Sentiment Analysis of COVID-19 Tweets

Team Members:

- ➤ V. Maheysh
- ➤ Amudhini P. K.
- ➤ Divya Shree
- ➤ Shalini Annadurai

Schedule:

1) Pre-Preparation - *Completed*
2) Tweepy Working And Analysis - *Completed*
3) Data Pre-Processing - *Completed*
4) Applying ML Algorithms - *Completed*
5) Application Of Our Project - *Completed*
6) Application Of User Interface - *Completed*
7) Post Processing(Additional Ideas/Conclusion) - *Completed*

Our Github Link:

https://github.com/SmartPracticeschool/SBSPS-Challenge-1315-Sentiment-Analysis-of-Tweets

1)Pre-Preparation

TSK-1574 -- Deciding The Scope Of Our Project

Our aim is to do sentimental analysis of tweets to identify the feelings of people. We decided the scope of the project based on how it is going to help people and government and how they will benefit. We determined the list of goals, features, function, tasks, deadlines, setting success criteria and identifying major risks in our project and also decided in what way our project is going to help people and government.

TSK-1576 -- Bootcamp And External Research

We attended the Bootcamp offered by IBM to get familiar with IBM cloud and its services and the knowledge to use the tools wisely. We learned new ideas from the Bootcamp which included easy analysis and also helped in developing our project. Also, we gathered new ideas with the help of google and youtube to get ideas about the project.

TSK-1579 -- Creating GitHub And Slack Accounts

All the team members present in the group have created their github accounts.https://github.com/SmartPracticeschool/SBSPS-Challenge-1315-Sentiment-Analysis-of-Tweets

We created a slack workspace to communicate regarding the project.

Slack workspace url: team-bytebiters.slack.com

TSK-1580 -- Final Plan Of Our Project

The aim of this project is to give the user a vague idea about the sentiments and emotions prevailing in the place that they wish to find during this lock down. We will be using Twitter API to get the tweets related to this pandemic. Using these tweets we will be analyzing the emotions of the people by performing sentimental analysis on the data set. We'll be analyzing the number of times a particular keyword is used and its nature by performing basic ML operations and sentimental analysis. Also, the user can input a state, and the top keywords used in the locality are displayed using a pie chart.

The project serves as a tool to analyze the emotions of people in the lock down and the emotional quotient in that particular place.

2) Tweepy Working And Analysis

TSK-1584 -- Getting The Access And The Consumer Keys From Tweepy

The access and consumer keys are obtained from developer.twitter.com/en/apps by creating an app. We need a twitter account to do the same. Under Keys and tokens the following is required for accessing the tweets. The four keys are access_code, access_secret, consumer_code and consumer_secret. These keys are stored in a python file which acts as a header file to the other python file where the tweets are filtered.

TSK-1585 -- Filtering Out The Tweets Using Tweepy Header Files

The tweets are listened by tweepy.StreamListener and streamed with tweepy.stream. The streamed tweets are filtered with the help of tweepy.filter tracking only the keywords 'corona', 'covid-19', 'lockdown'.

TSK-1586 -- Storage Of Filtered Keywords

The filter tweets in which their location is enabled are formatted and stored. The details which is stored are:-

- User ID
- Created Date and Time
- Country Code
- State
- Latitude
- Longitude
- Tweet text
- Sentiment Compound

- ➤ Overall Sentiment

3) Data Pre-Processing

TSK-1588 -- Tokenization

Tokenization is the process of breaking up a sequence of strings into pieces called tokens.In our project we have performed tokenization using the nltk package supported by Anaconda cloud.We have performed tokenization on the tweets after filtering them out using tweepy based on keywords pertaining to COVID -19 .

Snippets of this task from our project:

```
import nltk

from nltk.stem.porter import PorterStemmer

porter=PorterStemmer()

def tokenizer_porter(text):

    return[porter.stem(word) for word in text]
```

TSK-1589 -- Removing the stop words and Unwanted special characters

In this task we have cleaned the data by removing the stopwords,numeric values and special characters which will ensure correct, consistent and useable data for analysis.The clean data is fed into our machine learning model which will perform the sentimental analysis.the removal of stop words is facilitated by an nltk package .

Snippets of this task from our project

```
import nltk

from nltk.corpus import stopwords

def remove(text):

  stop_words = stopwords.words('english')

  return([w for w in text if not w in stop_words and not
   w.isnumeric()])
```

4)Applying ML Algorithms

TSK-1590 -- Codification Of ML Algorithms In Watson Studio Notebook

Codification to create the sentiment analysis module called sentiment_mod was done.This module classifies the sentiment of a given tweet and gives it its respective confidence level.

The following snippet shows how confidence is calculated.

```
class VoteClassifier(ClassifierI):

    def __init__(self, *classifiers):

        self._classifiers = classifiers



    def classify(self, features):

        votes = []

        for c in self._classifiers:

            v = c.classify(features)

            votes.append(v)

        return mode(votes)
    def confidence(self, features):

        votes = []

        for c in self._classifiers:

            v = c.classify(features)

            votes.append(v)



        choice_votes = votes.count(mode(votes))

        conf = choice_votes / len(votes)

        return conf
```

The following import shows the models being used to calculate sentiment.

```
from sklearn.naive_bayes import MultinomialNB, BernoulliNB

from sklearn.linear_model import LogisticRegression, SGDClassifier

from sklearn.svm import SVC, LinearSVC, NuSVC
```

TSK-1591 --Verification With AutoAI

Auto AI was experimented with by understanding its use and applying our datasets for an assessment.

5)Application Of Our Project

TSK-1592 -- Line chart on the basis of polarity of tweets

A line chart is used to graph real-time the sentiments as predicted by our Machine Learning model.

A snippet of the code is shown below:

```
conn = sqlite3.connect('twitter.db')

c = conn.cursor()



df = pd.read_sql("SELECT * FROM sentiment WHERE tweet LIKE '%covid%'
   ORDER BY unix DESC LIMIT 1000", conn)



df.sort_values('unix', inplace=True)



df['sentiment_smoothed'] =
   df['sentiment'].rolling(int(len(df)/5)).mean()



df['date'] = pd.to_datetime(df['unix'],unit='ms')

df.set_index('date', inplace=True)

df = df_resample_sizes(df)

df.dropna(inplace=True)



X = df.index

Y = df.sentiment_smoothed



data = go.Scatter(x=X, y=Y, name='Scatter', mode= 'lines+markers',)



figure= {'data': [data],
   'layout':go.Layout(xaxis=dict(range=[min(X),max(X)],title='Time'),
   yaxis=dict(range=[min(Y),max(Y)],title='Sentiment'),
   font={'color':'#FFFFFF'}, plot_bgcolor = '#0C0F0A',
   paper_bgcolor='#0C0F0A', )}
```

TSK-1593 --- Geographic Mapping

Based on the overall sentiment in each tweet, geographical distribution of the tweets are displayed. Two type of maps are made:

1. Orthographic map:

Tweets around the world are displayed here.

```
data=[go.Scattergeo(

            lat=df1[df1['Overall_sentiment']=='Positive']['Latitude'],

            lon=df1[df1['Overall_sentiment']=='Positive']['Longitude'],

            mode='markers',

            marker_color='rgb(128, 255, 0)',

            marker_size=3,

            name='Positive'),

     go.Scattergeo(

            lat=df1[df1['Overall_sentiment']=='Negative']['Latitude'],

            lon=df1[df1['Overall_sentiment']=='Negative']['Longitude'],

            mode='markers',

            marker_color='rgb(255, 51, 51)',

            marker_size=3,

            name='Negative'),

    go.Scattergeo(

            lat=df1[df1['Overall_sentiment']=='Neutral']['Latitude'],

            lon=df1[df1['Overall_sentiment']=='Neutral']['Longitude'],

            mode='markers',

            marker_color='rgb(174, 179, 179)',

            marker_size=3,

            name='Neutral')]
```

```
layout =go.Layout(template='plotly_dark',
        geo=go.layout.Geo(
            showland = True,
            showcountries = True,
            showocean = True,
            countrywidth = 0.5,
            landcolor = 'rgb(51, 102, 0)',
            oceancolor = 'rgb(0, 0, 102)',
            projection_type='orthographic',
            center_lon=50,
            center_lat=0,
            projection_rotation_lon=50
        ))
    lon_range = np.arange(-180, 180, 2)

frames =[go.Frame(layout=go.Layout(geo_center_lon=lon,
               geo_projection_rotation_lon =lon,
               geo_center_lat=0,geo_projection_rotation_lat=0),
                  name =f'{k+1}') for k, lon in enumerate(lon_range)]

sliders = [dict(steps = [dict(method= 'animate',args= [[f'{k+1}'],
                              dict(mode= 'immediate',
                               frame= dict(duration=0, redraw= True),
                               transition=dict(duration= 0))],
                       label=f'{k+1}') for k in range(len(lon_range))],
                transition= dict(duration= 0 ),
                x=0,
                y=0,

                len=1.0) #slider length

              ]



fig = go.Figure(data=data, layout=layout, frames=frames)

fig.update_layout(sliders=sliders)
```

2. Street-view map:

User is provided with an option to view either the distribution only in India or the world.

```
fig = px.scatter_mapbox(df1, lat="Latitude", lon="Longitude",

                               color="Overall_sentiment",

        hover_data=["Sentiment_compound","Created_at"],

        zoom=z,template='plotly_dark',

        color_discrete_map={'Positive':'rgb(128, 255, 0)',

                               'Negative':'rgb(255, 51, 51)',

                               'Neutral':'rgb(174, 179, 179)'})
```

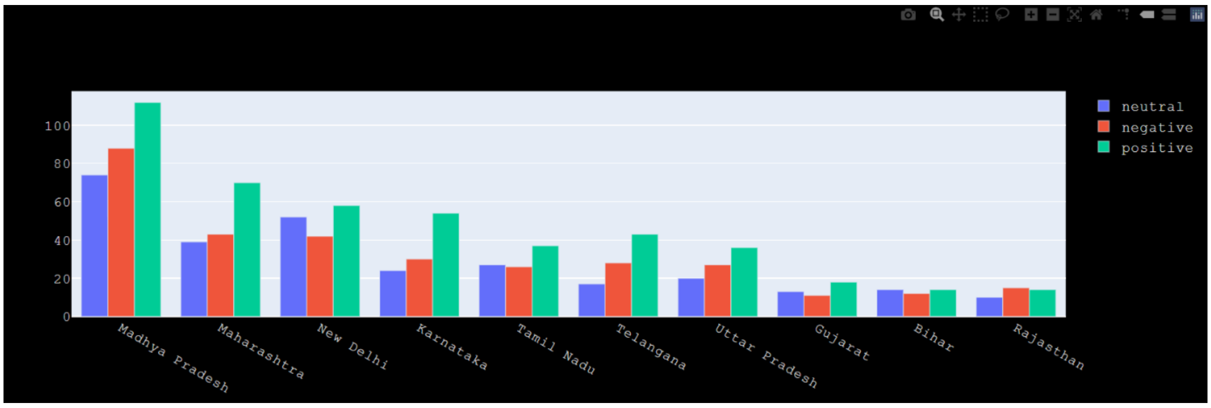
```
fig.update_layout(mapbox_style="dark",

                  mapbox_accesstoken=plotly_token)

fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
```

TSK-1594 --- Tracking the most frequent words in a state

Tracking the most frequent words in a state facilitates the analyzing of the sentiments of people in a particular state.There are 3 sub-tasks under this task and are:

1.Sentiment analysis of the top 10 states in our country

This is to show the user about the sentiments of people in top 10 states of our country during the pandemic and is presented in the form of a graph to make it user friendly.



```
df1=df[df['Country_code']=='IN']



a=pd.DataFrame()

a['State']=df1['State'].unique()



a['total_count']=a['State'].apply(lambda x: len(df1[df1['State']==x]))
```

```
a['positive_count']=a['State'].apply(lambda x:
   len(df1[df1['State']==x]['State'][df1[df1['State']==x]['Overall_senti
   ment']=='Positive']))



a['negative_count']=a['State'].apply(lambda x:
   len(df1[df1['State']==x]['State'][df1[df1['State']==x]['Overall_senti
   ment']=='Negative']))



a['neutral_count']=a['State'].apply(lambda x:
   len(df1[df1['State']==x]['State'][df1[df1['State']==x]['Overall_senti
   ment']=='Neutral']))



a.sort_values(by=['total_count'], inplace=True,ascending=False)



trace3 = go.Bar(x =a.State.head(10), y = a.positive_count.head(10),
   name='positive')

trace2 = go.Bar(x= a.State.head(10), y = a.negative_count.head(10), name
   ='negative')

trace1 = go.Bar(x = a.State.head(10), y = a.neutral_count.head(10), name
   ='neutral' )

data = [trace1, trace2, trace3]



layout = go.Layout(barmode = 'group',paper_bgcolor="Black",

               font=dict(family="Nunito Sans",size=15,color="#DDDDDD"))



fig= go.Figure(data = data, layout = layout)
```

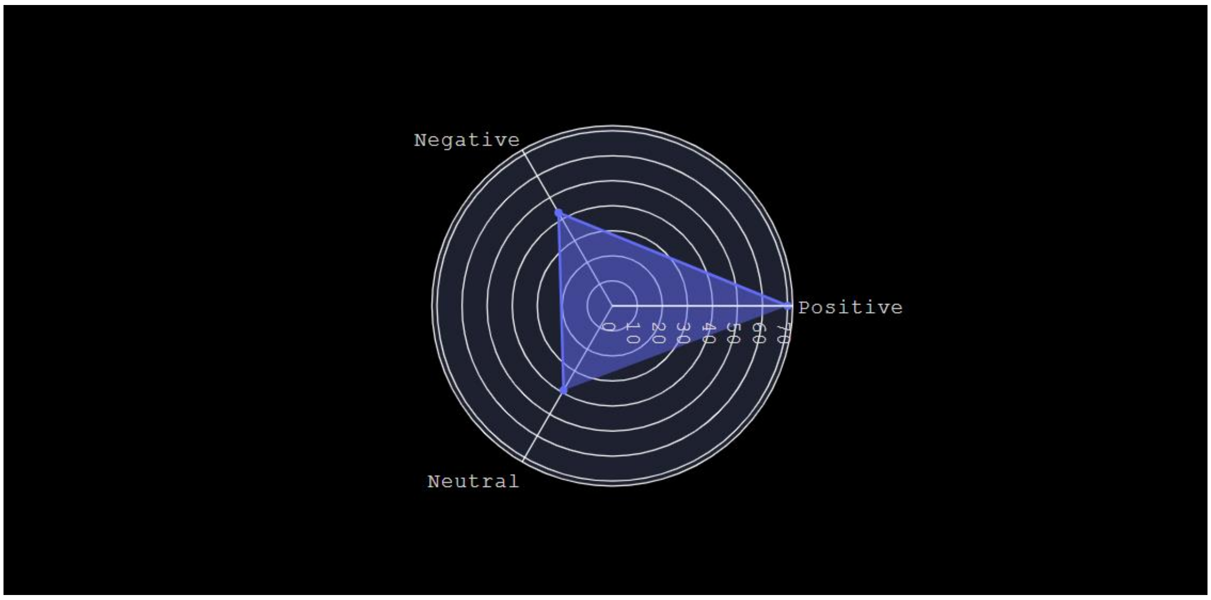
2.Sentiment analysis of the state that the user wants to find out

This feature will help the user to find out the sentiments of people in this state.The following process is done by plotting a graph of the top words used and the context in which it has been used and is represented in the form of a bar graph

```
df1=df[df['State']==m]



f=return_tweet_words(df1)

sort_orders = sorted(f.items(), key=lambda x: x[1],
   reverse=True)    x=[];y=[]

for j in range(0,10):

   x.append(sort_orders[j][0])

   y.append(sort_orders[j][1])



y1=[];y2=[];y3=[];x1=[]



for k in range(0,7):

   p=0;n=0;l=0

      for j in df1.index:

        if df1['Overall_sentiment'][j]=='Positive' and
   str(df1['Tweet_text'][j]).lower().count(x[k])>0:

                p+=str(df1['Tweet_text'][j]).lower().count(x[k])



        elif df1['Overall_sentiment'][j]=='Negative'and
   str(df1['Tweet_text'][j]).lower().count(x[k])>0:

                n+=str(df1['Tweet_text'][j]).lower().count(x[k])



        elif df1['Overall_sentiment'][j]=='Neutral'and
   str(df1['Tweet_text'][j]).lower().count(x[k])>0:

                l+=str(df1['Tweet_text'][j]).lower().count(x[k])



   if(p>0 or n>0 or l>0):

            y1.append(p)

            y2.append(n)

            y3.append(l)
            x1.append(x[k])



fig = go.Figure(go.Bar(x=x1, y=y3, name='Neutral'))

fig.add_trace(go.Bar(x=x1, y=y2, name='Negative'))

fig.add_trace(go.Bar(x=x1, y=y1,
   name='Positive'))    fig.update_layout(barmode='stack')

fig.update_layout(xaxis_title="Frequent Words Used",
   yaxis_title="Count",

                      font=dict(family="Nunito Sans", size=15,

                       color="#DDDDDD"),paper_bgcolor="Black")
```

3.The overview of the sentiments in the state

The overall sentiments of the people in the state is represented in the form of a gauge chart containing the percentage of positive,negative and neutral sentiments in the particular state.



```
  p=len(df1[df1['Overall_sentiment']=='Positive'])

 n=len(df1[df1['Overall_sentiment']=='Negative'])

 l=len(df1[df1['Overall_sentiment']=='Neutral'])
```

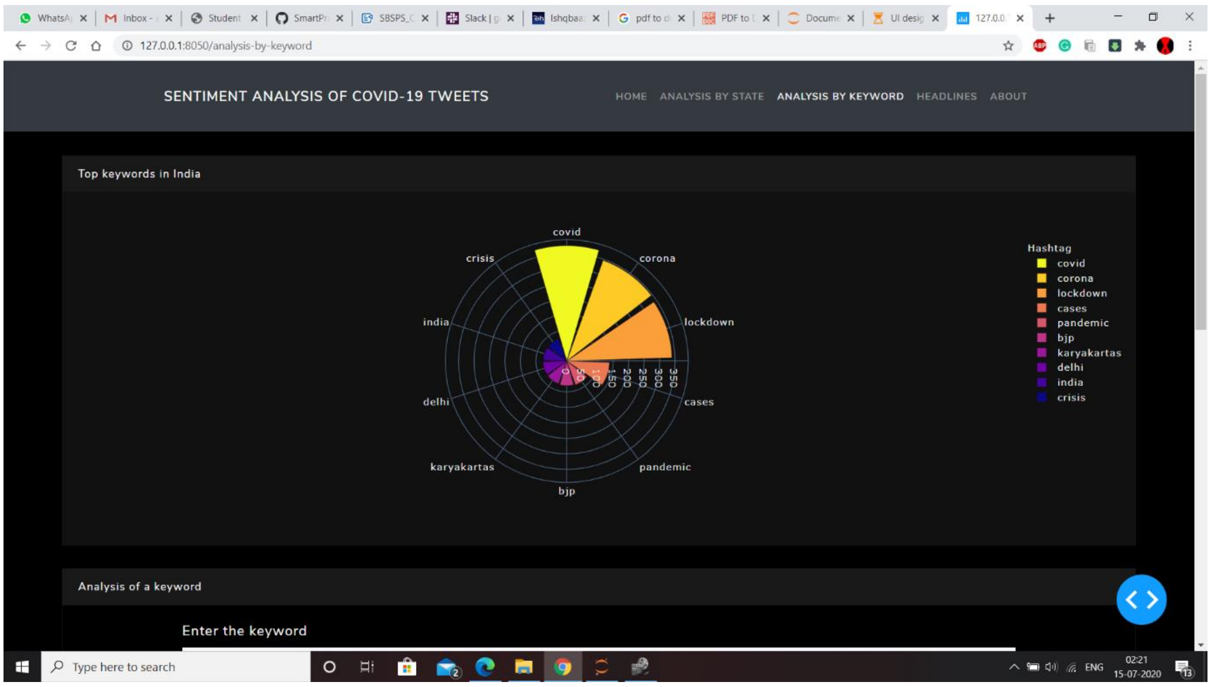
```
fig1 = go.Figure(data=go.Scatterpolar(

r=[p,n,l],theta=['Positive','Negative','Neutral'],

fill='toself')

    fig1.update_layout(paper_bgcolor="Black",font=dict(family="Nunito
   Sans",size=15,color="#DDDDDD"),

polar=dict(bgcolor='#1e2130',

radialaxis=dict(visible=True,)),showlegend=False)



fig1.update_layout(title_text=

'Sentiment polarity of the state', title_x=0.5)
```

TSK -1595 --- Analysis by Keywords

Search by keyword help us to analyze the occurrence of words in the tweets of India. This process include two tasks

1.Analyzing the overall occurrence of words in the tweets :

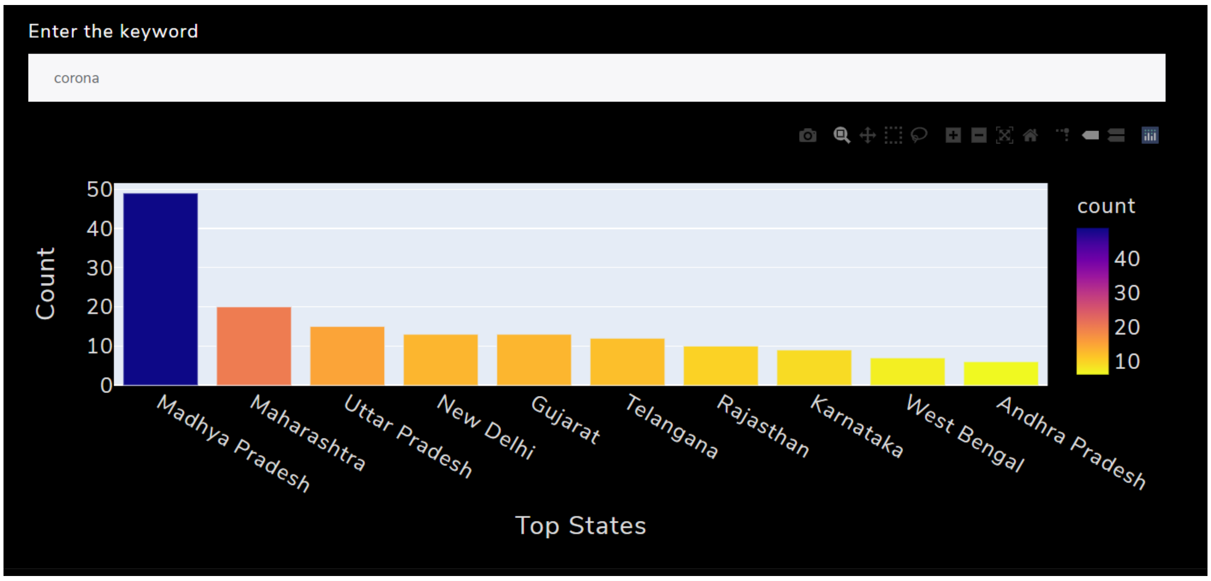
This process shows the top 10 words occurred in the tweets and it is plotted as a polar bar chart which helps the user to understand the process easily.



```
fd =return_tweet_words(df[df["Country_code"]=='IN'])

d = pd.DataFrame({'Hashtag': list(fd.keys()),

                  'Count' : list(fd.values())})

d = d.nlargest(columns = 'Count', n = 10)

fig= px.bar_polar(d, r='Count', theta='Hashtag',

              hover_data=['Hashtag','Count'],

              color='Hashtag', template="plotly_dark",

                color_discrete_sequence=
   px.colors.sequential.Plasma_r)
```

2.Top 10 states that uses the word given by the user:

When the user input a word, this task help the user to get the top five states that uses the word given by the user and it is plotted as a bar chart which helps the user to understand the task in a better manner.


Enter the keyword

corona

```
def search(t):

   d1=str(t).count(h.lower())

   return d1



df1=df[df["Country_code"]=='IN']

df1["count"]=df1["Tweet_text"].apply(search)

df2=df1[df1['count']>0]

    df3=df2.drop(['User_Id','Created_at','Country_code','Latitude','Lo
   ngitude','Tweet_text', 'Sentiment_compound',
   'Overall_sentiment'],axis=1)

df3=df3.groupby('State').sum()

df3.sort_values(by=['count'], inplace=True,ascending=False)

df3=df3.rename_axis('index').reset_index()

df3.rename(columns={"index":"state"})



fig5=px.bar(df3.head(10),x='index',y='count',

color='count',color_continuous_scale=px.colors.sequential.Plasma_r,

                labels={'pop':'count'},height=400)
```

6)Application Of User Interface

TSK-1596-User interface using dash:

Dash is Python framework for building web applications. It built on top of Flask, Plotly. js, React and React Js. It enables you to build dashboards using pure Python. Dash is open source, and its apps run on the web browser. Using dash we have created the user interface that displays the various features of our project.

7)Post Processing(Additional Ideas/Conclusion)

TSK -1597 --- Tinkering With Additional Ideas

After ensuring that the core components of our projects are completed, we brainstormed to make it even better. A nav-based design was added to separate the contents of our project accordingly in Dash. Representative figures such as word-clouds and bar graphs were also included to support the respective idea. A headlines section will be added to provide hyperlinks to official news sources. An about section felt necessary to explain our project and its components to our end-users.

TSK -1597 --- Working Model

The model which works locally now needs to be deployed using the required cloud software. Github updates will be made at this point.

TSK -1597 --- Conclusion And Submission

After all the previous steps are accomplished, the project will be submitted.