Project Report
on
Intelligent Customer Helpdesk with Smart Document Understanding
in
Artificial Intelligence
by
Emo-co-ti team.
(emotion-covid-twitter)

1. Introduction

1.1 Overview

The sentiment analysis of Indians after the extension of lockdown announcements to be analyzed with the relevant #tags on twitter and build a predictive analytics model to understand the behavior of people if the lockdow is further extended.
Also develop a dashboard with visualization of people reaction to the govt announcements on lockdown extension.

- **Technical Requirements:** Tweets**,** Artificial Intelligence, ML, Watson AI, Node JS.
- **Project Requirements**: Node-RED, IBM Cloud, IBM Watson, Notebooks, Dashboards.
- **Software Requirements**: Watson Tone Analyzer, Node-RED, Python
- **Project Deliverables**: Sentiment Analysis of COVID-19 Tweets – Visualization Dashboard.
- **Project Team**: Aneri Shah, Janki Patel, Sanjeev Khtwani
- **Project Duration**: 30 days
- **Scope of work**:
    - Fetch Tweets from IEEE dataset using Tweepy API. The hash-tags used are “corona”, "coronavirus", "covid", "pandemic", "lockdown", "quarantine", "hand sanitizer", "ppe", "n95", different possible variants of "sarscov2", "nCov", "covid-19", "ncov2019", "2019ncov", "flatten(ing) the curve", "social distancing", "work(ing) from home" and the language as “english”.
    - Pre-Process the extracted Tweets using IBM Watson Studio Service-Notebooks.
    - Analyze the Data for Sentiments and Tone.
    - Build an interactive dashboard using IBM Watson Studio Service-Dashboards to display the resluts.
    - Build COVID Chatbot using IBM Assistant and IBM Discovery Services.
    - Build a live Sentiment Analyzer for the input tweet given by the user.
    - Integrate the Dashboard, Chatbot and the Analysis using Node-RED and deploy the same on IBM Cloud Platform.

.

2. Literature Survey

2.1 Existing problem

People are not able to predict what would be the situation after lockdown and because of that they can't even take their decisions firmly so we need to overcome that problem for which we provide solution in our dashboard.

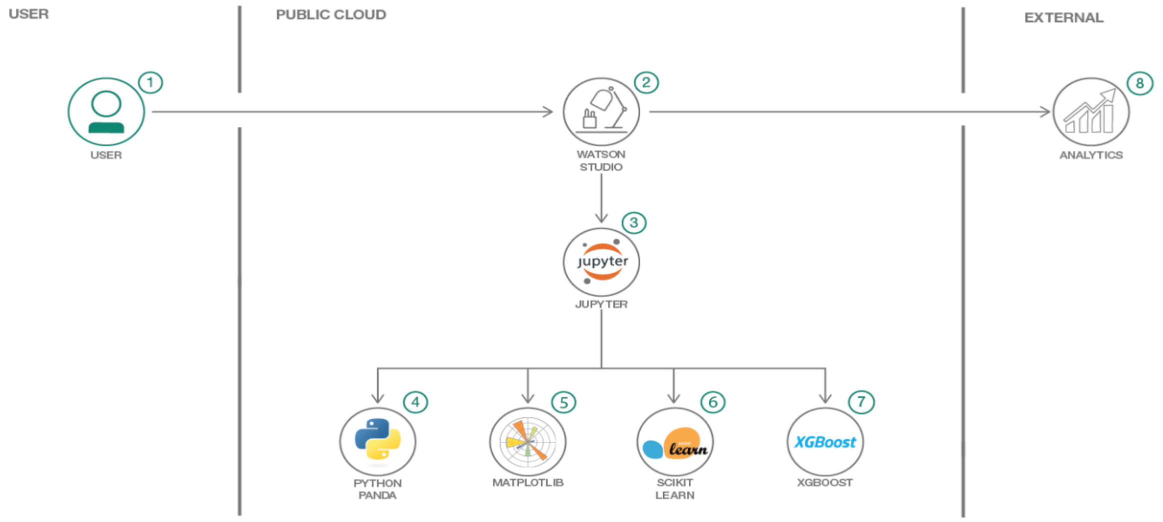
2.2 Proposed Solution

- We provide a detailed solution in dashboard.The facts include in dashboard are as follows:
- Classification of sentiments in positive, negative and neutral countrywise and citywise.
  Same classification of sentiments with tone analyzer is also presented over there.
- A prediction of the sentiment before you post the tweet .

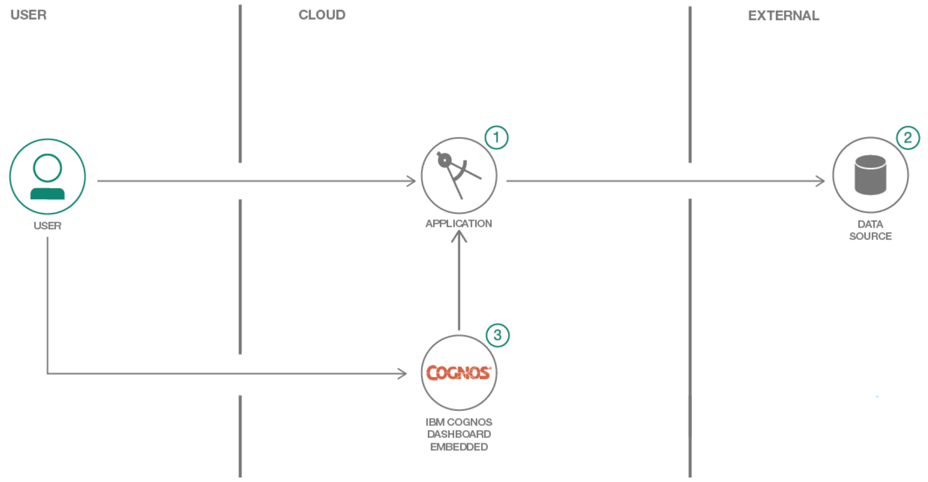
3 Theortical anaysis

Flow diagram

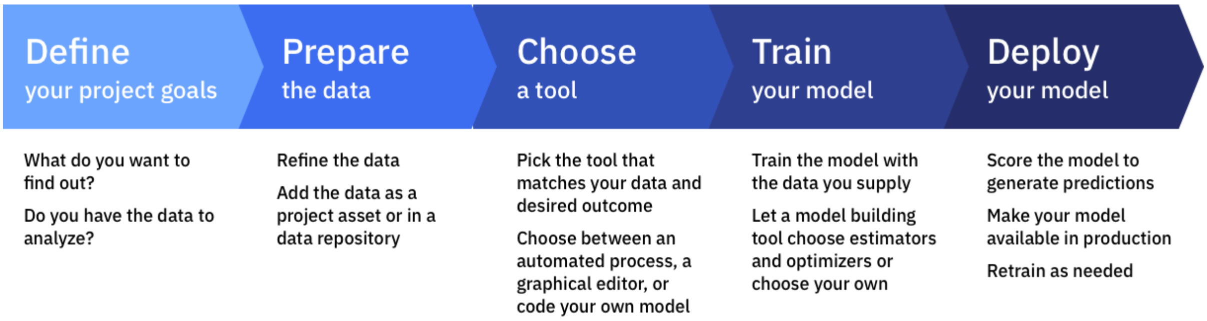
Data preprocess Flow



Cognos flow



ML flow



Define your project goals

What do you want to find out?

Do you have the data to analyze?

Prepare the data

Refine the data

Add the data as a project asset or in a data repository

Choose a tool

Pick the tool that matches your data and desired outcome

Choose between an automated process, a graphical editor, or code your own model

Train your model

Train the model with the data you supply

Let a model building tool choose estimators and optimizers or choose your own

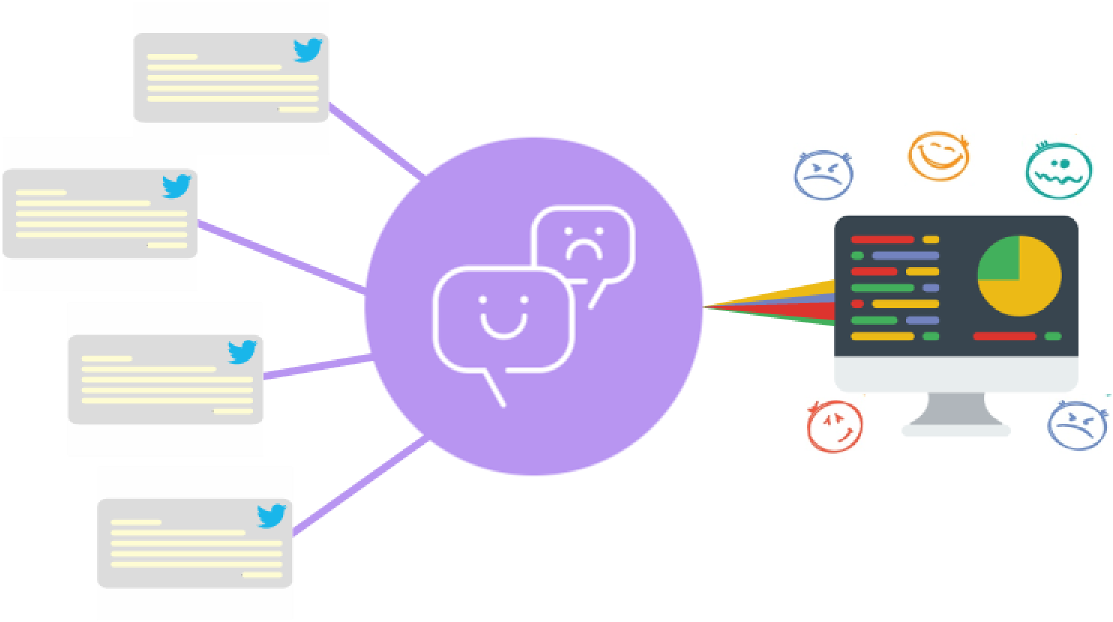
Deploy your model

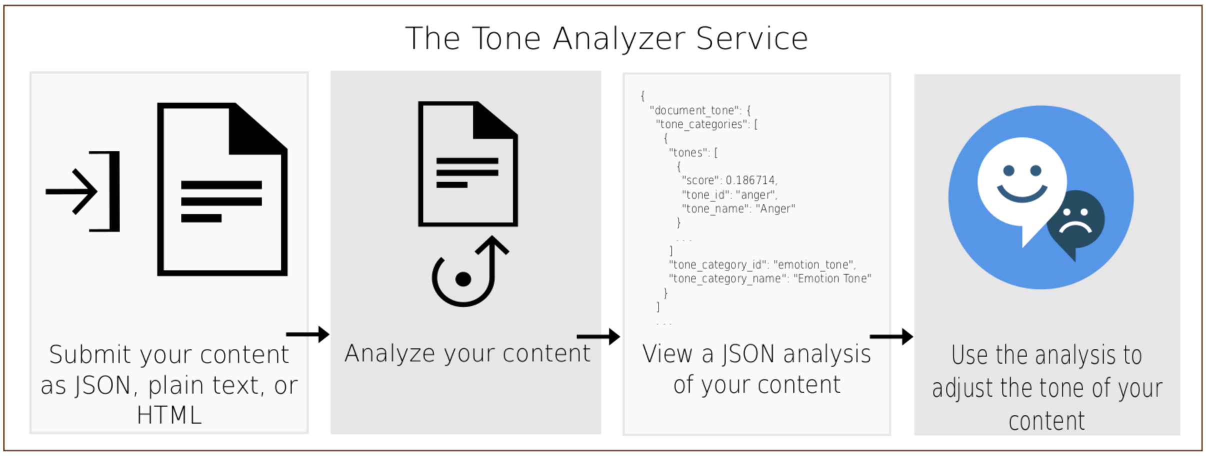
Score the model to generate predictions

Make your model available in production

Retrain as needed

Tone Analyzer flow





The Tone Analyzer Service

Submit your content as JSON, plain text, or HTML →

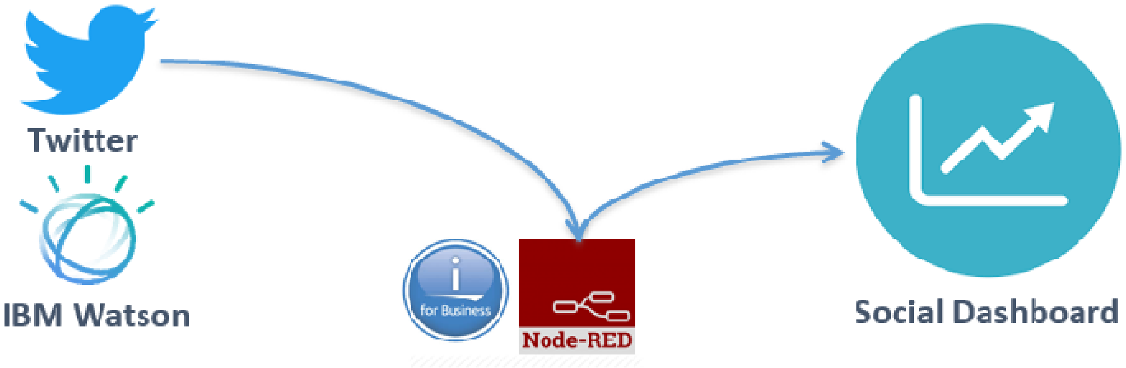
Analyze your content →

```
{
  "document_tone": {
   "tone_categories": [
     {
      "tones": [
        {
         "score": 0.186714,
         "tone_id": "anger",
         "tone_name": "Anger"
        }
        . . .
      ]
      "tone_category_id": "emotion_tone",
      "tone_category_name": "Emotion Tone"
     }
   ]
   . . .
```

View a JSON analysis of your content →

Use the analysis to adjust the tone of your content

Node red basic idea



Hardware/Software design

- Create Watson Services.
- Write code for Extraction and pre processing tweets in Watson studio.
- Write ML code in Watson studio.
- Integrate the results in Cognos Dashboard.
- Perform Sentiment analysis.
- Perform tone analysis by launching tone analyzer service .
- Creating node-red services to integrate with cognos dashboard.

4.Experimental Investigations

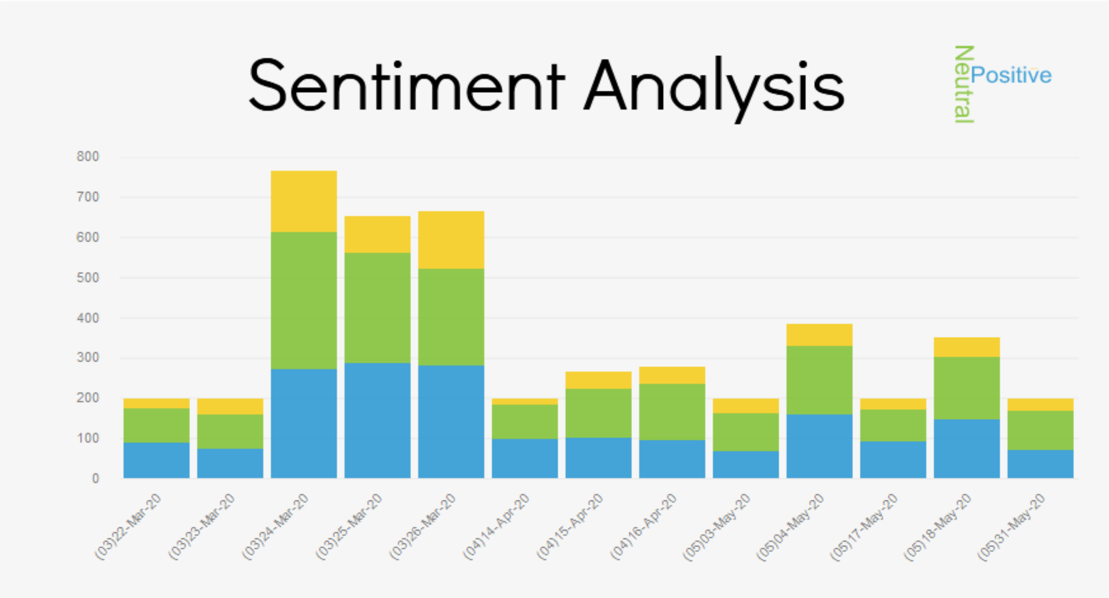
4,569 Tweets

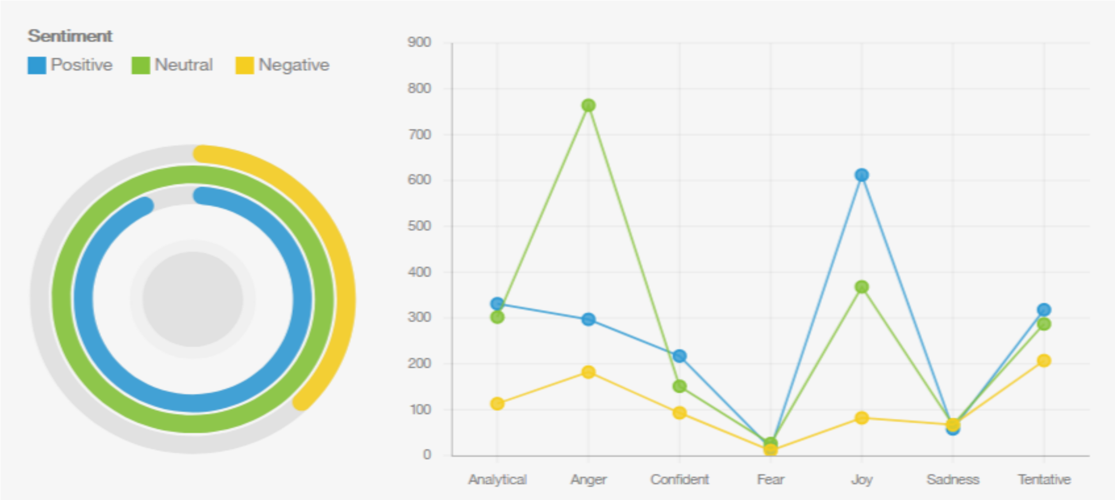
2,115 Location

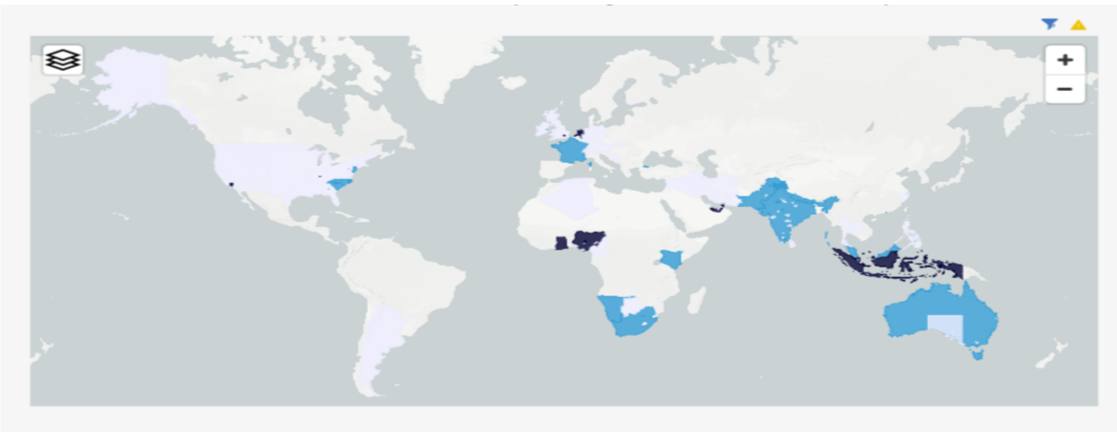
760 Retweets

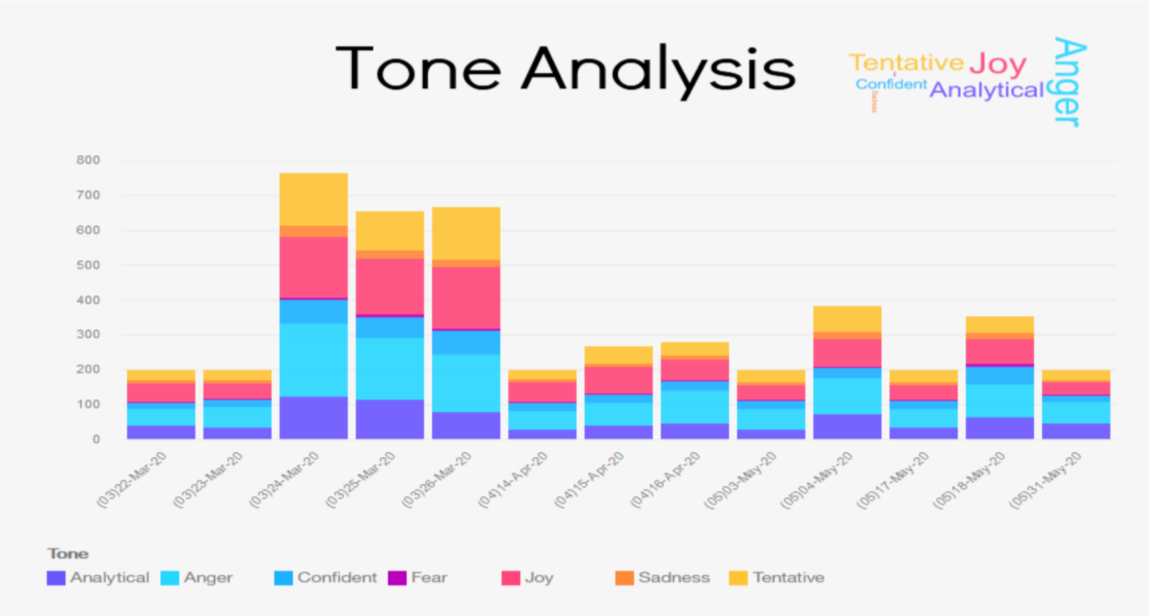
7 Tone

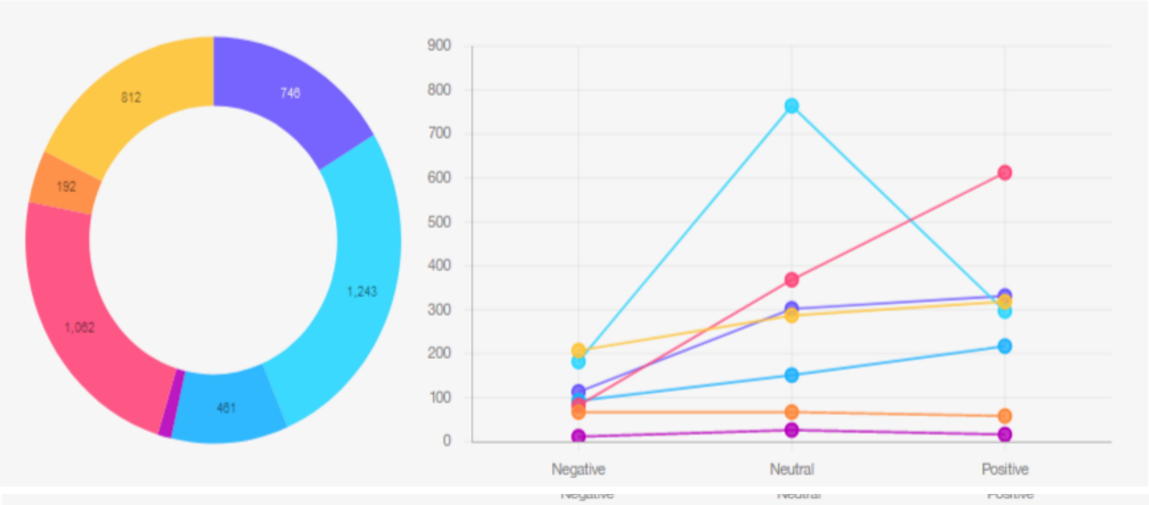
3 Sentiment

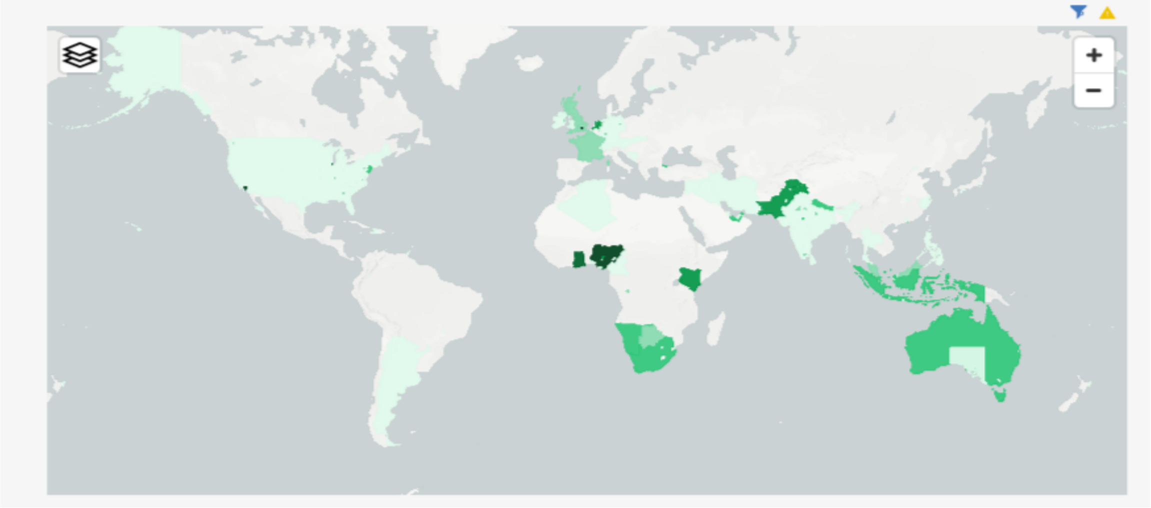


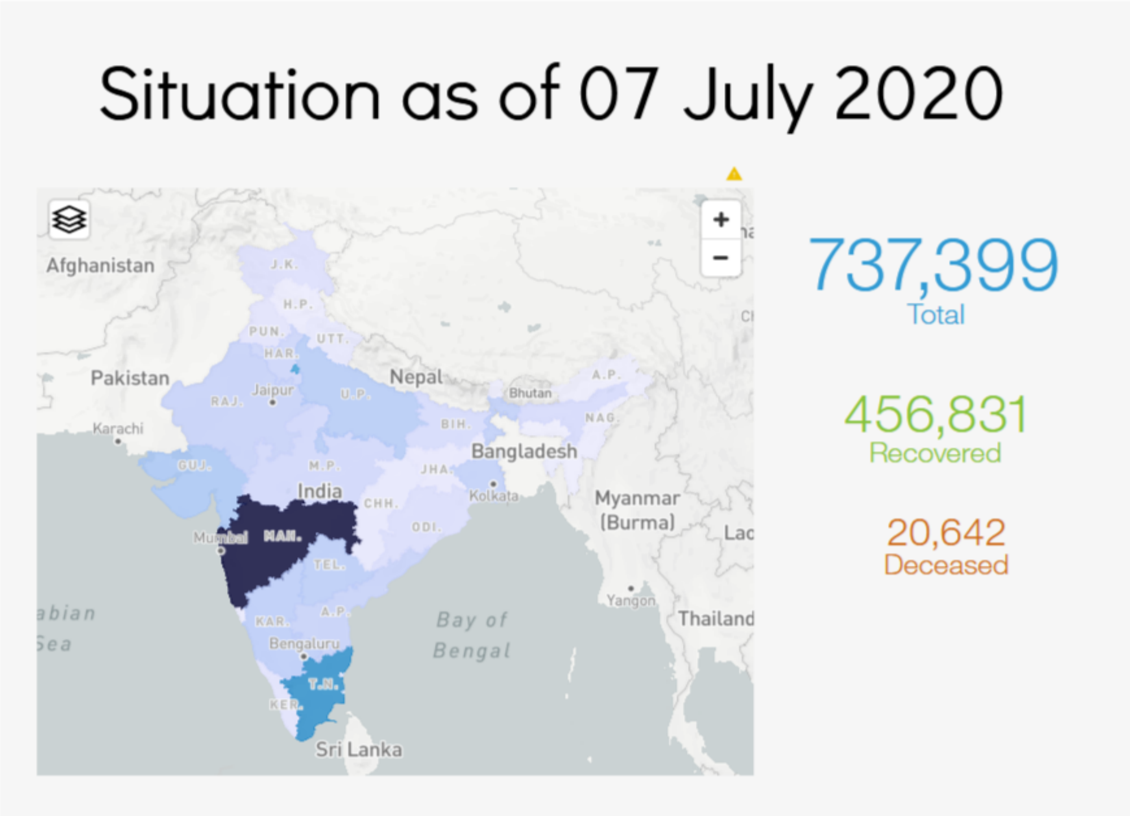





Tone Analysis
Tentative
Joy
Confident
Analytical
Anger
800
700
600
500
400
300
200
100
0
(03)22-Mar-20
(03)23-Mar-20
(03)24-Mar-20
(03)25-Mar-20
(03)26-Mar-20
(04)14-Apr-20
(04)15-Apr-20
(04)16-Apr-20
(05)03-May-20
(05)04-May-20
(05)17-May-20
(05)18-May-20
(05)31-May-20
Tone
Analytical
Anger
Confident
Fear
Joy
Sadness
Tentative


812
746
192
1,243
1,062
461
900
800
700
600
500
400
300
200
100
0
Negative
Neutral
Positive


+
−

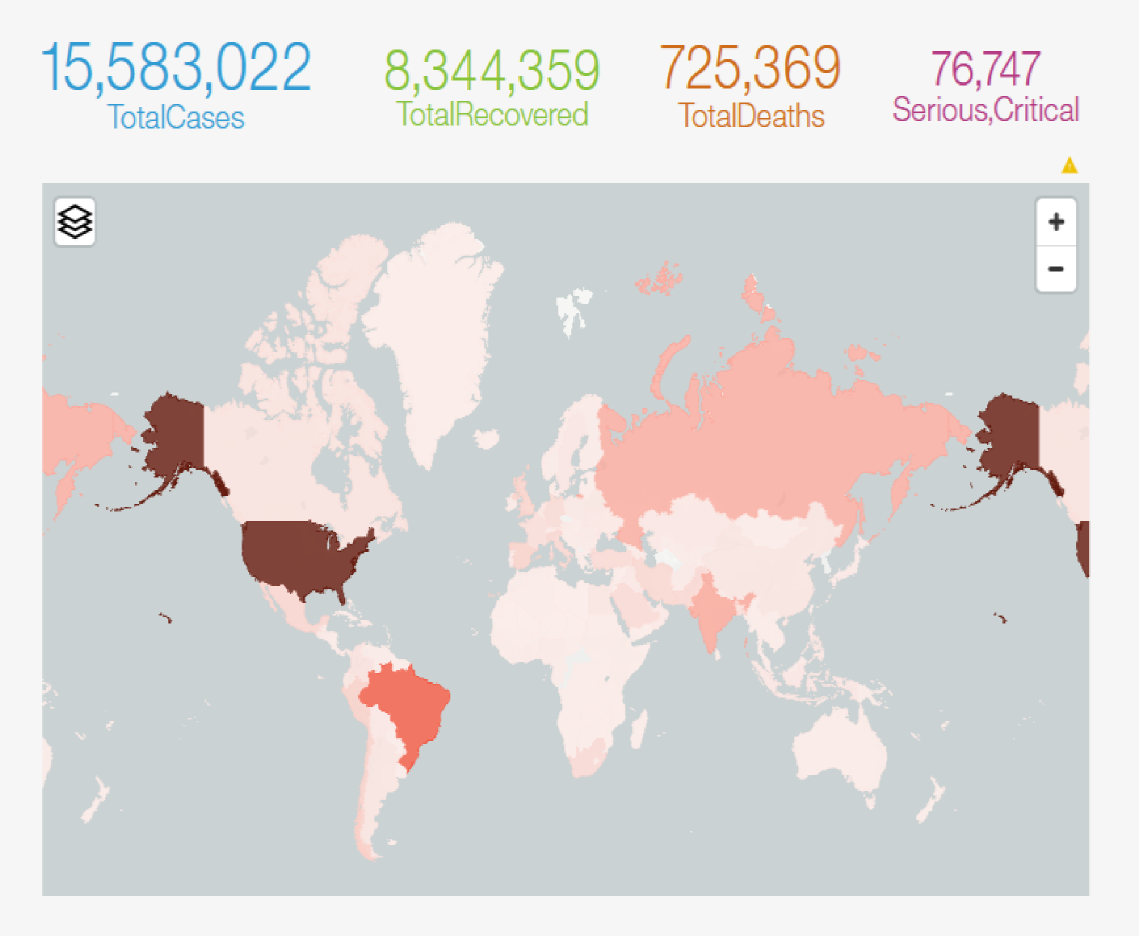


Situation as of 07 July 2020

737,399
Total

456,831
Recovered

20,642
Deceased

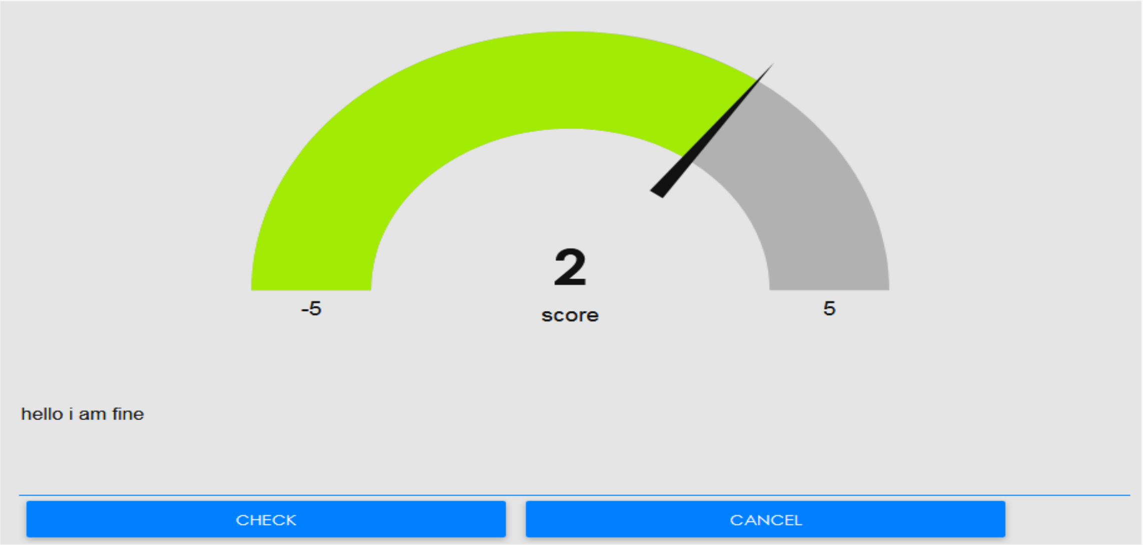


15,583,022
TotalCases

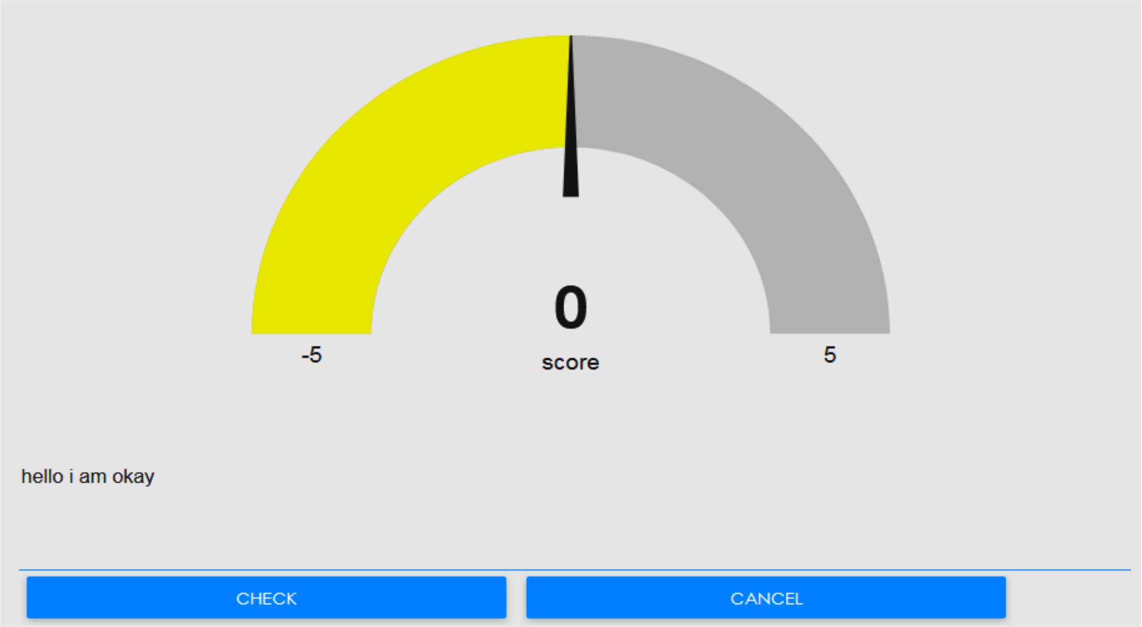
8,344,359
TotalRecovered

725,369
TotalDeaths

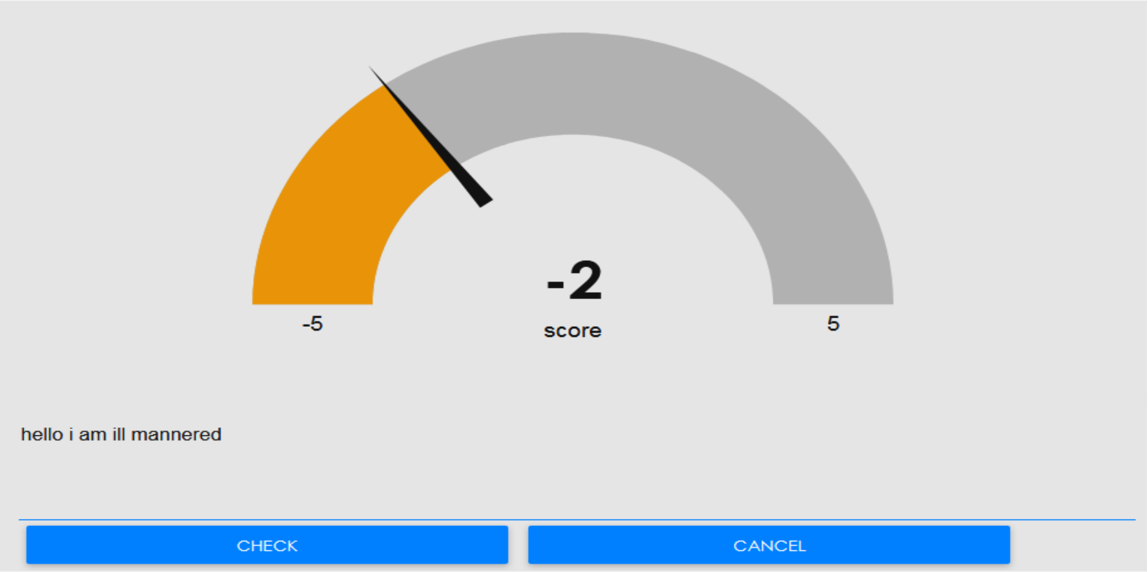
76,747
Serious,Critical



hello i am fine

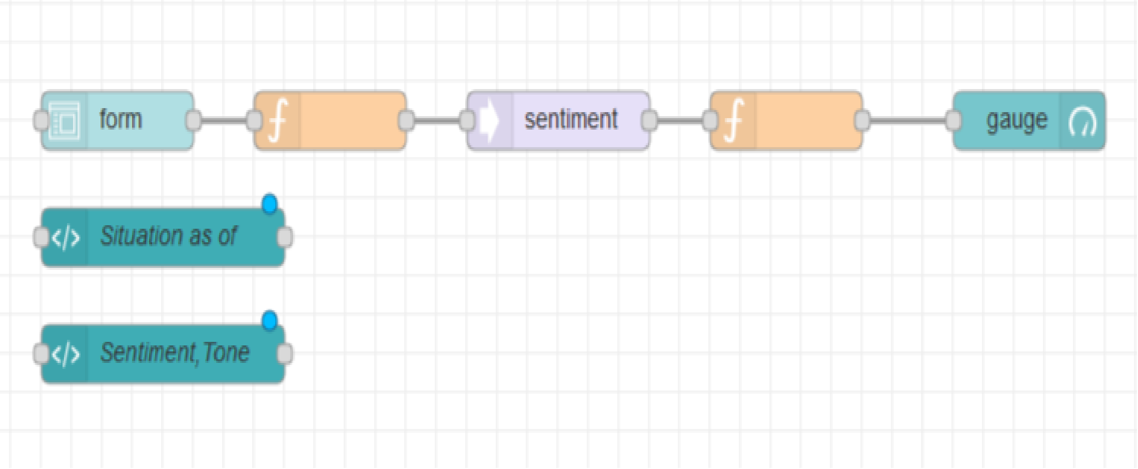


hello i am okay



hello i am ill mannered

5.Node red flow code



Insert the following nodes into NODE RED:

1. ui_form(1)
2. function(2)
3. sentiment(1)
4. ui_gauge(1)
5. ui_template(2)

6.Results

Youtube Link:
https://www.youtube.com/watch?v=Rv6r3Ey-q9I&feature=youtu.be

Github Link:
https://github.com/SmartPracticeschool/SBSPS-Challenge-3580-Emo-co-ti

Node red Link:
https://node-red-emocoti.mybluemix.net/ui/

7. Advantages and Disadvantages

7.1 Advantages

Helps to predict further situation about lockdown.
Clear vision of all data of active cases,recovered and death related cases upto date.
Tweet classifier according to moods i.e. tones.
An extra topping to all this is you could analyze your tweet and as a result it gives sentiment accordingly before you tweet it.
World level analysis is also added here.

7.2 Disadvantages

It works on static data.

8.Application

Help in taking decisions about further situation from past analysis.

One do not need to surf lots of boring data an interesting representation is displayed rather than it.

Score gauge which gives positive , negative or neutral as per your tweet.

9.Conclusion

Finally here we conclude that this dasboard would help anyone to see data in very interesting and simple manner which could be very helpful for predictions.

10.Future Scope

Here in dasboard we may enhance it by adding chatbot which gives all information regrading datas , about coronavirus and all recent news regarding this.

11.Bibliography

1. Project Template:

https://www.youtube.com/embed/LOCkV-mENq8

2 .IBM Cloud:

https://www.ibm.com/cloud/get-started

3 . Node-RED Starter Application :

https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/

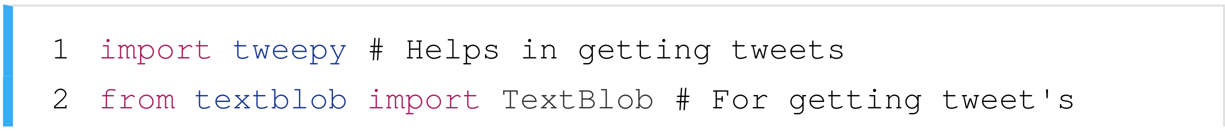
4.Smartbridge online bootcamp:

https://www.youtube.com/watch?v=x_5kH26xics

12.Appendix

Source Code:

Extracting, Pre processing tweets



```python
import tweepy # Helps in getting tweets
from textblob import TextBlob # For getting tweet's
```

```python
subjectivity and polarity
from twython import Twython # Helps in getting tweets
from wordcloud import WordCloud # Helps in forming a word cloud
import pandas as pd # Making and Manipulating a dataframe
import numpy as np # Helps in various vector and matrix calculations
import re # The Regex library
import matplotlib.pyplot as plt # For plotting various visualizations
from functools import reduce # An important method which will help in joining data frames
plt.style.use('fivethirtyeight')
import os # The os library to get current path and various other things
from PIL import Image # To get a mask for the word cloud
CONSUMER_KEY = "*********************" # Enter your CONSUMER KEY
CONSUMER_SECRET = "*****************************************" # Enter your CONSUMER SECRET KEY
OAUTH_TOKEN = "**********************************" # Enter your OAUTH TOKEN KEY
OAUTH_TOKEN_SECRET = "***************************************" # Enter your OAUTH TOKEN SECRET KEY
twitter = Twython(
    CONSUMER_KEY, CONSUMER_SECRET,
    OAUTH_TOKEN, OAUTH_TOKEN_SECRET)

df1 = pd.read_csv('phase1.csv',header=None) # Reading the IEEE file (which contains tweet id and polarity)
df1.rename(columns = {0:'tweet_id',1:'polarity'}, inplace=True)
# A function for fwtching tweets from tweet ids
def check_error(x):
    try:
        tweet = twitter.show_status(id=x)
        return tweet
    except Exception as e:
        return np.nan

df1['tweets'] = df1['tweet_id'].apply(lambda x : check_error(x))
# Making a copy of the dataframe
df = df1.copy()
#Removing Nulls
df.dropna(inplace=True)
temp = []
for i in df['tweets']:
    temp.append((i['text']))

df['tweet_text'] = pd.Series(temp)
df.head()

def cleanTxt(text):
    text = re.sub('@[A-Za-z0-9]+', '', str(text)) #Removing @mentions
    text = re.sub('#', '', str(text)) # Removing '#' hash tag
    text = re.sub('RT[\s]+', '', str(text)) # Removing RT
    text = re.sub('https?:\/\/\S+', '', str(text)) # Removing hyperlink
    return text

# Clean the tweets
df['tweet_text'] = df['tweet_text'].apply(cleanTxt)

# Show the cleaned tweets
df.head()
df.dropna(inplace=True)

```

```


def cleanTxt(text):
    text = re.sub('@[A-Za-z0-9]+', '', str(text)) #Removing @mentions
    text = re.sub('#', '', str(text)) # Removing '#' hash tag
    text = re.sub('RT[\s]+', '', str(text)) # Removing RT
    text = re.sub('https?:\/\/\S+', '', str(text)) # Removing hyperlink
    return text

# Clean the tweets
df['tweet_text'] = df['tweet_text'].apply(cleanTxt)

# Show the cleaned tweets
df.head()
df.dropna(inplace=True)

# A function to get the sentiment of the people on the basis of the polarity
def getAnalysis(score):
    if score < 0:
        return 'Negative'
    elif score == 0:
        return 'Neutral'
    else:
        return 'Positive'

df['Sentiment'] = df['Polarity'].apply(getAnalysis)
# Show the dataframe
df.head(10)

# Storing data in  a csv for further usage.
data = df.to_csv('data1.csv')

```

Machine Learning

```python
import tweepy # Helps in getting tweets
from textblob import TextBlob # For getting tweet's subjectivity and polarity
from twython import Twython # Helps in getting tweets
from wordcloud import WordCloud # Helps in forming a word cloud
import pandas as pd # Making and Manipulating a dataframe
import numpy as np # Helps in various vector and matrix calculations
import re # The Regex library
import matplotlib.pyplot as plt # For plotting various visualizations
from functools import reduce # An important method which will help in joining data frames
plt.style.use('fivethirtyeight')
import os # The os library to get current path and various other things
from PIL import Image # To get a mask for the word cloud
# Including various libraries for performing Machine Learning
from sklearn.model_selection import train_test_split
from stop_words import get_stop_words
from collections import Counter
import string

from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

df1 = pd.read_csv('data1.csv')# Reading the first file
df1.drop(df1.columns[0],axis=1,inplace=True)

df2 = pd.read_csv('data2.csv')# Reading the second file
df2.drop(df2.columns[0],axis=1,inplace=True)

df = reduce(lambda  left,right: pd.merge(left,right,on=['tweet_id','tweets','tweet_text', 'Subjectivity', 'Polarity', 'Sentiment','retweets','response','date','location'],
                                            how='outer'), data_frames)# Joining all the dataframes using reduce and merge
# Splitting data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['Sentiment'], test_size=0.1, random_state = 1)
# Define functions to process Tweet text and remove stop words
def ProTweets(tweet):
    tweet = ''.join(c for c in tweet if c not in string.punctuation)
    tweet = re.sub('((www\S+)|(http\S+))', 'urlsite', tweet)
    tweet = re.sub(r'\d+', 'contnum', tweet)
    tweet = re.sub(' +',' ', tweet)
    tweet = tweet.lower().strip()
    return tweet

def rmStopWords(tweet, stop_words):
    text = tweet.split()
    text = ' '.join(word for word in text if word not in
stop_words)
    return text

# Splitting data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['Sentiment'], test_size=0.1, random_state = 1)

# Applying the Support Vector Machine Learning technique to predict our results.
vectorizer = CountVectorizer()
svm = LinearSVC()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
_ = svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
print(classification_report(y_test, y_pred))

print(confusion_matrix(y_test, y_pred))
```