5s 56us/step - loss: 0.0951 - accuracy: 0.9478
Epoch 95/100
79991/79991 [==============================] - 5s 56us/step - loss: 0.1023 - accuracy: 0.9457
Epoch 96/100
79991/79991 [==============================] - 4s 55us/step - loss: 0.1007 - accuracy: 0.9463
Epoch 97/100
79991/79991 [==============================] - 4s 55us/step - loss: 0.0985 - accuracy: 0.9465
Epoch 98/100
79991/79991 [==============================] - 4s 56us/step - loss: 0.0960 - accuracy: 0.9476
Epoch 99/100
79991/79991 [==============================] - 5s 60us/step - loss: 0.0943 - accuracy: 0.9480
Epoch 100/100
79991/79991 [==============================] - 5s 57us/step - loss: 0.0940 - accuracy: 0.9481
Out[149]: <keras.callbacks.callbacks.History at 0x7f04a8555828>

In [ ]: import pickle
        pickle.dump(cv, open('cv.pkl', 'wb'))
```

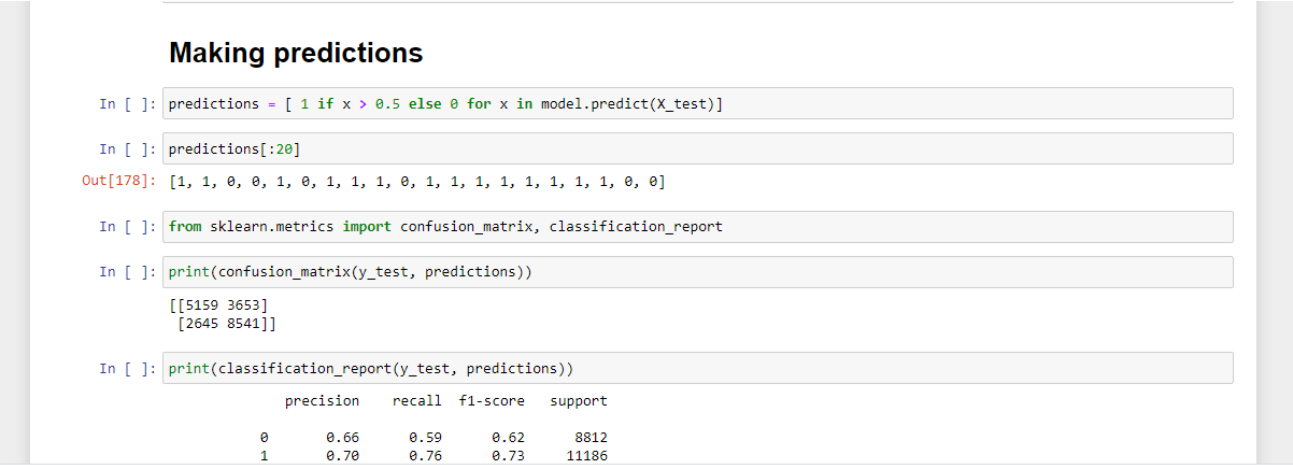
Training the model and saving the model :

Training begins by calling the **fit()** method. The arguments are batch size as you are using "**adam**" (bath gradient descent and epochs: no: of times the model should get trained.

Saving the model

```
In [ ]: import pickle
        pickle.dump(cv, open('cv.pkl', 'wb'))

In [ ]: model.save('model.h5')
```

Making Predictions:



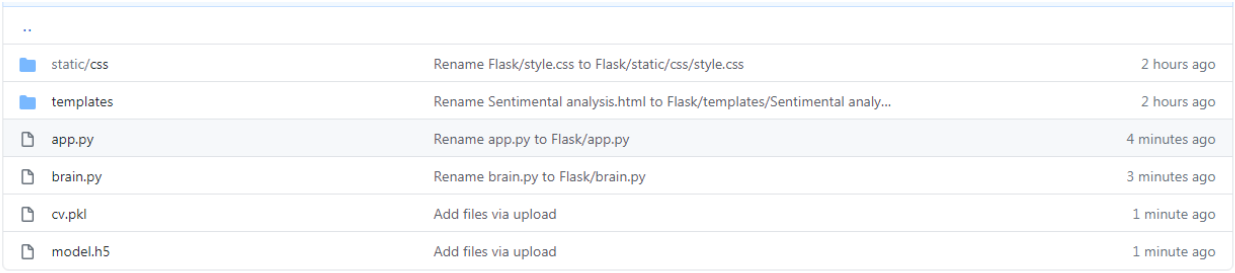
Making predictions

```
In [ ]: predictions = [ 1 if x > 0.5 else 0 for x in model.predict(X_test)]

In [ ]: predictions[:20]

Out[178]: [1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0]

In [ ]: from sklearn.metrics import confusion_matrix, classification_report

In [ ]: print(confusion_matrix(y_test, predictions))

        [[5159 3653]
         [2645 8541]]

In [ ]: print(classification_report(y_test, predictions))

                      precision    recall  f1-score   support

                   0       0.66      0.59      0.62      8812
                   1       0.70      0.76      0.73     11186
```

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 displayed.

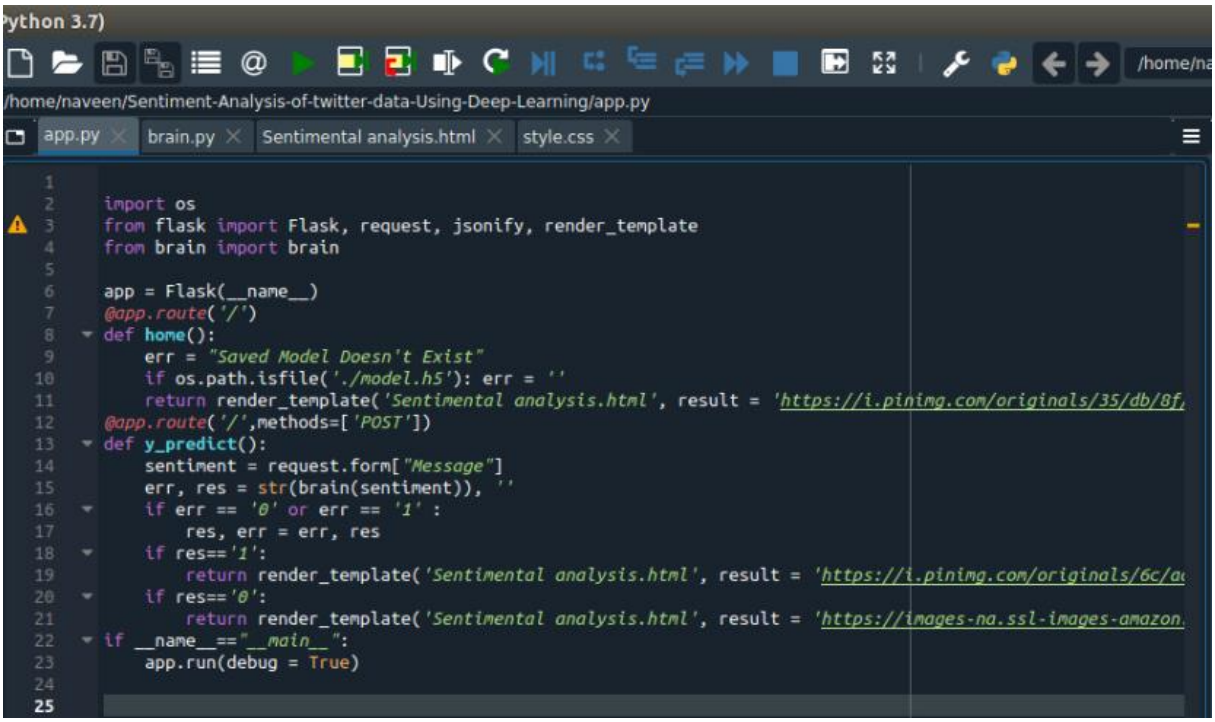
FILE STRUCTURE:



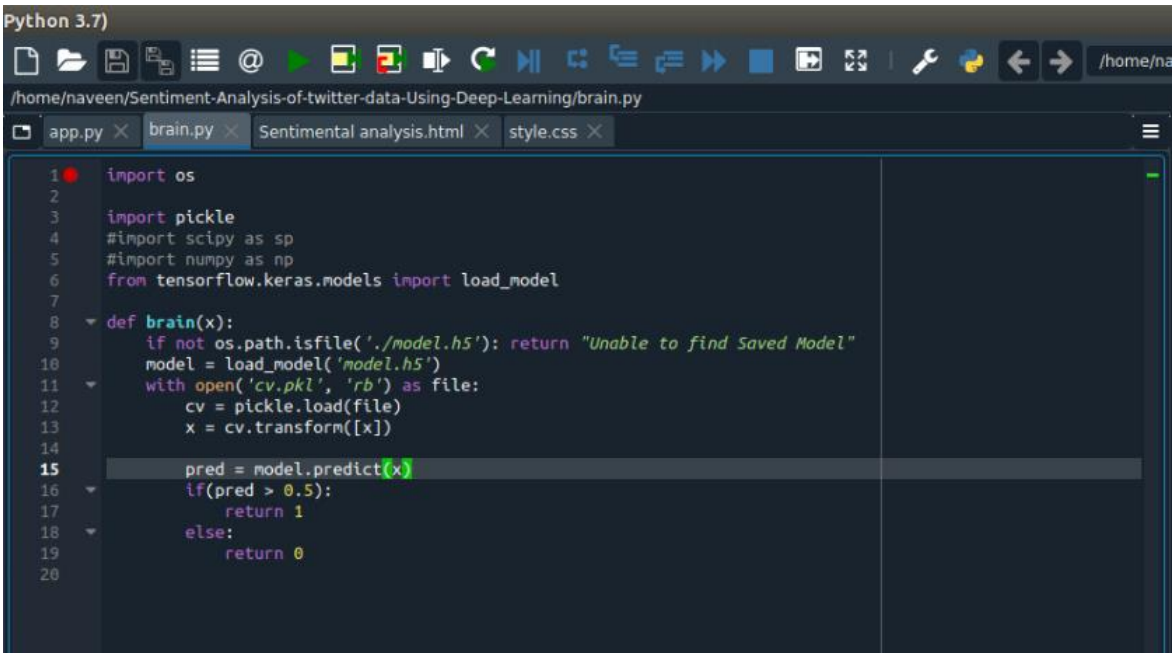
..		
static/css	Rename Flask/style.css to Flask/static/css/style.css	2 hours ago
templates	Rename Sentimental analysis.html to Flask/templates/Sentimental analy...	2 hours ago
app.py	Rename app.py to Flask/app.py	4 minutes ago
brain.py	Rename brain.py to Flask/brain.py	3 minutes ago
cv.pkl	Add files via upload	1 minute ago
model.h5	Add files via upload	1 minute ago

PYTHON CODE:

app.py file:



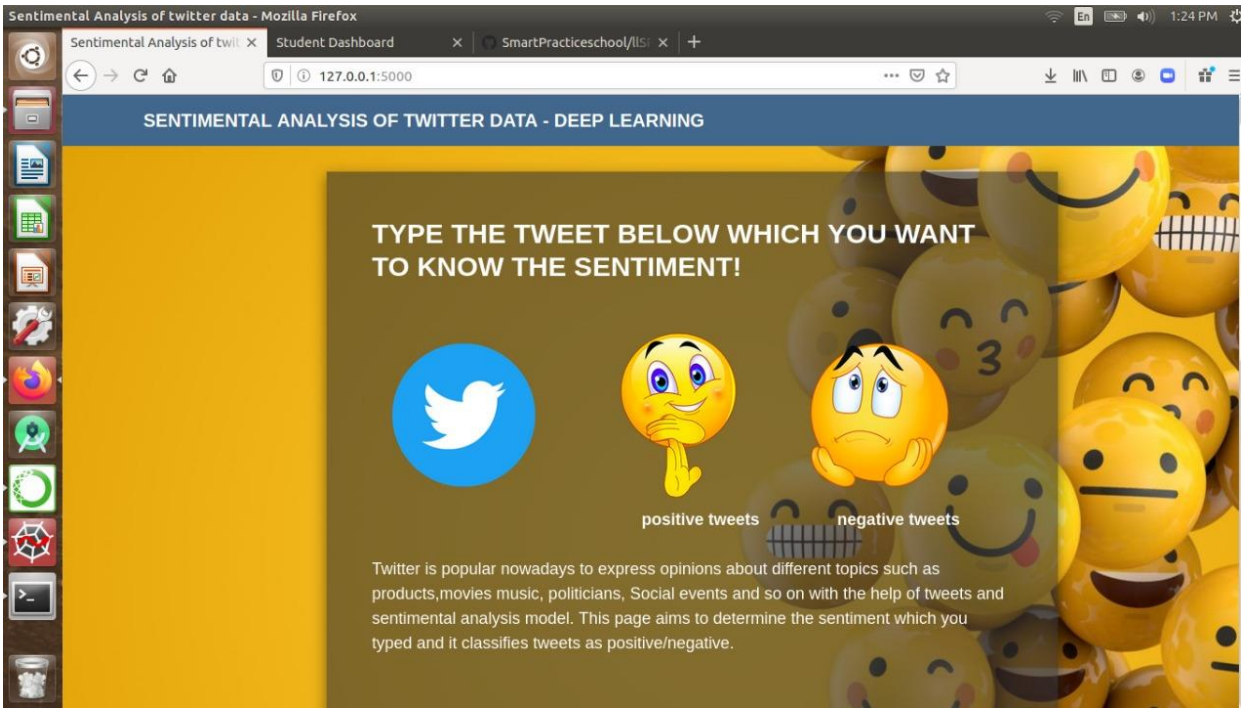
brain.py file:



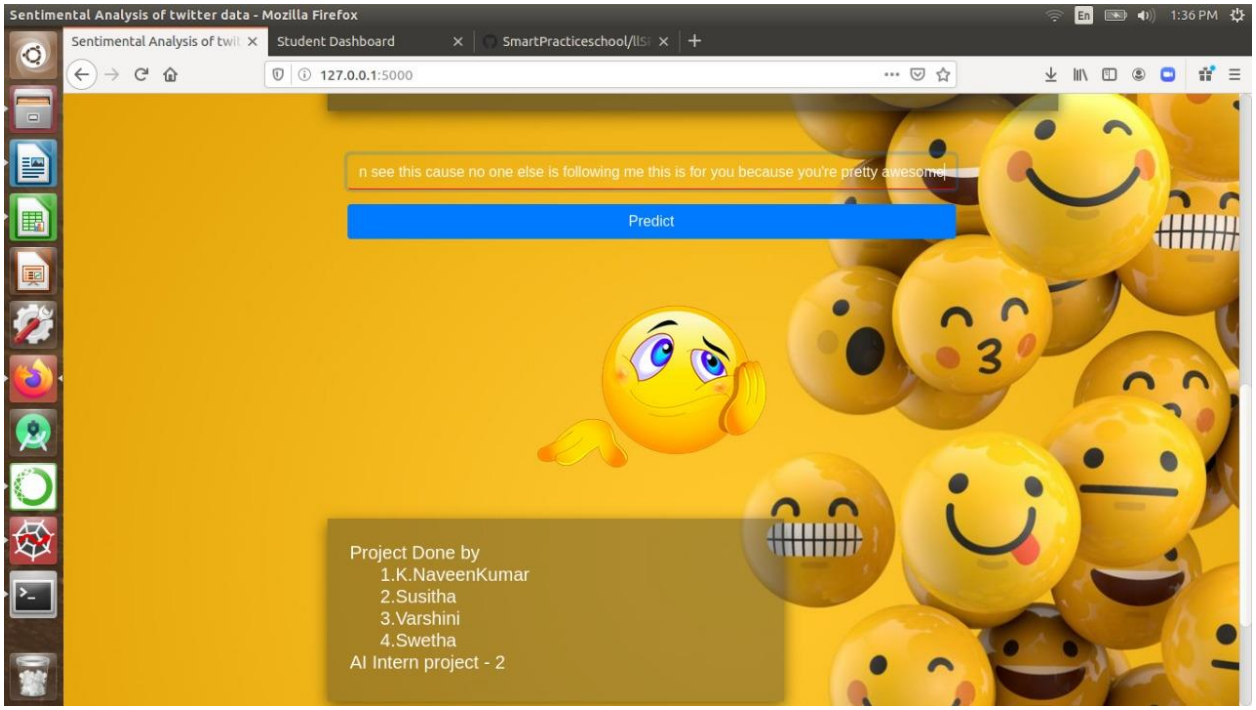
Running flask web app on local server using command line:

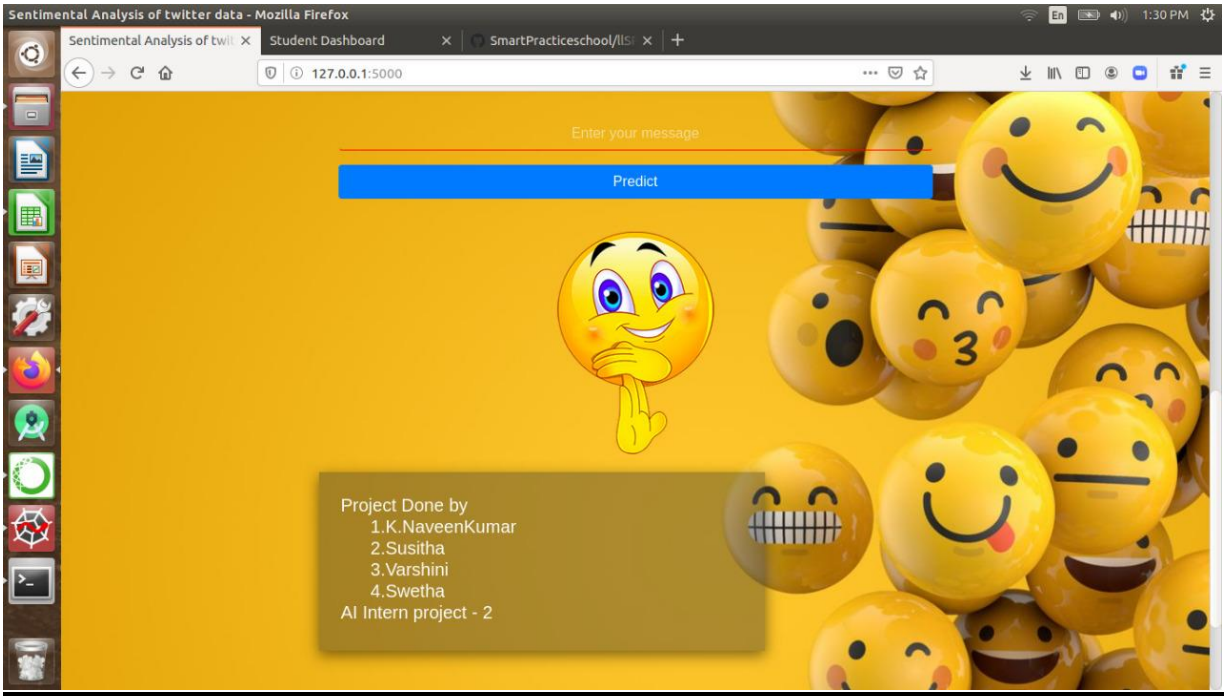
```
@naveen-Lenovo-B460e: ~/Sentiment-Analysis-of-twitter-data-Using-Deep-Learning
(tensorflow) naveen@naveen-Lenovo-B460e:~/Sentiment-Analysis-of-twitter-data-Using-Deep-Learning$ python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 681-126-054
```

WEBPAGE:

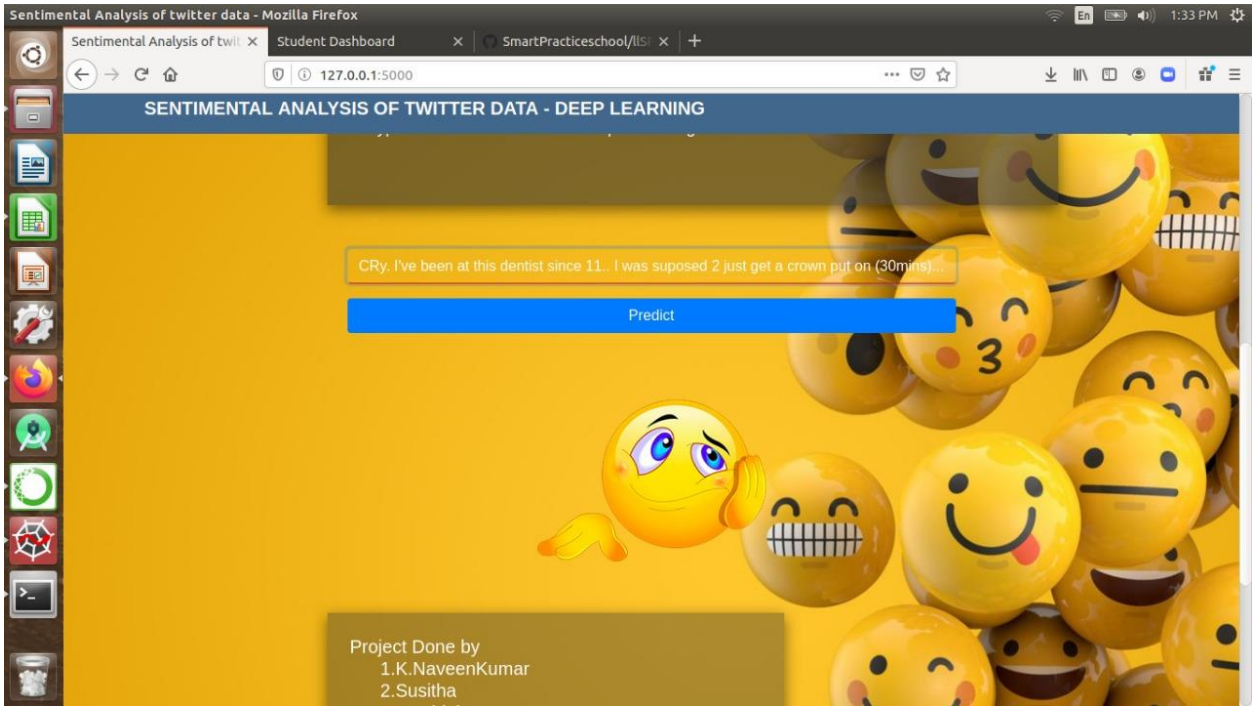


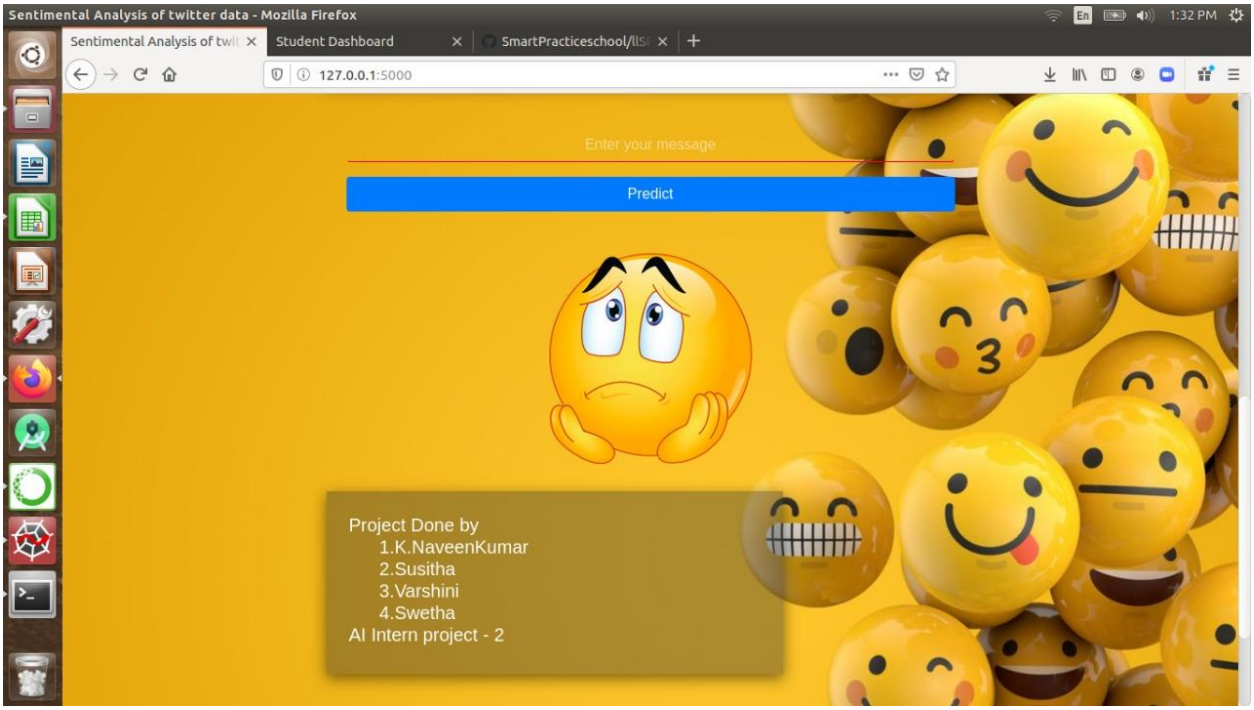
1.Example for Postitive Sentiment:





2.Example for Negative Sentiment:





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.

ADVANTAGES:

- Sentiment analysis is a useful tool for any organization or group for which public sentiment or attitude towards them is important for their success - whichever way that success is defined.
- The results from sentiment analysis help businesses understand the conversations and discussions taking place about them, and helps them react and take action accordingly.
- By listening to and analysing comments on Facebook and Twitter, local government departments can gauge public sentiment towards their department and the services they provide, and use the results to improve services such as parking and leisure facilities, local policing, and the condition of roads.

DISADVANTAGES:

- Misspellings and grammatical mistakes may cause the analysis to overlook important words or usage.
- Sarcasm and irony may be misinterpreted.Analysis is language-specific.
- Discriminating jargon, nomenclature, memes, or turns of phrase may not be recognized.
- 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. So sentiment analysis tool do a really great job of analysing text for opinion and attitude ,but they are not perfect.

APPLICATIONS:

- Social media monitoring
- Customer Experience Management and voice of customer
- People analytics

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.

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.