INTRODUCTION

1.1 Overview

Sentiment analysis on tweets related to COVID 19.

1.2 Purpose

It has been 100+ days since the earliest Lockdown in India.
How have you been feeling through this special time? Do you know how others are responding to the pandemic?
The Corona Virus endangers our physical health indeed, but alongside, social distancing also poses a threat to our emotional stability. Thus, it is crucial to understand public sentiments under COVID-19. Twitter is a great platform to use where we get abundance of public opinion. Hence we deployed Sentiment Analysis on tweets. This model provides a platform wherein you can enter a tweet and it tells the sentiment behind it. We can use this result to decide whether we should be posting this tweet.

LITERATURE SURVEY

2.1 Existing problem

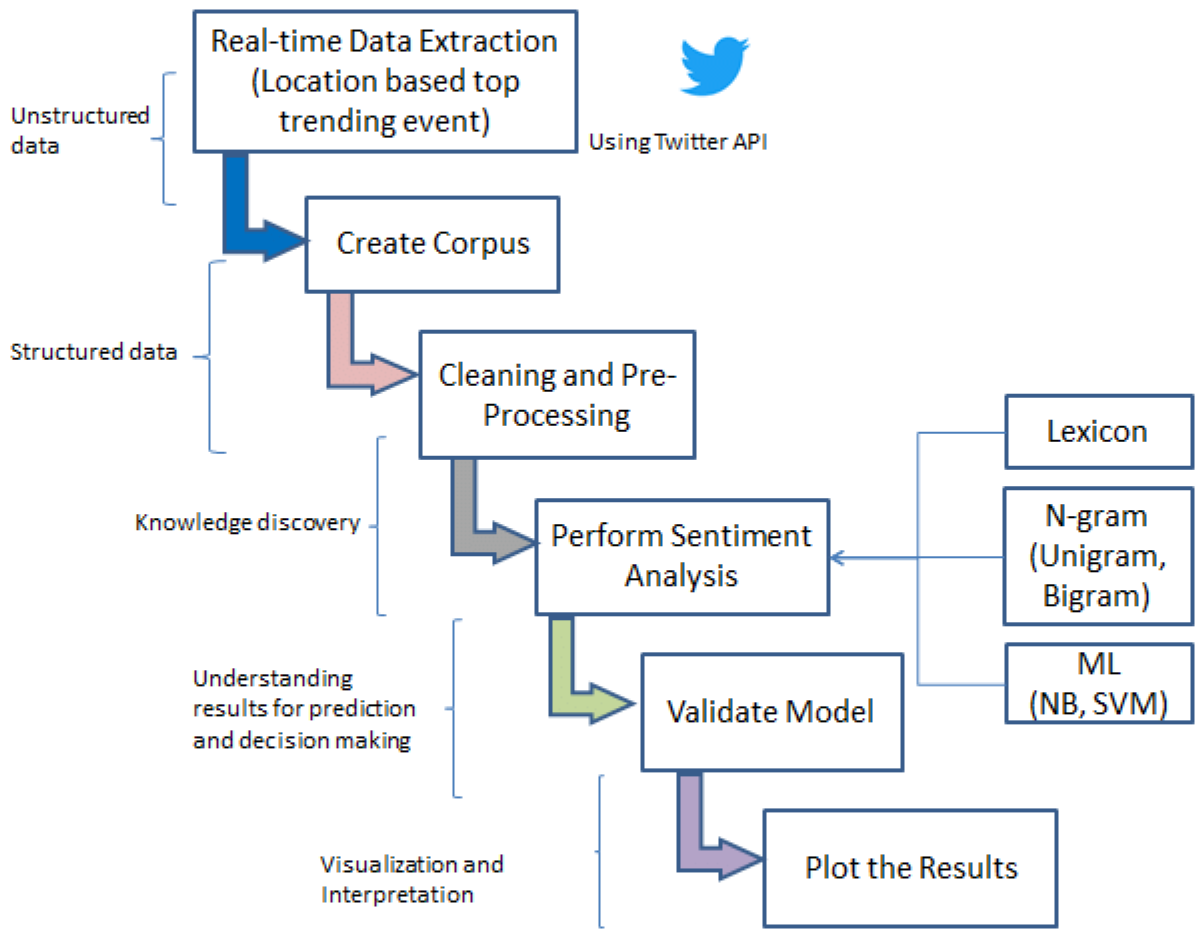
People on Twitter often get tired of negative and disrespectful users. Twitter can be a very dark place where the ugliest voices are often the loudest. Some people on Twitter really don’t care at all about who they hurt, whether that person is you or someone else. Every second, people are tweeting out rude, wrong, and racist things. The vast majority of which are nothing but lies. There has to be some mechanism to monitor the content that goes up on social media.

2.2 Proposed solution

The following model provides a solution for the same. The content of the tweet can be sent as a input to the model. The model will determine whether the following tweet has a positive or negative sentiment. This result can be used by the user to reflect on their words and change it accordingly. This can also be implemented realtime on twitter wherein the user is not allowed to post the tweet if it has a negative sentiment.

THEORITICAL ANALYSIS

3.1 Block diagram


Real-time Data Extraction (Location based top trending event)
Using Twitter API
Unstructured data
Create Corpus
Structured data
Cleaning and Pre-Processing
Knowledge discovery
Perform Sentiment Analysis
Lexicon
N-gram (Unigram, Bigram)
ML (NB, SVM)
Understanding results for prediction and decision making
Validate Model
Visualization and Interpretation
Plot the Results

3.2 Hardware / Software designing

The software used for this model is:

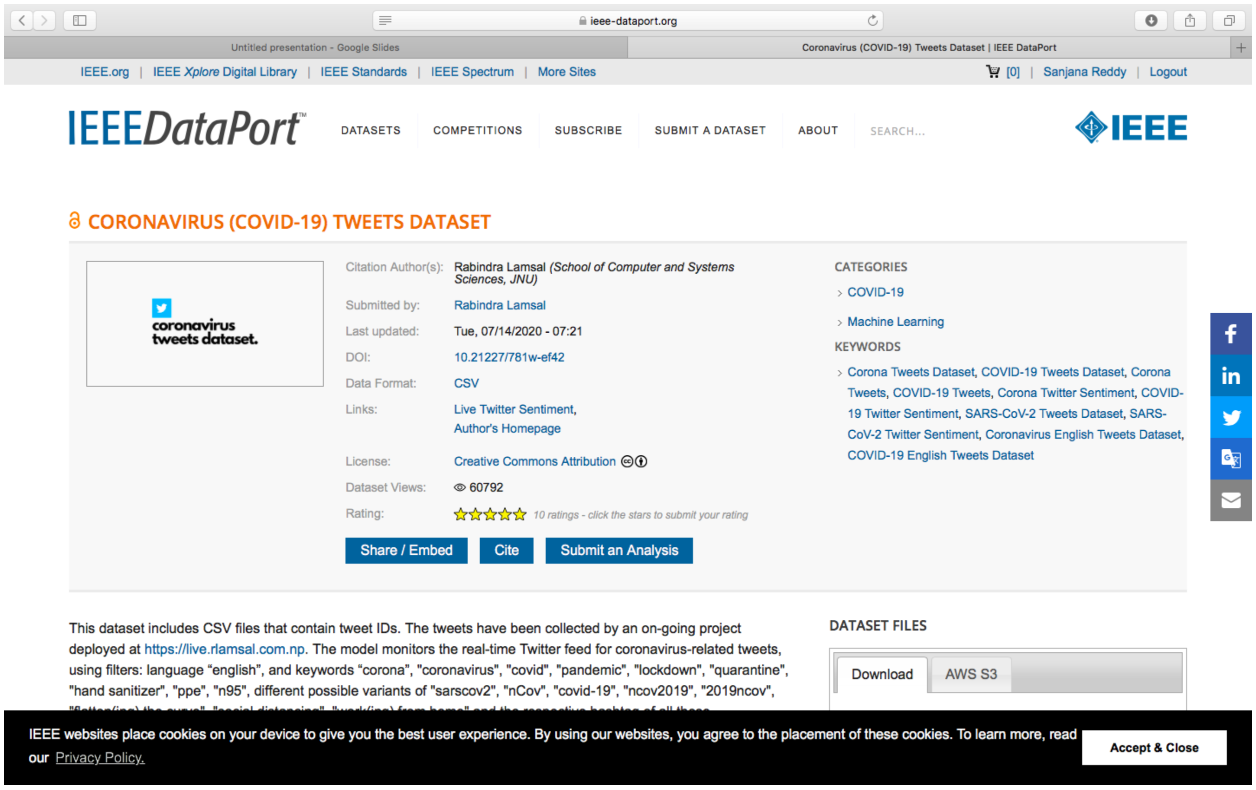
1. jupyter notebook
2. spyder
3. twitter developemont account

Libraries used

1. flask
2. scikit learn
3. NLTK
4. Numpy
5. gunicorn

EXPERIMENTAL INVESTIGATIONS

Dataset:



ieee-dataport.org

Untitled presentation - Google Slides | Coronavirus (COVID-19) Tweets Dataset | IEEE DataPort

IEEE.org | IEEE *Xplore* Digital Library | IEEE Standards | IEEE Spectrum | More Sites   [0] | Sanjana Reddy | Logout

IEEEDataPort™ DATASETS COMPETITIONS SUBSCRIBE SUBMIT A DATASET ABOUT SEARCH... IEEE

CORONAVIRUS (COVID-19) TWEETS DATASET

Citation Author(s):	Rabindra Lamsal *(School of Computer and Systems Sciences, JNU)*
Submitted by:	Rabindra Lamsal
Last updated:	Tue, 07/14/2020 - 07:21
DOI:	10.21227/781w-ef42
Data Format:	CSV
Links:	Live Twitter Sentiment, Author's Homepage
License:	Creative Commons Attribution
Dataset Views:	60792
Rating:	★★★★★ *10 ratings - click the stars to submit your rating*

Share / Embed Cite Submit an Analysis

CATEGORIES

- COVID-19
- Machine Learning

KEYWORDS

- Corona Tweets Dataset, COVID-19 Tweets Dataset, Corona Tweets, COVID-19 Tweets, Corona Twitter Sentiment, COVID-19 Twitter Sentiment, SARS-CoV-2 Tweets Dataset, SARS-CoV-2 Twitter Sentiment, Coronavirus English Tweets Dataset, COVID-19 English Tweets Dataset

This dataset includes CSV files that contain tweet IDs. The tweets have been collected by an on-going project deployed at https://live.rlamsal.com.np. The model monitors the real-time Twitter feed for coronavirus-related tweets, using filters: language "english", and keywords "corona", "coronavirus", "covid", "pandemic", "lockdown", "quarantine", "hand sanitizer", "ppe", "n95", different possible variants of "sarscov2", "nCov", "covid-19", "ncov2019", "2019ncov",

DATASET FILES

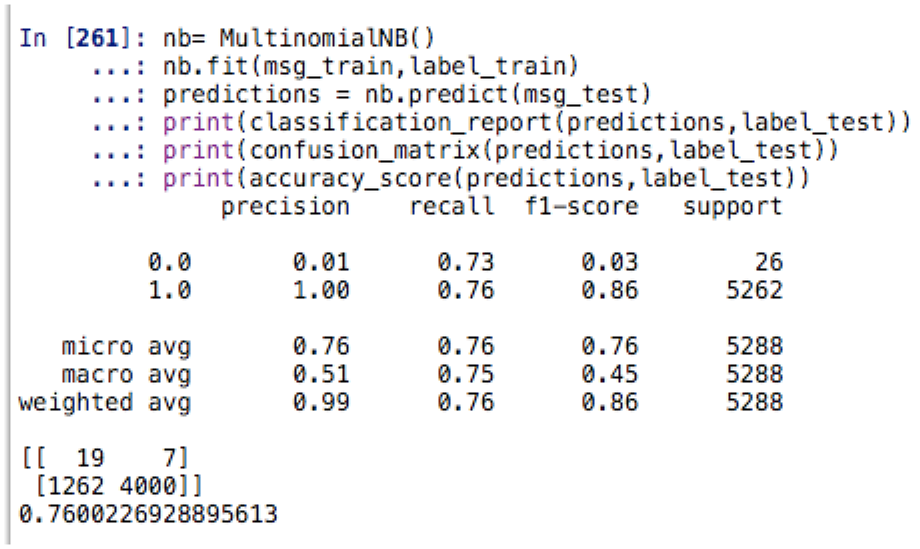
Download AWS S3

IEEE websites place cookies on your device to give you the best user experience. By using our websites, you agree to the placement of these cookies. To learn more, read our Privacy Policy.

Accept & Close

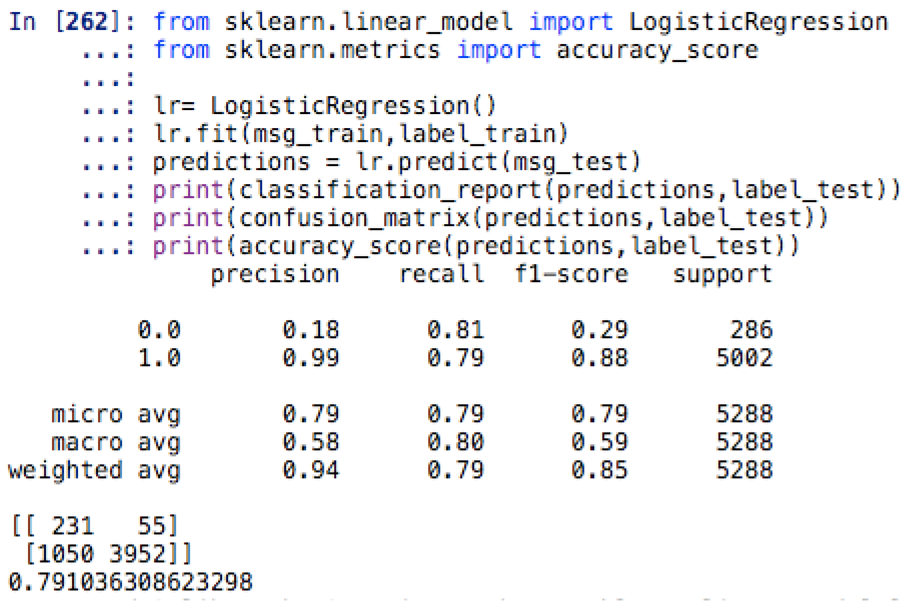
Since this is a classification problem, three classification algorithms were applied

1. logistic regression
2. Naive Bayes
3. Support vector machines



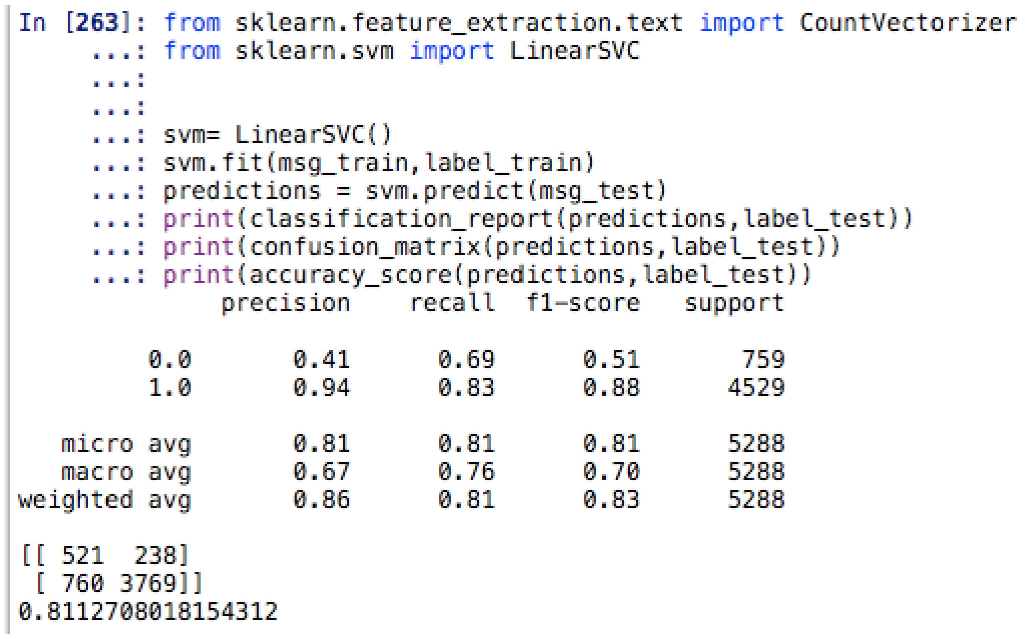
```
In [261]: nb= MultinomialNB()
     ...: nb.fit(msg_train,label_train)
     ...: predictions = nb.predict(msg_test)
     ...: print(classification_report(predictions,label_test))
     ...: print(confusion_matrix(predictions,label_test))
     ...: print(accuracy_score(predictions,label_test))
              precision    recall  f1-score   support

         0.0       0.01      0.73      0.03        26
         1.0       1.00      0.76      0.86      5262

   micro avg       0.76      0.76      0.76      5288
   macro avg       0.51      0.75      0.45      5288
weighted avg       0.99      0.76      0.86      5288

[[  19    7]
 [1262 4000]]
0.7600226928895613
```

Naive Bayes gave an accuracy of 76%



```
In [262]: from sklearn.linear_model import LogisticRegression
     ...: from sklearn.metrics import accuracy_score
     ...:
     ...: lr= LogisticRegression()
     ...: lr.fit(msg_train,label_train)
     ...: predictions = lr.predict(msg_test)
     ...: print(classification_report(predictions,label_test))
     ...: print(confusion_matrix(predictions,label_test))
     ...: print(accuracy_score(predictions,label_test))
              precision    recall  f1-score   support

         0.0       0.18      0.81      0.29       286
         1.0       0.99      0.79      0.88      5002

   micro avg       0.79      0.79      0.79      5288
   macro avg       0.58      0.80      0.59      5288
weighted avg       0.94      0.79      0.85      5288

[[ 231   55]
 [1050 3952]]
0.791036308623298
```

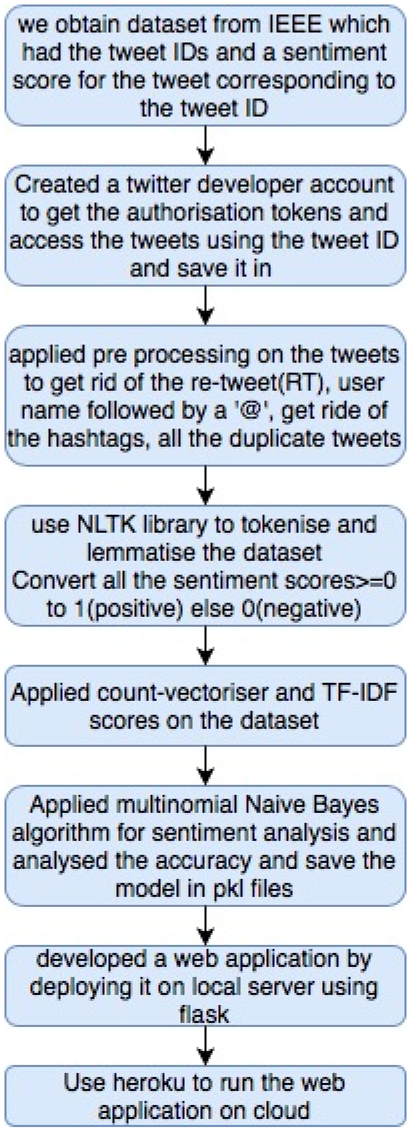
Logistic regression gave an accuracy of 79%



```
In [263]: from sklearn.feature_extraction.text import CountVectorizer
     ...: from sklearn.svm import LinearSVC
     ...:
     ...:
     ...: svm= LinearSVC()
     ...: svm.fit(msg_train,label_train)
     ...: predictions = svm.predict(msg_test)
     ...: print(classification_report(predictions,label_test))
     ...: print(confusion_matrix(predictions,label_test))
     ...: print(accuracy_score(predictions,label_test))
              precision    recall  f1-score   support

         0.0       0.41      0.69      0.51       759
         1.0       0.94      0.83      0.88      4529

   micro avg       0.81      0.81      0.81      5288
   macro avg       0.67      0.76      0.70      5288
weighted avg       0.86      0.81      0.83      5288

[[ 521  238]
 [ 760 3769]]
0.8112708018154312
```

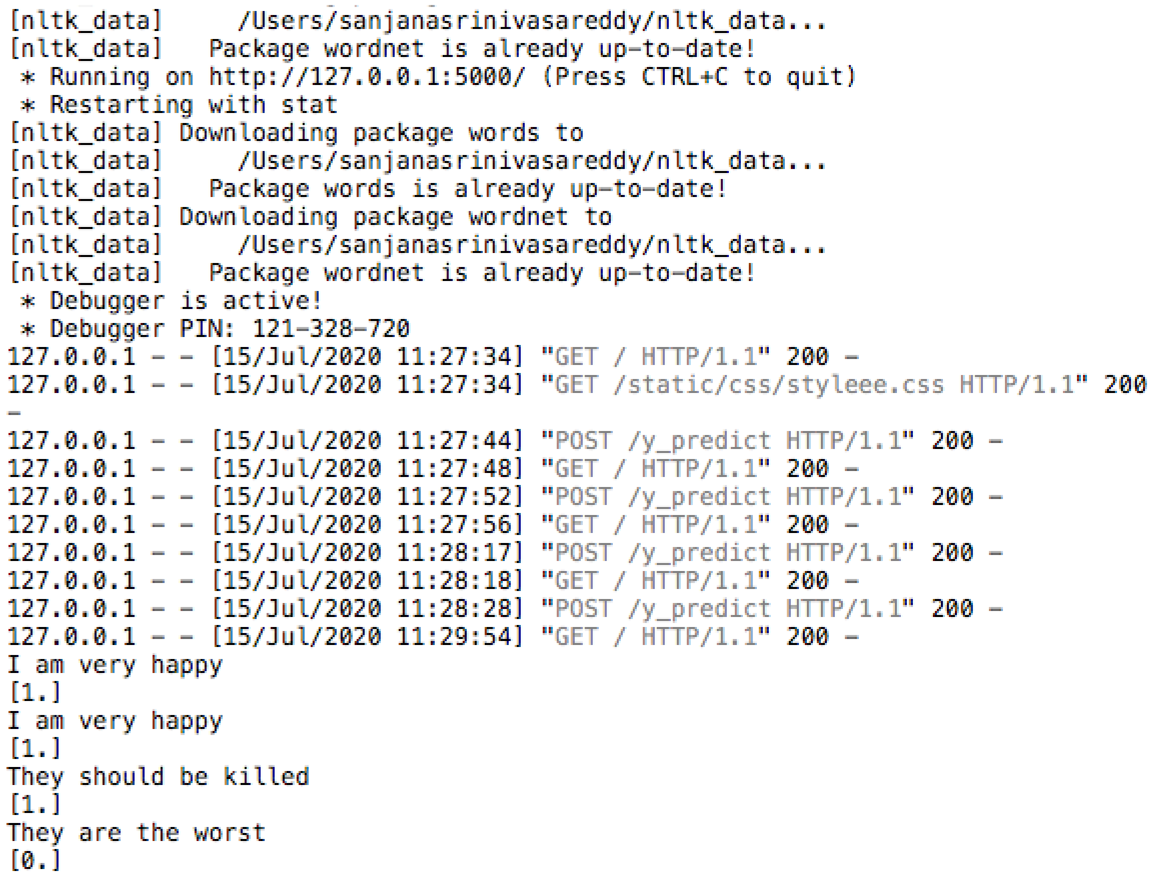
Support vector machines gave an accuracy of 81%

FLOWCHART


we obtain dataset from IEEE which had the tweet IDs and a sentiment score for the tweet corresponding to the tweet ID
Created a twitter developer account to get the authorisation tokens and access the tweets using the tweet ID and save it in
applied pre processing on the tweets to get rid of the re-tweet(RT), user name followed by a '@', get ride of the hashtags, all the duplicate tweets
use NLTK library to tokenise and lemmatise the dataset Convert all the sentiment scores>=0 to 1(positive) else 0(negative)
Applied count-vectoriser and TF-IDF scores on the dataset
Applied multinomial Naive Bayes algorithm for sentiment analysis and analysed the accuracy and save the model in pkl files
developed a web application by deploying it on local server using flask
Use heroku to run the web application on cloud

RESULT

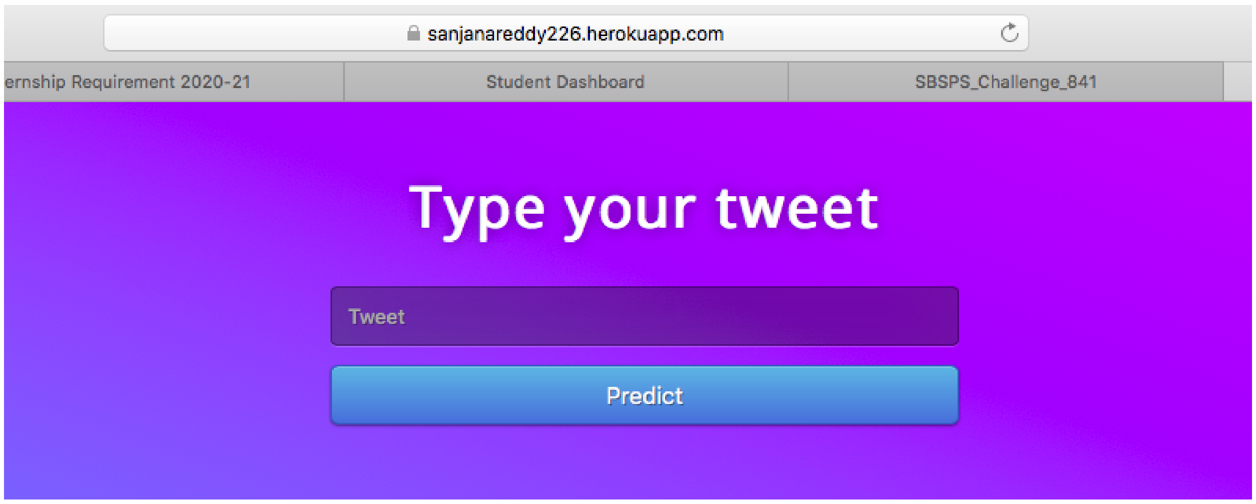
Running the model locally:

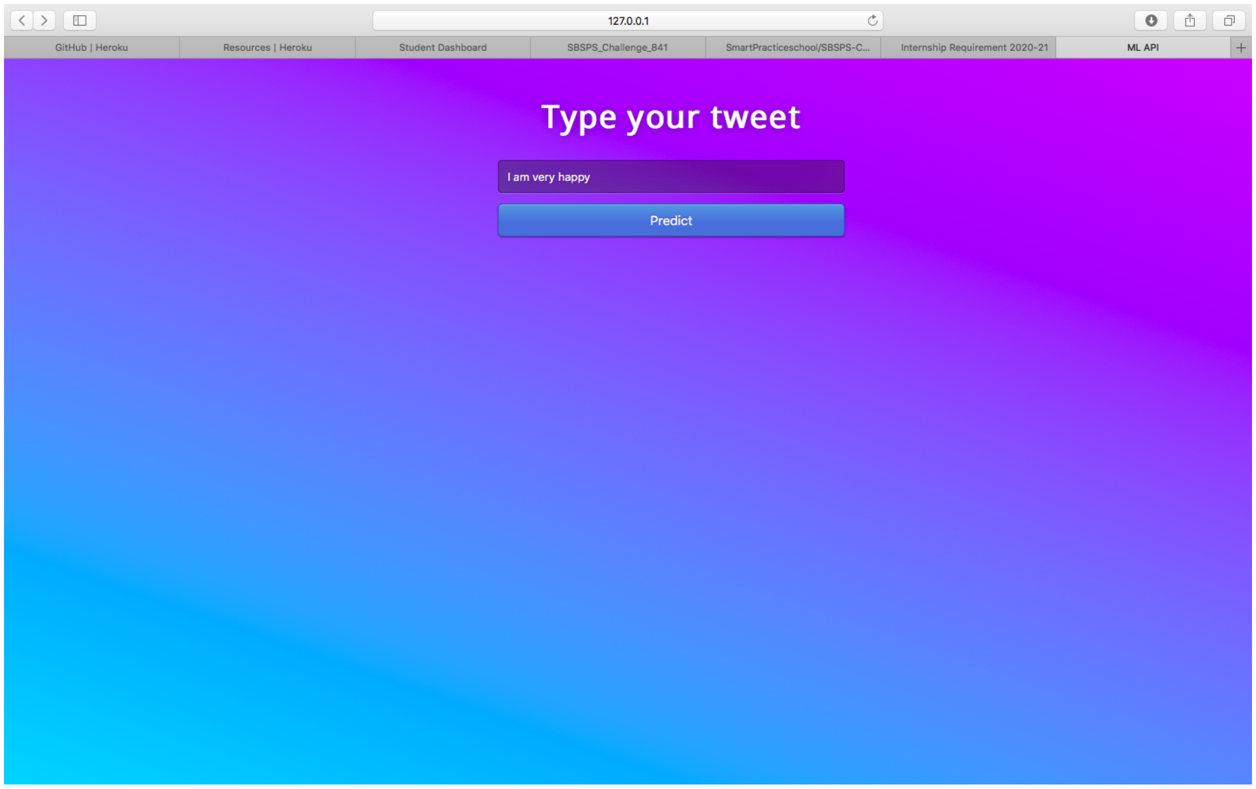


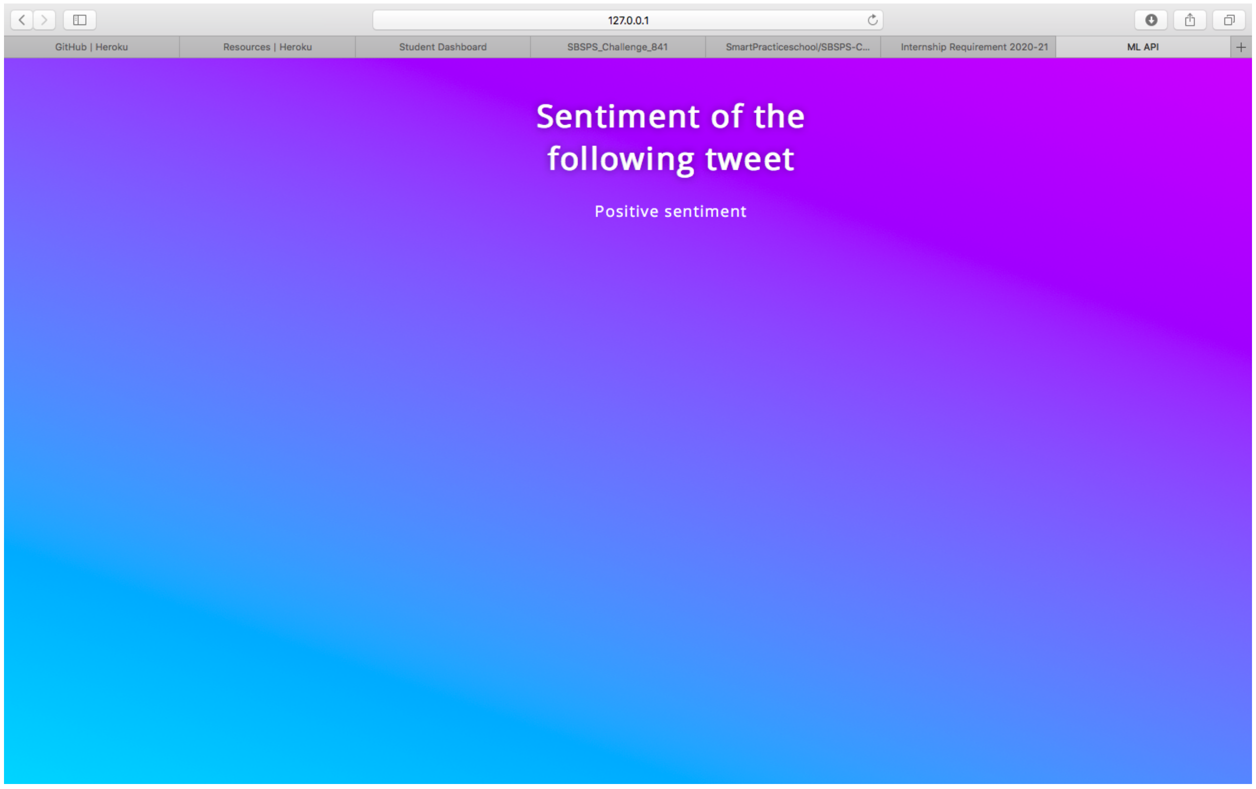
```
[nltk_data]     /Users/sanjanasrinivasareddy/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
[nltk_data] Downloading package words to
[nltk_data]     /Users/sanjanasrinivasareddy/nltk_data...
[nltk_data]   Package words is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/sanjanasrinivasareddy/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
 * Debugger is active!
 * Debugger PIN: 121-328-720
127.0.0.1 - - [15/Jul/2020 11:27:34] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2020 11:27:34] "GET /static/css/styleee.css HTTP/1.1" 200
-
127.0.0.1 - - [15/Jul/2020 11:27:44] "POST /y_predict HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2020 11:27:48] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2020 11:27:52] "POST /y_predict HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2020 11:27:56] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2020 11:28:17] "POST /y_predict HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2020 11:28:18] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2020 11:28:28] "POST /y_predict HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2020 11:29:54] "GET / HTTP/1.1" 200 -
I am very happy
[1.]
I am very happy
[1.]
They should be killed
[1.]
They are the worst
[0.]
```

The deployed model can be accessed though this link:

https://sanjanareddy226.herokuapp.com

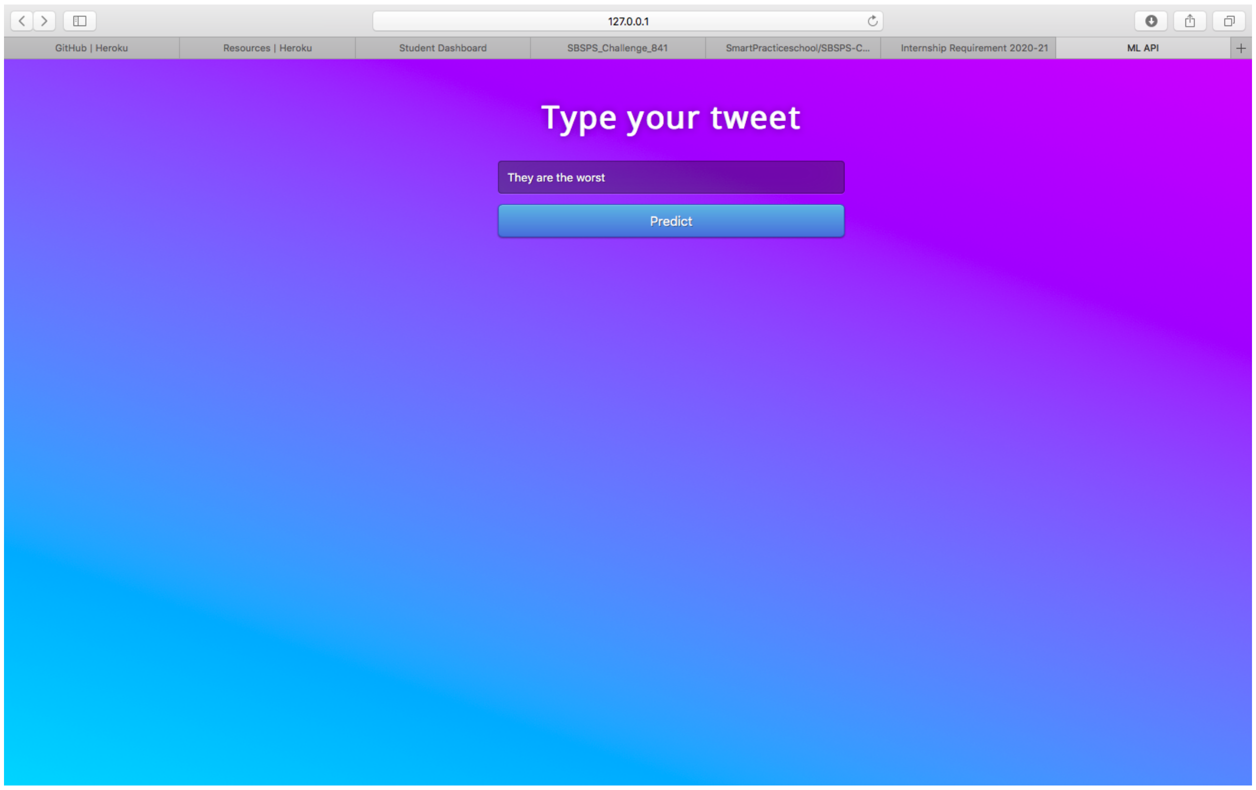


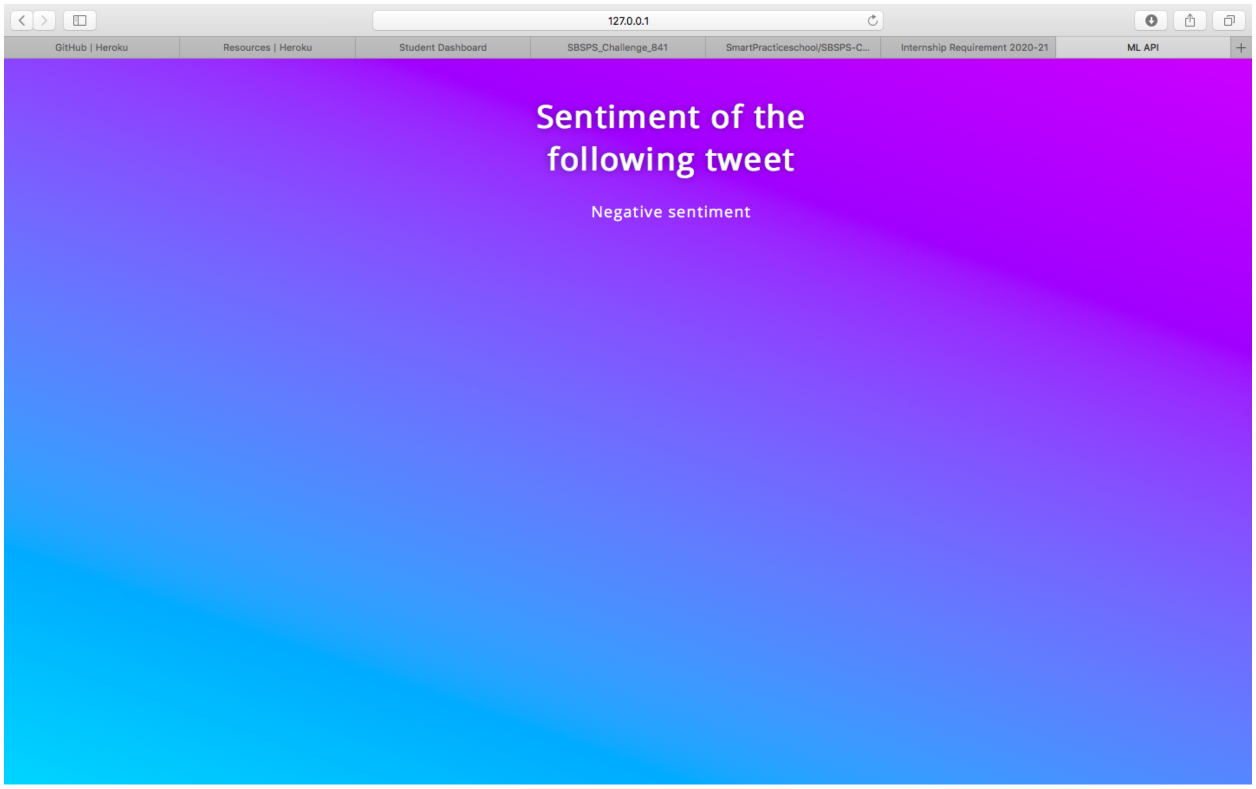




Sentiment of the following tweet

Positive sentiment





Sentiment of the following tweet

Negative sentiment

ADVANTAGES & DISADVANTAGES

Advantages

Information obtained from sentiment analysis can be applied to make wiser decisions related to the use of resources, to make improvements in organizations, providing better products/services, and ultimately to improve the citizen lifestyle and the human relations in order to achieve a better society.

Social media is the current environment for data collection and analysis of sentiments of people. People can share and comment on everything, from personal thoughts to common events or topics in society. The access to social media also can provide more information in the form of hidden metadata.

Disadvantage

Despite the possible positive outcomes, there are some disadvantages in applying automatic analysis due to the difficulty to implement it because of the ambiguity of natural language and also the characteristics of the posted content. The analysis of tweets is an example of this, for they are usually coupled with hashtags, emoticons and links, creating difficulties in determining the expressed sentiment. In addition, there is a need for automatic techniques that require large datasets of annotated posts or lexical databases where emotional words are associated with sentiment values. Another important aspect is that analyses are suitable for the English language, in which there is a limitation for other languages.

In the field of sentiment analysis are some challenges in a range of scenarios, in terms of architecture and application domains with unclear or scarce datasets. Also, there is a lack of labelled data, which can pose a barrier to the advancements in this area.

APPLICATIONS

Sentiment analysis is extremely useful in social media monitoring as it allows us to gain an overview of the wider public opinion behind certain topics.

This makes that process quicker and easier than ever before, thanks to real-time monitoring capabilities.

The applications of sentiment analysis are broad and powerful. The ability to extract insights from social data is a practice that is being widely adopted by organisations across the world.

CONCLUSION

It is known that how sentiment analysis has its limitations and is not to be used as a

100% accurate marker.
As with any automated process, it is prone to error and often needs a human eye to watch over it.
Beyond reliability, it's important to acknowledge that human's expression doesn't fit into just three buckets; not all sentiment can be categorised as simply as positive, negative or neutral.

FUTURE SCOPE

While it's difficult to speculate how a relatively immature system might evolve in the future, there is a general assumption that sentiment analysis needs to move beyond a one-dimensional positive to negative scale.
We will see a shift in perception of the reliability of sentiment analysis. Users will become more comfortable with the idea that the automatic analysis of individual text material is hard to match human performance.
Instead, the focus will be on how to make results interpretable and actionable. In the meantime, we'll be ensuring we are working at making sentiment analysis as accurate and easy to understand as possible.

BIBILOGRAPHY

1.https://towardsdatascience.com/the-real-world-as-seen-on-twitter-sentiment-analysis-part-one-5ac2d06b63fb

2.https://towardsdatascience.com/twitter-sentiment-analysis-classification-using-nltk-python-fa912578614c

3.https://towardsdatascience.com/twitter-sentiment-analysis-based-on-news-topics-during-covid-19-c3d738005b55

4.https://medium.com/analytics-vidhya/deploy-machinelearning-model-with-flask-and-heroku-2721823bb653

5.https://towardsdatascience.com/designing-a-machine-learning-model-and-deploying-it-using-flask-on-heroku-9558ce6bde7b

6.https://www.bloomberg.com/opinion/articles/2018-10-30/twitter-s-problem-is-bigger-than-the-like-button

7.https://www.brandwatch.com/blog/understanding-sentiment-analysis/

8.https://ieee-dataport.org/open-access/coronavirus-covid-19-tweets-dataset

APPENDIX
A. Source code

twitter.py to extract tweets from twitter

```
from twython import Twython,TwythonRateLimitError,TwythonError
import pandas as pd
import itertools
import time

CONSUMER_KEY = "8Y7JmfyzAGoN4ynp38OBokbad"
CONSUMER_SECRET = "w4rWBwRH1YbbRYyVu3oRLLDfb0GtEYYYoIniAoqxFm4jkijhoQ"
OAUTH_TOKEN = "1272828877954404354-KziDz09I8t9LFIoq5LFUPSqfm4iUzP"
OAUTH_TOKEN_SECRET = "83jssKdgUYhzyBRZbof4Eys0FTWXr0rAO1xaTNucAz5S6"
twitter = Twython(
  CONSUMER_KEY, CONSUMER_SECRET,
   OAUTH_TOKEN, OAUTH_TOKEN_SECRET)

ds=pd.read_csv('/Users/sanjanasrinivasareddy/Documents/ibm-hack/corona_tweets_3
1.csv')
ds=ds.iloc[3600:4500,:]
#x=ds['Id']
x=[]
a=0
for i,j in ds.iterrows():
   #print(j['Id'])
   print(a)
   a=a+1
   try:
      tweet = twitter.show_status(id=int(j['Id']))
      print(int(j['Id']))
      print(tweet['text'])
```

```
        #ds['Tweet']=tweet['text']
        x.append(tweet['text'])
    except TwythonRateLimitError as error:
        remainder = float(twitter.get_lastfunction_header(header='x-rate-limit-reset')) -
time.time()
        time.sleep(remainder)
        tweet = twitter.show_status(id=int(j['Id']))
        print(int(j['Id']))
        print(tweet['text'])
        #ds['Tweet']=tweet['text']
        x.append(tweet['text'])
        continue
    except TwythonError as error:
        print("")
        #ds['Tweet']=""
        x.append("")
    except:
        print("")

        x.append("")
ds['Tweet']=x
ds.to_csv('file3.csv', mode='a', header=False)
```

app.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Jul 11 16:49:13 2020

@author: sanjanasrinivasareddy
"""
#from sklearn.externals import joblib


import numpy as np
import os
```

```
#import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
#import joblib
import nltk

nltk.download('words')
words = set(nltk.corpus.words.words())
import re
from sklearn.pipeline import Pipeline
nltk.download('wordnet')
import joblib
from nltk.stem.wordnet import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
app = Flask(__name__)
model = pickle.load(open('decision.pkl', 'rb'))
sc=pickle.load(open('vector.pkl','rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/y_predict',methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML GUI
    '''
    def normalization(tweet_list):
        lem = WordNetLemmatizer()
        normalized_tweet = []
        for word in tweet_list:
            normalized_text = lem.lemmatize(word,'v')
            normalized_tweet.append(normalized_text)
        return normalized_tweet

    def no_user_alpha(tweet):
        tweet_list = [ele for ele in tweet.split() if ele != 'user']
        clean_tokens = [t for t in tweet_list if re.match(r'[^\W\d]*$', t)]
        clean_s = ' '.join(clean_tokens)
        clean_mess = [word for word in clean_s.split()]
        return clean_mess
    x_test = request.form.get('Tweet')
    print(x_test)
    x_test=" ".join(w for w in nltk.wordpunct_tokenize(x_test) if w.lower() in words or not
w.isalpha())
    txt = x_test
    txt=re.sub(r'@[A-Z0-9a-z_:]+','',txt)#replace username-tags
    txt=re.sub(r'^[RT]+','',txt)#replace RT-tags
    txt = re.sub('https?://[A-Za-z0-9./]+','',txt)#replace URLs
    txt=re.sub("[^a-zA-Z]", " ",txt)#replace hashtags
    txt=no_user_alpha(txt)
    txt=normalization(txt)
    txt=[" ".join(txt)]
    x_test=sc.transform(txt)
    #print(x_test)
    prediction = model.predict(x_test)
    print(prediction)
#    output=prediction[0][0]
#    return render_template('index.html', prediction_text='the employee stayed or no
{}'.format(output))
    if(prediction==1):
        return render_template('indexx.html', prediction_text='Positive sentiment')
    else:
        return render_template('indexx.html', prediction_text='Negative sentiment')
    return render_template('index.html', prediction_text='The tweet has that following
sentiment'.format(output))



    return jsonify(output)

if __name__ == "__main__":
    app.run(debug=True)
```

model.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 30 13:51:20 2020

@author: sanjanasrinivasareddy
"""
from nltk.stem.wordnet import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

#Model Selection and Validation
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.metrics import confusion_matrix, classification_report,accuracy_score
import nltk
import sklearn
import pandas as pd
df1 = pd.read_csv('file.csv', encoding = 'unicode_escape')
df2 = pd.read_csv('file2.csv', encoding = 'unicode_escape')
df3= pd.read_csv('file3.csv', encoding = 'unicode_escape')
df4= pd.read_csv('file4.csv', encoding = 'unicode_escape')
df5= pd.read_csv('file5.csv', encoding = 'unicode_escape')
df6= pd.read_csv('file6.csv', encoding = 'unicode_escape')
df7= pd.read_csv('file7.csv', encoding = 'unicode_escape')

##########
df8= pd.read_csv('file8.csv', encoding = 'unicode_escape')
df9= pd.read_csv('file9.csv', encoding = 'unicode_escape')
##########



print(len(df1.index))
df1=df1.drop('Id',1)
```

```
df2=df2.drop('Id',1)
df2=df2.drop('Unnamed: 0',1)

df3=df3.drop('Id',1)
df3=df3.drop('Unnamed: 0',1)

df4=df4.drop('Id',1)
df4=df4.drop('0',1)

df5=df5.drop('Id',1)
df5=df5.drop('0',1)

df6=df6.drop('Id',1)
df6=df6.drop('Unnamed: 0',1)

df7=df7.drop('Id',1)
df7=df7.drop('Unnamed: 0',1)

##############
df8=df8.drop('Id',1)
df8=df8.drop('900',1)

df9=df9.drop('Id',1)
df9=df9.drop('5900',1)

##############

all_dfs = [df1, df2,df3,df4,df5,df6,df7]
df=pd.concat(all_dfs).reset_index(drop=True)


#######
new_df=[df8,df9]
new_df=pd.concat(new_df).reset_index(drop=True)
serlis=new_df.duplicated(['Tweet']).tolist()
print(serlis.count(True))
```

```
new_df=new_df.drop_duplicates(['Tweet'])

new_df=new_df.iloc[1:,:]
new_df=new_df.reset_index(drop=True)

#######

#serlis=df.duplicated().tolist()
serlis=df.duplicated(['Tweet']).tolist()
print(serlis.count(True))

df=df.drop_duplicates(['Tweet'])

df=df.iloc[1:,:]
df=df.reset_index(drop=True)


nltk.download('wordnet')
def normalization(tweet_list):
        lem = WordNetLemmatizer()
        normalized_tweet = []
        for word in tweet_list:
            normalized_text = lem.lemmatize(word,'v')
            normalized_tweet.append(normalized_text)
        return normalized_tweet

def no_user_alpha(tweet):
    tweet_list = [ele for ele in tweet.split() if ele != 'user']
    clean_tokens = [t for t in tweet_list if re.match(r'[^\W\d]*$', t)]
    clean_s = ' '.join(clean_tokens)
    clean_mess = [word for word in clean_s.split()]
    return clean_mess
import nltk
nltk.download('words')
words = set(nltk.corpus.words.words())
import re
for i in range(len(df)):
    print(i)
    df['Tweet'][i]=" ".join(w for w in nltk.wordpunct_tokenize(df['Tweet'][i]) if w.lower() in
words or not w.isalpha()) #remove non english words
    txt = df.loc[i]["Tweet"]
    txt=re.sub(r'@[A-Z0-9a-z_:]+','',txt)#replace username-tags
    txt=re.sub(r'^[RT]+','',txt)#replace RT-tags
    txt = re.sub('https?://[A-Za-z0-9./]+','',txt)#replace URLs
    txt=re.sub("[^a-zA-Z]", " ",txt)#replace hashtags
    df.at[i,"Tweet"]=txt
    df['Tweet'][i]=no_user_alpha(df['Tweet'][i])
    df['Tweet'][i]=normalization(df['Tweet'][i])
    if(df['Sentiment'][i]>=0):
        df['Sentiment'][i]=1
    else:
        df['Sentiment'][i]=0
#delete_row = df[df.iloc[:,2]==" "].index
#df = df.drop(delete_row)
###############
df=new_df
for i in range(len(df)):
    print(i)
    df['Tweet'][i]=" ".join(w for w in nltk.wordpunct_tokenize(df['Tweet'][i]) if w.lower() in
words or not w.isalpha()) #remove non english words
    txt = df.loc[i]["Tweet"]
    txt=re.sub(r'@[A-Z0-9a-z_:]+','',txt)#replace username-tags
    txt=re.sub(r'^[RT]+','',txt)#replace RT-tags
    txt = re.sub('https?://[A-Za-z0-9./]+','',txt)#replace URLs
    txt=re.sub("[^a-zA-Z]", " ",txt)#replace hashtags
    df.at[i,"Tweet"]=txt
    df['Tweet'][i]=no_user_alpha(df['Tweet'][i])
    df['Tweet'][i]=normalization(df['Tweet'][i])
    if(df['Sentiment'][i]>=0):
        df['Sentiment'][i]=1
    else:
        df['Sentiment'][i]=0
#################
df_new=df
```

```
df_og=df

all_dfs = [df_og,df_new]
df=pd.concat(all_dfs).reset_index(drop=True)


df.to_csv('vector_final.csv', mode='a', header=False)#df_og is already saved here

delete_row = df[df.iloc[:,1]==""].index
df = df.drop(delete_row)
df_final=df
df_new['Tweet']=[" ".join(review) for review in df_new['Tweet'].values]

pipeline = Pipeline([
   ('bow',CountVectorizer(analyzer='word')),  # strings to token integer counts
   ('tfidf', TfidfTransformer())  # integer counts to weighted TF-IDF scores
     # train on TF-IDF vectors w/ Naive Bayes classifier
])
msg_train, msg_test, label_train, label_test = train_test_split(df['Tweet'], df['Sentiment'],
test_size=0.3)
msg_train=pipeline.fit_transform(msg_train)
msg_test=pipeline.transform(msg_test)


nb= MultinomialNB()
nb.fit(msg_train,label_train)
predictions = nb.predict(msg_test)
print(classification_report(predictions,label_test))
print(confusion_matrix(predictions,label_test))
print(accuracy_score(predictions,label_test))

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

lr= LogisticRegression()
lr.fit(msg_train,label_train)
predictions = lr.predict(msg_test)
```

```
print(classification_report(predictions,label_test))
print(confusion_matrix(predictions,label_test))
print(accuracy_score(predictions,label_test))
#    lr.fit(msg_train,label_train)
#    print ("Accuracy for C=%s: %s"
#          % (c, accuracy_score(label_test, lr.predict(label_train))))
#




from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import LinearSVC

svm= LinearSVC()
svm.fit(msg_train,label_train)
predictions = svm.predict(msg_test)
print(classification_report(predictions,label_test))
print(confusion_matrix(predictions,label_test))
print(accuracy_score(predictions,label_test))
#    svm = LinearSVC(C=c)
#    svm.fit(msg_train,label_train)
#    print ("Accuracy for C=%s: %s"
#          % (c, accuracy_score(label_test, lr.predict(label_train))))
#
#
import pickle
pickle.dump(pipeline,open('vector.pkl','wb'))

import pickle
pickle.dump(svm,open('decision.pkl','wb'))

import pickle
pickle.dump(nb,open('decision_nb.pkl','wb'))

import pickle
```

```
pickle.dump(lr,open('decision_lr.pkl','wb'))

import pickle
pickle.dump(svm,open('decision_svm.pkl','wb'))
```