IBM Summer Training cum Internship Pro

2020

SENTIMENT ANALYSIS OF TWEETS REGARDING PEOPLE"S REACTION TO LOCKDOWN ANNOUNCEMENTS

Siddhi Mishra

IGDTUW, Delhi

Table of Content

INTRODUCTION

a. Overview

Sentiment analysis is the process of mining of patterns in texts in order to determine the sentiment or opinion being conveyed in the text. Using sentiment analysis, we can determine the emotion of the text being analyzed and “compute” the positivity, negativity or the neutrality of the tone of the text .Twitter, a popular micro blogging site, provides a huge amount of data regarding the sentiments of people with respect to an event or a product.

The analysis of sentiments of the tweets given out with respect to something can be used to identify the general opinion about it. Hence, sentiment analysis is widely used for consumer analysis and gaining insight into the workings of a society. This project aims to analyse the sentiments of the tweets put out by Indians regarding the COVID-19 lockdown in India and also build a model for predicting the future behaviour of Indian citizens and their emotions regarding the lockdown.

b. Purpose

The purpose of the project is to:

- Perform sentiment analysis on the tweets about the pandemic lockdown in India
- Build a predictive model to analyse the future behaviour of people in case of a lockdown extension

LITERATURE SURVEY

EXISTING SOLUTION:

- **Surveys, questionnaires, etc.**

Manual obtainment of data, after which the sentiment of each person is to be identified and classified. It is a long and tedious process.

- **NCSU Tweet Sentiment Visualization App**

This is a free, cloud-based tool that allows users to query it with a keyword and obtain a detailed report regarding the opinion analysis of tweets associated with the keyword.

The tool analyses the tweets from the past week for a particular keyword and time range of tweets analysed may be shortened depending upon the popularity of the keyword.

Outputs of the query include tweets, geo-mapping visualization, timeline, sentiment scatter plot, heatmap, tag cloud and word cloud.

- **Meaning Cloud**

Meaning Cloud can perform sentiment analysis of multilingual content from multiple sources.

It can identify the sentiment and the global polarity value of each text analysed.
Meaning cloud is a very powerful tool because it is highly equipped to identify contradictions, ambiguous statements, sarcasm and irony.

It also provides the feature for users to upload custom dictionaries for sentiment classification.

- **Social Mention**

Social Mention allows the user to enter a particular keyword search query and obtain content collected from various media platforms such as Flickr, Twitter, YouTube, Google +, Reddit and so on.The tool can also perform searches from a time frame specified by the user and give out all the mentions of the keyword along with a CSV file of the results. It also gives the stats regarding the most popular authors, the reach, sentiment and passion of the content along with the latest time of mention of the keyword.

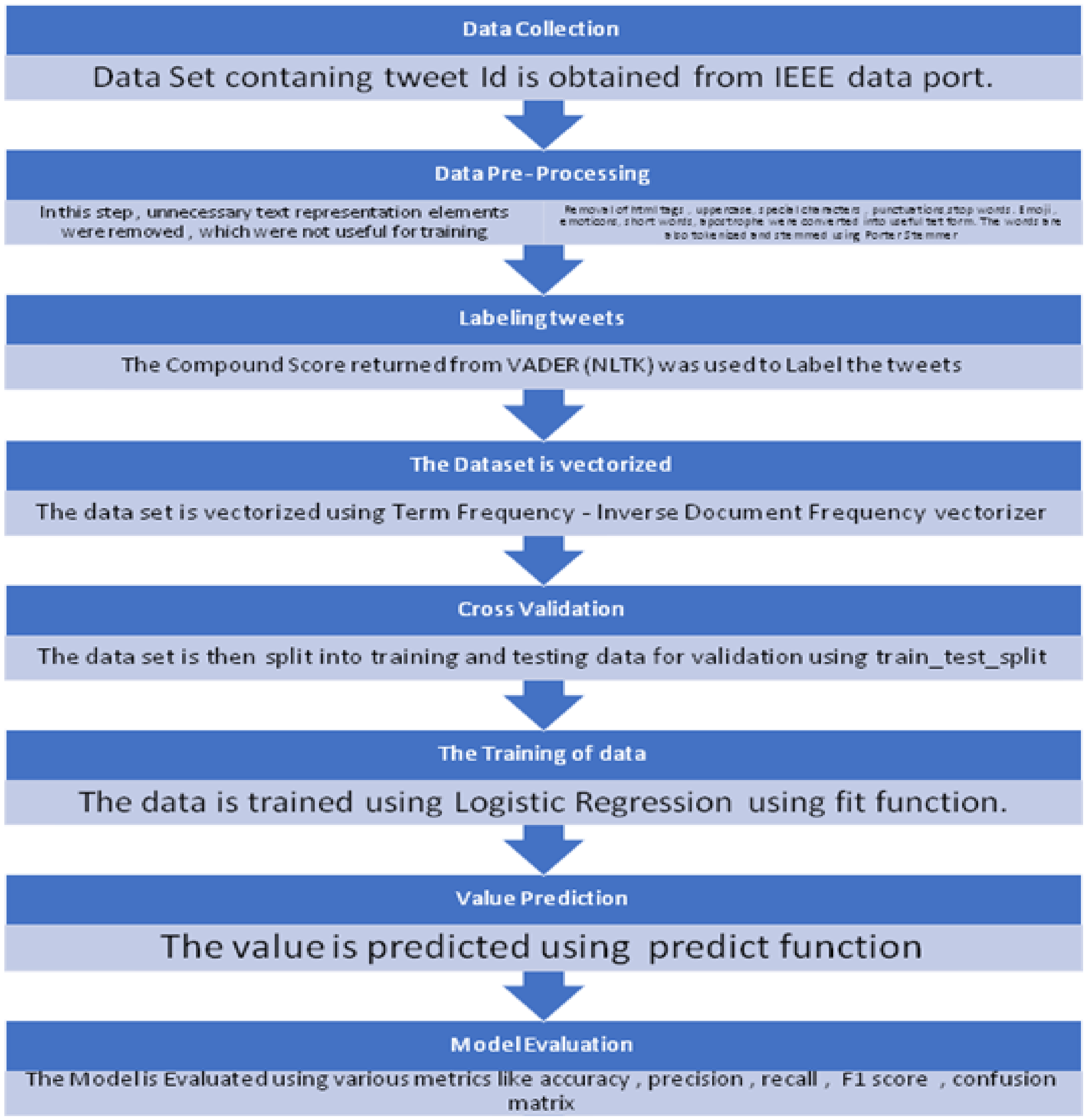
PROPOSED SOLUTION

This model analyses the sentiments of tweets regarding the lockdown in India and using a predictive model, projects the behavior of people in the event of the extension of the lockdown.

The complete dataset is first obtained from Kaggle/IEEE data port and split into training and testing data. The training dataset is to be vectorized and then used to train models such as logistic regression, multinomial Naïve Bayes, K-Nearest Neighbours, random forest classifier. The polarities of the tweets under each model are also checked after which model testing is also done.

The predictive model, once built, can now be used for sentiment classification of tweets in real time by setting up and utilizing a Twitter API.

Project Design



Data Set contaning tweet Id is obtained from IEEE data port.

In this step, unnecessary text representation elements were removed, which were not useful for training

Removal of html tags, uppercase, special characters, punctuations stop words. Emoji, emoticons, short words, apostrophe were converted into useful text form. The words are also tokenized and stemmed using Porter Stemmer

The Compound Score returned from VADER (NLTK) was used to Label the tweets

The data set is vectorized using Term Frequency - Inverse Document Frequency vectorizer

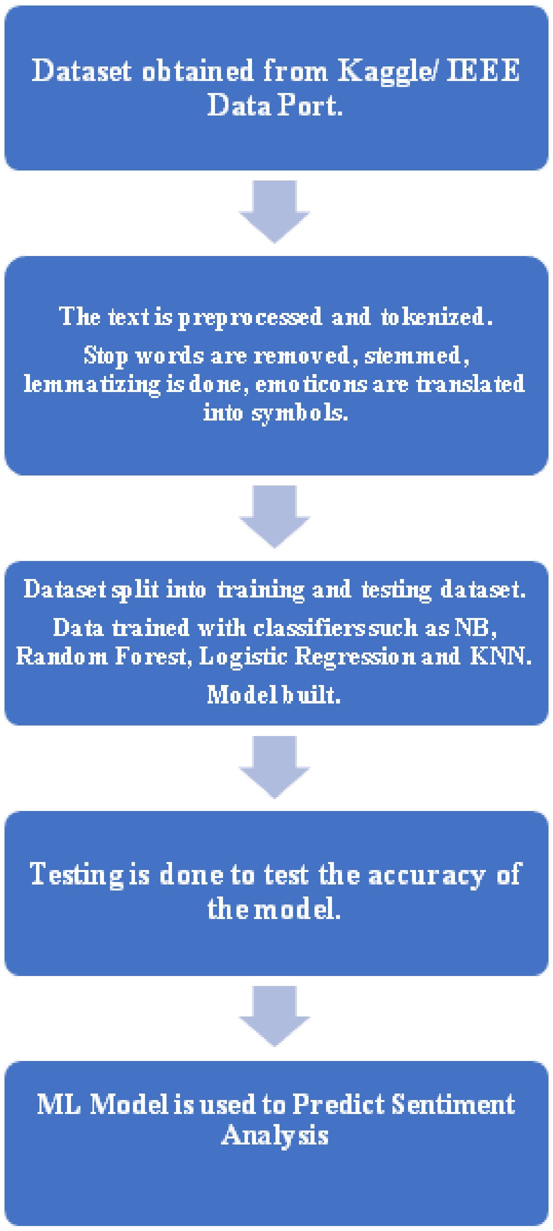
The data set is then split into training and testing data for validation using train_test_split

The data is trained using Logistic Regression using fit function.

The value is predicted using predict function

The Model is Evaluated using various metrics like accuracy, precision, recall, F1 score, confusion matrix

Flowchart


Dataset obtained from Kaggle/ IEEE Data Port.
The text is preprocessed and tokenized.
Stop words are removed, stemmed, lemmatizing is done, emoticons are translated into symbols.
Dataset split into training and testing dataset.
Data trained with classifiers such as NB, Random Forest, Logistic Regression and KNN.
Model built.
Testing is done to test the accuracy of the model.
ML Model is used to Predict Sentiment Analysis

Step followed in each module of project

Task 1: Data collection (From twitter / IEEE data port)

Data was collected from IEEE data port Covid 19 Data Set .The file contained 392,847 tweet id which was hydrated with an app called hydrator Hydrator 329,532 were hydrated with deletion percentage of 16%.It was converted into a csv file containing tweets , retweets , user - id , location , profile etc.Spider/notebook was used to run the code . The csv file was imported using Pandas tool kit.

Task 2: Data Pre - processing

The following steps were involved in pre processing:

1. Removal of Html tags
2. Removal of @ user and converting all to lower space :
3. Replacements

(Replacement of Apostrophe/Short words/emoticons/emoji/punctuation with spaces/Special Characters with spaces)

4. Removing words lesser than length 2
5. Tokenizing the words
6. Removing Stop Words
7. Stemming the words

Task 3: Labeling the data and training

1. Sentiment analysis using VADER

2 .Labeling based on returned values

3. Vectorization of the data using Tf-IDF vectorizer

4. Splitting of the dataset into training and testing data set

5. Training of the dataset using different algorithms

Task 4: Evaluation of the model

Using classification & confusion matrix & time module

Note: source code for the above is available in appendix.

EXPERIMENTAL INVESTIGATIONS

Following are the experimental results obtained after training various models.

○ Logistic Regression (stemmed tweets) with TF-IDF:

■ Accuracy: 94.06 %

■ Run time: 370 s

○Logistic Regression (lemmatized tweets) with TF-IDF:

■ Accuracy: 83.7813 %

■ Run time: 362 s

○Multinomial Naïve Bayes with Hash Vectorizer stemmed tweets:

■ Accuracy: 82.6492 %

■ Run time: 363.6856 s

○ Multinomial Naïve Bayes with TF-IDF stemmed tweets:

■ Accuracy: 81.9743 %

■ Run time: 341.841- s

○ Multinomial Naïve Bayes with Hash Vectorizer lemmatized tweets:

■ Accuracy: 78.675 %

■ Run time: 308 s

○ Multinomial Naïve Bayes with TF-IDF lemmatized tweets:

■ Accuracy: 79.098%

■ Run time: 310 s

○ K-Nearest Neighbours with (stemmed tweets):

■ Accuracy: 81.0043 %

■ Run time: 854.9188 s

RESULT

In this project we have trained and developed a model that analyzes the tweet fed into it as an input and predicts the tone of the tweet .The model is specifically built for the purpose of sentiment analysis of tweets regarding lockdown in India and it is very much helpful in predicting the nature, behavior of people through their tweets. I have analysed the result using different models and found that Logistic Regression (stemmed tweets) with TF-IDF is best in view of accuracy (94.06%) as well as its average of all other models.

APPLICATIONS

The following can be the applications of the developed project:

●Decision-making regarding lockdown extension/removal.
● Identifying issues regarding the lockdown and coming up with solutions for the same.
● Identifying correlations between economy and the public's opinion/ shift in opinion.
●Gain valuable insight into the society.
● Identification of health/ mental issues.
●Identify the impact the lockdown has had on the Indian economy, society and the political scenario.

ADVANTAGES AND DISADVANTAGES

a)ADVANTAGES

- No manual collection of data and identification of sentiments, very easy way to classify sentiments.
- Sentiment analysis of millions of tweets done within a short span of time.
- Algorithm with high accuracy, low runtime and which chooses low number of features chosen.
- Prediction of the behaviour of the citizen regarding lockdown extension can be done, which may become influential to decision-making regarding the lockdown.
- Cost-effective and time saving.

b. DISADVANTAGES

- Uses content only from Twitter, not any other social media.
- Only a limited amount of result visualizations are available.

CONCLUSION

The task of sentiment analysis, especially in the domain of micro-blogging, is still in the developing stage and far from complete. I made sentiment analysis using messages people post on twitter. Twitter is a source of vast unstructured and noisy data sets that can be processed to locate interesting patterns and trends. This data analysis makes it possible for business organizations to keep track of their services and generates opportunities to promote, advertise and improve from time to time. We have experimented with the results after training the model with various models and classifiers. We look forward to use bigger dataset to improve the accuracy, considering the emoticons and
emoji .

FUTURE SCOPE

Artificial intelligence and data science are going to be very influential factors in the future. With that being said, this project has immense scope for development in the future.

The project follows a scalable model, which uses an algorithm that has a low training and run time and uses up a low number of features, while possessing a high accuracy of classification.

Hence, the model can be scaled up to work with larger datasets and even process content from multiple social media platforms to produce a more detailed result of the public's sentiment with regard to the lockdown.

The project can also be scoped up with using multiple visualization outputs such as tag clouds, word clouds, heat maps and so on.

BIBLIOGRAPHY

DATA COLLECTION:

https://www.kaggle.com/smid80/coronavirus-covid19-tweets

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

VECTORIZATION:

https://towardsdatascience.com/natural-language-processing-count-vectorization-with-scikit-learn-e7804269bb5e

https://monkeylearn.com/blog/what-is-tf-idf/

https://ieeexplore.ieee.org/document/77135

DATA PRE PROCESSING:

https://towardsdatascience.com/nlp-text-preprocessing-a-practical-guide-and-template-d80874676e79

https://medium.com/analytics-vidhya/text-preprocessing-for-nlp-natural-language-processing-beginners-to-master-fd8 2dfecf95

DATA TRAINING AND TESTING:

https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/ https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

https://scikit-learn.org/stable/modules/tree.html

https://www.kdnuggets.com/2019/01/solve-90-nlp-problems-step-by-step-guide.html

DATA INTERPRETATION:

https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

https://towardsdatascience.com/word2vec-made-easy-139a31a4b8ae

https://github.com/marcotcr/lime

Source code

. **Task 1 : Data collection (From twitter / IEEE data port)**

Data was collected from IEEE data port Covid 19 Data Set / Real time data from the twitter was collected .

```
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from textblob import TextBlob
import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)
data_df = pd.read_csv("ready_corona_tweets1_30")
```

Task 2 : Data Pre - processing

1. Removal of Html tags:

2. Removal of @ user and converting all to lower space :

```
def remove_pattern(input_text, pattern):
    r= re.findall(pattern, input_text)
    for i in r:
        input_text = re.sub(i,'', input_text)
    return input_text 6

data_df['clean_text'] =
np.vectorize(remove_pattern)(data_df['clean_text'],"@[¥w]*"
)
data_df['clean_text'] = data_df['clean_text'].apply(lambda x:
x.lower())
```

3. Replacements

a. Function for Replacement :

```
def lookup_dict(text, dictionary):
    for word in text.split():
        if word.lower() in dictionary:
            if word.lower() in text.split():
                text = text.replace(word, dictionary[word.lower()])
    return text
```

b . Replacement of Apostrophe
Link to the apos_dict file : Apos_dict file

```
# Replacing Apostrophe
import pickle

pickle_in = open(″apos_dict.pickle″,″rb″)
apostrophe_dict = pickle.load(pickle_in)
data_df['clean_text'] = data_df['clean_text'].apply(lambda
x: lookup_dict(x,short_word_dict))
```

c. Replacement of Short words
Link to the short_dict file : short_dict file

```
# converting abbreivations
pickle_in = open(″short_dict.pickle″,″rb″)
short_word_dict = pickle.load(pickle_in)

data_df['clean_text'] = data_df['clean_text'].apply(lambda
x: lookup_dict(x,short_word_dict))
```

d. Replacement of emoticons
Link to the emot_dict file: emot_dict file

```
pickle_in = open(″emot_dict.pickle″,″rb″)
emoticon_dict = pickle.load(pickle_in)
data_df['clean_text'] = data_df['clean_text'].apply(lambda x:
lookup_dict(x,emoticon_dict))
```

e. Function for emoji replacement:

```
import emoji
#def extract_emojis(s):
 # return ''.join(c for c in s if c in emoji.UNICODE_EMOJI)
def rep_emoji(tweet):
    tweet = emoji.demojize(tweet)
    tweet = tweet.replace(":" , " ")
    tweet=' '.join(tweet.split())
    return tweet
```

f. Replacement of emoji's :

```
data_df['clean_text'] = data_df['clean_text'].apply(lambda x:
  lookup_dict(x,emoticon_dict))
```

g. Replacing punctuation with spaces :

```
data_df['clean_text'] = data_df['clean_text'].apply(lambda x:
  re.sub(r'[^¥w¥s]',' ',x))
```

h. Replacing Special Characters with spaces :

```
data_df['clean_text'] = data_df['clean_text'].apply(lambda x:
re.sub(r'[^a-zA-Z0-9]',' ',x))
data_df['clean_text'] = data_df['clean_text'].apply(lambda x:
re.sub(r'[^a-zA-Z]',' ',x))
```

i. Removing words lesser than length 2 :

```
data_df['clean_text'] = data_df['clean_text'].apply(lambda x: ' '.join([w
  for w in x.split() if len(w)>2]))
```

j. Tokenizing the words :

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize from
nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.sentiment.vader import SentimentIntensityAnalyzer from
wordcloud import WordCloud
from textblob import TextBlob

#word tokenizing
data_df['tokenized_tweet']=
data_df['clean_text'].apply(lambda x: word_tokenize(x))
```

k. Removing Stop Words

```
stop_words = set(stopwords.words('english'))
#remove stopwords
data_df['tweet_token_filter'] =
data_df['tokenized_tweet'].apply(lambda x: [word for word in x if
not word in stop_words])
```

l. Stemming the words :

```
stemming = PorterStemmer()
data_df['tweet_stemmed'] =
data_df['tweet_token_filter'].apply(lambda x:
''.join([stemming.stem(i) for i in x]))
```

3. <u>Labeling the data and training</u>

a. Sentiment analysis using VADER:

```

sid = SentimentIntensityAnalyzer()
data_df['sentiment_stemmed'] =
data_df['tweet_stemmed'].apply(lambda x:
sid.polarity_scores(x))
```

```

def convert(x):
      if x < -0.05:
            return 0
      elif -0.05 < x < 0.05:
            return 1
      else :
```

b. Labeling based on returned values:

```
data_df['label_stemmed'] =
data_df['sentiment_stemmed'].apply(lambda x:
convert(x['compound']))
data_df.to_csv('final_corona_tweets.csv')
#data_df = pd.read_csv("final_corona_tweets.csv",
lineterminator='¥n')

from sklearn.neighbors import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix
```

c. Vectorization of the data using Tf-IDF vectorizer

```
X= data_df['tweet_stemmed'].fillna(' ')
tfidf_vectorizer =
TfidfVectorizer(max_df=0.90,min_df=2,max_features=1000,stop_
words='english')
tfidf_stem = tfidf_vectorizer.fit_transform(X) y=
data_df['label_stemmed']
print("data vectorized")

Vectorizing_time = time.time()
```

```
print("Vectorizing_time :",Vectorizing_time -start_time)
```

d. Splitting of the dataset into training and testing data set

```
train_tfidf = tfidf_stem[:319685, :]
test_tfidf    = tfidf_stem[319685:,:]
x_train, x_test , y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2)

x_train = train_tfidf[y_train.index]

x_test= train_tfidf[y_test.index]

print("data split properly")
```

e.Training of the dataset using Logistic Regression

```
lreg = LogisticRegression(solver='lbfgs',class=multinomial,max_iter=1000)
lreg.fit(x_train , y_train)


print("data training time:", time.time()-start_time)
```

4. Prediction of y values and testing the accuracy of the data

```
prediction = lreg.predict(x_test)
#prediction_int = prediction[:,1] >= 0.3
#prediction_int = prediction_int.astype(np.int)
print(confusion_matrix(y_test , prediction , labels = [0,1,2]))
```

s.Evaluation of the model

```
print(classification_report(y_test, prediction , labels=[0,1,2],target_names=['negative','neutral','positive']))
print(f1_score(y_test, prediction,average ='macro')) #
 calculating f1 score
end_time = time.time()
run_time = end_time - start_time
print("run_time:", run_time)
```

```
   print(confusion_matrix(y_test , prediction , labels = [0,1,2]))
   print(classification_report(y_test,      prediction      ,
   labels=[0,1,2],target_names=['negative'
   ,'neutral','positive']))
    # calculating f1 score
print(f1_score(y_test, prediction, average='macro'))
   end_time = time.time()
   run_time   =   end_time   -   start_time
print("run_time:",run_time)
```

The Evaluation Results

	Predicted Negative	**Predicted Neutral**	**Predicted Positive**
Reference Negative	19625	950	555
Reference Neutral	289	20110	324
Reference Positive	510	967	20597

	Precision	**recall**	**f1-score**	**support**
Negative	0.96	0.93	0.94	21140
Neutral	0.91	0.97	0.94	20723
Positive	0.96	0.93	0.95	22074
Accuracy			0.94	63937
macro avg	0.94	0.94	0.94	63937
weighted avg	0.94	0.94	0.94	63937

THANKS!!