

COVID ANALYZER

1.INTRODUCTION

1.1 Overview:

The Covid Analyzer is a one stop dock where people's emotions are brought into limelight by gathering their tweets and thus analysing their sentiments; taking into consideration each individuals perspective.

1.2 Purpose:

Covid 19 has created a major impact on our day to day lives. Extension of lockdown has caused a variety of mixed emotions which are released in a common hub. People feel free to emote here, which results in the boom of social media platforms such as Twitter, Instagram , Facebook etc. The goal of "The Covid Analyzer" is to understand their reactions towards this pandemic.

2.LITERATURE SURVEY

2.1 Existing Problem:

Ever since the pandemic started, people have been thorough in expressing their emotions via various media such as blogs or statuses or tweets, through which people try to share their undergoings. Need of the hour is that people would like to know the varied reactions of everyone and also share their combined sentiments.

2.2 Proposed Solution:

The way people express themselves during lockdown and the way they react to the Covid pandemic is very well captured by their feeds. A machine learning model which uses NLP(Natural Language Processing) to enact these sentiments in the stage, is a well reaching solution to this pandemic. This solution helps people analyse each individual's views pertaining to not only the lockdown, but also about the generalised views of Covid as such. Clustering is inbuilt to show the different emotions expressed by people with respect to a single domain. Each country's individual response is also captured and data analysis based on the language is determined. The approach towards the analysis is first supervised and later made as a hybrid terrain. The comments, likes for each post, verified accounts , followers count and even the no. of

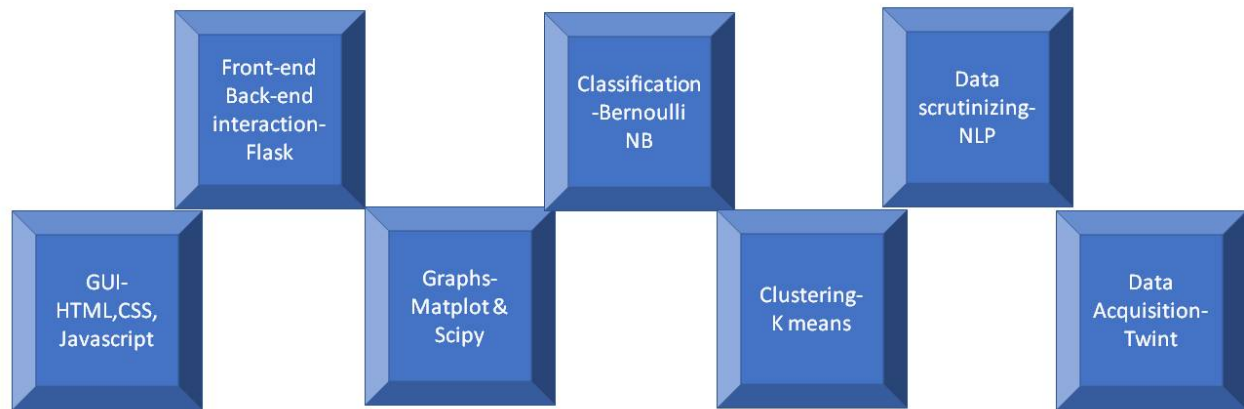
favourites of each comment is analysed and the model is trained respectively. Ultimately this model is trained to make people aware about the effects, Covid 19 has caused and the mixed reactions each individual has towards it, in different countries.

3.THEORETICAL ANALYSIS

3.1 Block Diagram:



3.2 Hardware/Software Designing:



4.EXPERIMENTAL INVESTIGATION

Steps involved in experimental investigation are as follows:

Step 1: Project Idea

With all the chaos generated due to the pandemic , having a clear cut idea about what is happening in the world during this time is vital. As a step towards achieving it , we have created a website that can effectively make people get further insights about all the happenings.

Step 2: Background Research

In order to familiarize people about the effects of the pandemic and about their responses and views, social networking sites are of great help. By the technique of web scraping , data is obtained from these sites which helps in deriving numerous analytical conclusions.

Step 3: Hypothesis

Creating a dashboard with varied content that helps people understand

each other's mindset and thoughts.

Step 4: Design of experiment

Based on the data acquired , an optimal way is chosen to group the data to convey the clear information .

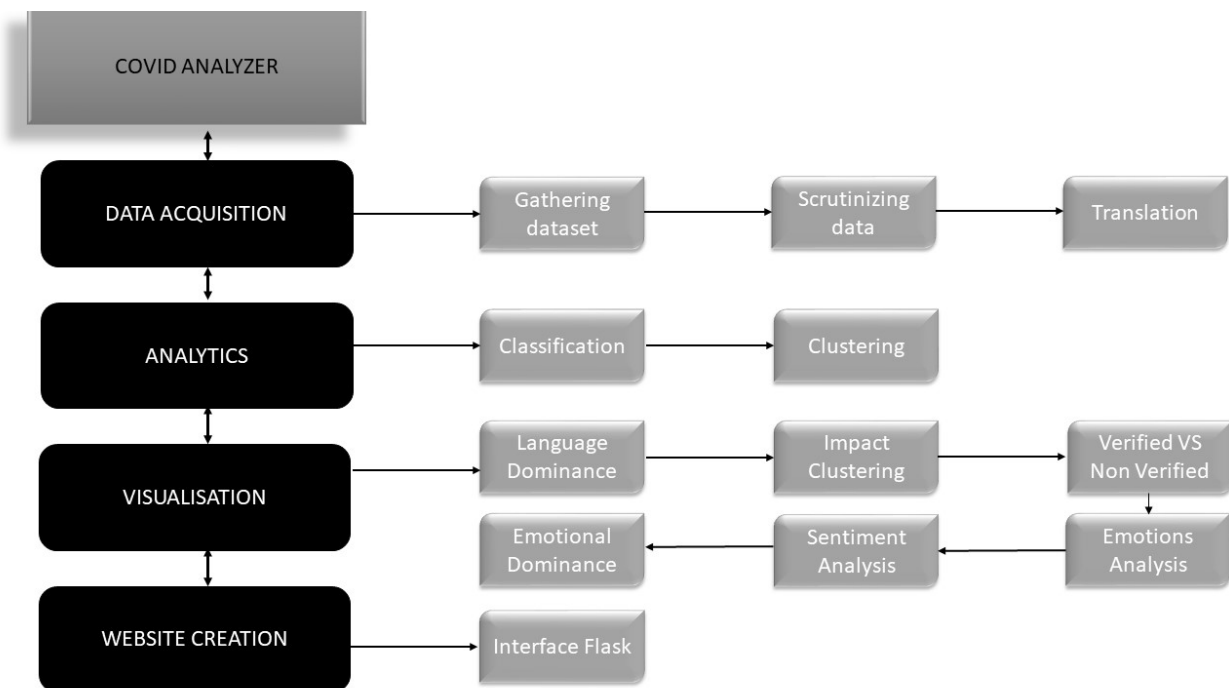
Step 5: Data Collection

Using web scraping and libraries like 'twint' , data from various backgrounds is gathered.

Step 6: Data Analysis and Conclusion

Analysis of data is made by employing various machine learning models based on classification and clustering. The data is visualized as donuts,histograms,etc using libraries like matplotlib.pyplot and scipy.interpolate.

5.FLOWCHART



6.RESULT

With "Covid Analyzer" , information about the existing pandemic can reach people in a better and an effective way.

7.ADVANTAGES AND DISADVANTAGES

By creating a dynamic website, timely information about the pandemic can be made facile. Further, the analysis of the thoughts through a website could make the information reach a wider range of people. With all the positivity depicted by the analysis , people could see through the good things that are still around which is quite necessary at this point of time.

Data acquisition is a challenge due to the restrictions imposed by sites like Twitter. A limit that is set on the amount of data that can be scraped makes it difficult to run complex programs that involves machine learning.

8.APPLICATIONS

1. Instilling a positive outlook towards the deadly virus
2. Changing the perspective of a common man towards this pandemic
3. Data Visualization

9.CONCLUSION

The notion of this project is to make people cognizant about the existing pandemic in every possible way so that they become more resilient. With "Covid Analyzer" in the market , this was made possible .

10.FUTURE SCOPE

In the positive side , a future without this website would be a happier option to all of us. With all the happenings due to the Coronavirus , a future that is free of this virus is the best that one can hope for.

11.BIBLIOGRAPHY

Flask-

https://www.youtube.com/channel/UC4JX40jDee_tINbkjycV4Sg

Matplot library-

https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html

SciPy-

https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.interp1d.html#:~:text=interp1d,_class%20scipy.interpolate&text=Interpolate%20a%20%2DD%20function,the%20value%20of%20new%20points.

Twint-

<https://github.com/twintproject/twint>

Kaggle-

<https://www.kaggle.com/smld80/coronavirus-covid19-tweets>

Time Series Visualizations-

<https://towardsdatascience.com/introduction-to-interactive-time-series-visualizations-with-plotly-in-python-d3219eb7a7af>

Implementation of Twint-

<https://nealcaren.org/lessons/twint/>

Favicon-

<https://fontawesome.com/v4.7.0/icons/>

Google Forms-

<https://www.google.com/forms/about/>

Unsupervised Sentiment Analysis-

<https://towardsdatascience.com/unsupervised-sentiment-analysis-a38bf1906483>

Preparing text for NLP-

<https://medium.com/datadriveninvestor/preparing-text-for-natural-language-processing-f93b11bfb3fe>

Sentiment Analyzer with text classification-

<https://itnext.io/how-to-create-a-sentiment-analyzer-with-text-classification-python-ai-f3a5d10922c5>

Sentiment Score-

<https://www.kdnuggets.com/2018/08/emotion-sentiment-analysis-practitioners-guide-nlp-5.html>

13.APPENDIX:

A.SOURCE CODE:

```
1 #COVID ANALYZER
2
3 #Importing libraries
4
5 import pandas as pd
6 import numpy as np
7 import matplotlib.pyplot as plt
8 import re
9 import nltk
10 from nltk.corpus import stopwords
11 from nltk.stem.porter import PorterStemmer
12 from sklearn.cluster import KMeans
13 from sklearn.feature_extraction.text import CountVectorizer
14 from sklearn.preprocessing import LabelEncoder,
   OneHotEncoder
15
16 #Importing Dataset
17
18 data=pd.read_csv('CORONA.CSV',engine='python')
19 X=data.iloc[:,4].values
20
21 #Filling missing data
22
23 def isNaN(string):
24     return string != string
25
26 for i in range(463418):
27     if(isNaN(X[i])):
28         X[i]="coronavirus"
29 data['text']=X
```

```

30
31 #Cleaning the texts
32
33 corpus = []
34 for i in range(0, 463418):
35     rev=re.sub(r'http\S+', '',data['text'][i] )
36     review = re.sub('[^a-zA-Z0-9]', ' ',rev)
37     review = review.lower()
38     review = review.split()
39     ps = PorterStemmer()
40     sw=stopwords.words('english')
41     sw.remove('not')
42     review = [ps.stem(word) for word in review if not word
43               in set(sw) ]
44     review = ' '.join(review)
45     corpus.append(review)
46
47 #Obtaining sentiment scores
48
49 nltk.download('vader_lexicon')
50 from nltk.sentiment.vader import SentimentIntensityAnalyzer
51 sia=SentimentIntensityAnalyzer()
52 data['neg']=data['text'].apply(lambda
53     x:sia.polarity_scores(x) ['neg'])
54 data['neu']=data['text'].apply(lambda
55     x:sia.polarity_scores(x) ['neu'])
56 data['pos']=data['text'].apply(lambda
57     x:sia.polarity_scores(x) ['pos'])
58 data['compound']=data['text'].apply(lambda
59     x:sia.polarity_scores(x) ['compound'])
60
61 #Percentage values of sentiments
62
63 pos_review=[j for i,j in enumerate(data['text']) if
64     data['compound'][i]>0.2]
65 neu_review=[j for i,j in enumerate(data['text']) if

```



```

0.2>=data['compound'][i]>=-0.2]
61 neg_review=[j for i,j in enumerate(data['text']) if
    data['compound'][i]<=-0.2]
62 print("Percentage of positive
    review:{}%".format(len(pos_review)*100/len(data['text'])))
63 print("Percentage of neutral
    review:{}%".format(len(neu_review)*100/len(data['text'])))
64 print("Percentage of negative
    review:{}%".format(len(neg_review)*100/len(data['text'])))
65
66 #CLUSTERING USING LENGTH
67
68 #Finding length of tweets
69
70 length=[]
71 for i in range(463418):
72     length.append(len(str(X[i])))
73 data['len']=length
74
75 #Elbow method
76
77 '''
78 wcss = []
79 for i in range(1,11):
80     kmeans = KMeans(n_clusters = i, init = 'k-means++',
        random_state = 42)
81     kmeans.fit(Y)
82     wcss.append(kmeans.inertia_)
83 plt.plot(range(1, 11), wcss)
84 plt.title('The Elbow Method')
85 plt.xlabel('Number of clusters')
86 plt.ylabel('WCSS')
87 plt.show()
88 '''
89
90 #K-Means for Clustering using length
91

```

```

92 Y=data.iloc[:, [17,16]].values
93 kmeans = KMeans(n_clusters =2, init = 'k-means++',
    random_state = 0)
94 y_kmeans = kmeans.fit_predict(Y)
95 fig,ax=plt.subplots()
96 ax.scatter(Y[y_kmeans == 1, 0],Y[y_kmeans == 1, 1], s = 10,
    c = 'red', label = 'Cluster 1')
97 ax.scatter(Y[y_kmeans == 0, 0],Y[y_kmeans == 0, 1], s = 10,
    c = 'blue', label = 'Cluster 2')
98 '''
99 plt.scatter(Y[y_kmeans == 2, 0],Y[y_kmeans == 2, 1], s =
    30, c = 'green', label = 'Cluster 3')
100 plt.scatter(Y[y_kmeans == 3, 0],Y[y_kmeans == 3, 1], s =
    100, c = 'cyan', label = 'Cluster 4')
101 plt.scatter(arr[y_kmeans == 4, 0], arr[y_kmeans == 4, 1],
    s = 100, c = 'magenta', label = 'Cluster 5')
102 plt.scatter(arr[y_kmeans == 5, 0], arr[y_kmeans == 5, 1],
    s = 100, c = 'black', label = 'Cluster 6')
103 plt.scatter(arr[y_kmeans == 6, 0], arr[y_kmeans == 6, 1],
    s = 100, c = 'violet', label = 'Cluster 7')
104 plt.scatter(arr[y_kmeans == 7, 0], arr[y_kmeans == 7, 1],
    s = 100, c = 'purple', label = 'Cluster 8')
105 plt.scatter(arr[y_kmeans == 8, 0], arr[y_kmeans == 8, 1],
    s = 100, c = 'grey', label = 'Cluster 9')
106 plt.scatter(arr[y_kmeans == 9, 0], arr[y_kmeans == 9, 1],
    s = 100, c = 'pink', label = 'Cluster 10')
107 '''
108 ax.scatter(kmeans.cluster_centers_[0, 0],
    kmeans.cluster_centers_[0, 1], s = 30, c = 'yellow', label
    = 'Centroids')
109 plt.title('Clusters of people')
110 plt.xlabel('Length of tweet')
111 plt.ylabel('Sentiment score')
112 plt.legend()
113 plt.show()
114 fig.savefig("C:/Users/pjail/OneDrive/Desktop/IBM/images/clus

```

```

    terlength.png")
115
116 #CLUSTERING USING MAX FEATURES
117
118 #Obtaining bag of words model
119
120 cv=CountVectorizer(max_features=1500)
121 X=cv.fit_transform(corpus).toarray()
122
123 #Finding number of max features
124
125 l=[]
126 for i in range(463418):
127     s=0
128     for j in range(1500):
129         s=s+X[i][j]
130     l.append(s)
131 data['len']=l
132
133 #Elbow method
134
135 '''
136 wcss = []
137 for i in range(1,11):
138     kmeans = KMeans(n_clusters = i, init = 'k-means++',
139         random_state = 42)
140     kmeans.fit(Y)
141     wcss.append(kmeans.inertia_)
142 plt.plot(range(1, 11), wcss)
143 plt.title('The Elbow Method')
144 plt.xlabel('Number of clusters')
145 plt.ylabel('WCSS')
146 plt.show()
147
148 #K-Means for Clustering using max features
149

```

```

150 kmeans = KMeans(n_clusters =2, init = 'k-means++',
    random_state = 0)
151 y_kmeans = kmeans.fit_predict(Y)
152 fig,ax=plt.subplots()
153 ax.scatter(Y[y_kmeans == 1, 0],Y[y_kmeans == 1, 1], s = 1,
    c = 'red', label = 'Cluster 1')
154 ax.scatter(Y[y_kmeans == 0, 0],Y[y_kmeans == 0, 1], s = 1,
    c = 'blue', label = 'Cluster 2')
155 '''
156 plt.scatter(Y[y_kmeans == 2, 0],Y[y_kmeans == 2, 1], s =
    1, c = 'green', label = 'Cluster 3')
157 plt.scatter(Y[y_kmeans == 3, 0],Y[y_kmeans == 3, 1], s =
    100, c = 'cyan', label = 'Cluster 4')
158 plt.scatter(arr[y_kmeans == 4, 0], arr[y_kmeans == 4, 1],
    s = 100, c = 'magenta', label = 'Cluster 5')
159 plt.scatter(arr[y_kmeans == 5, 0], arr[y_kmeans == 5, 1],
    s = 100, c = 'black', label = 'Cluster 6')
160 plt.scatter(arr[y_kmeans == 6, 0], arr[y_kmeans == 6, 1],
    s = 100, c = 'violet', label = 'Cluster 7')
161 plt.scatter(arr[y_kmeans == 7, 0], arr[y_kmeans == 7, 1],
    s = 100, c = 'purple', label = 'Cluster 8')
162 plt.scatter(arr[y_kmeans == 8, 0], arr[y_kmeans == 8, 1],
    s = 100, c = 'grey', label = 'Cluster 9')
163 plt.scatter(arr[y_kmeans == 9, 0], arr[y_kmeans == 9, 1],
    s = 100, c = 'pink', label = 'Cluster 10')
164 '''
165 ax.scatter(kmeans.cluster_centers_[0, 0],
    kmeans.cluster_centers_[0, 1], s = 30, c = 'yellow', label
    = 'Centroids')
166 plt.title('Clusters of people')
167 plt.xlabel('Max features')
168 plt.ylabel('Sentiment score')
169 plt.legend()
170 plt.show()
171 fig.savefig("C:/Users/pjail/OneDrive/Desktop/IBM/images/clus
    termax.png")

```

```

172
173 #VERIFIED AND NON-VERIFIED USING SENTIMENT SCORES
174
175 #Obtaining verified and non-verified accounts count
176
177 X=data.iloc[:,11].values
178 vcount=0
179 ncount=0
180 for i in range(463418):
181     if(X[i]==1):
182         vcount+=1
183     else:
184         ncount+=1
185
186 #Count of positive,negative and neutral tweets
187
188 v=[]
189 n=[]
190 for i in range(463418):
191     if(X[i]==1):
192         v.append(data['compound'][i])
193     else:
194         n.append(data['compound'][i])
195 vlist=[0,0,0]
196 nlist=[0,0,0]
197 for i in range(vcount):
198     if(v[i]>0):
199         vlist[0]+=1
200     elif(v[i]==0):
201         vlist[1]+=1
202     else:
203         vlist[2]+=1
204 for i in range(ncount):
205     if(n[i]>0):
206         nlist[0]+=1
207     elif(n[i]==0):

```

```

208         nlist[1]+=1
209     else:
210         nlist[2]+=1
211
212 #Pie Chart for Verified Tweets
213
214 labels = 'Positive','Neutral','Negative'
215 sizes=vlist
216 explode = (0.1, 0, 0)
217 fig1, ax1 = plt.subplots()
218 colors=['#008080','#CACACA','#575757']
219 ax1.pie(sizes, explode=explode,
        labels=labels,colors=colors, autopct='%1.1f%%',shadow=True,
        startangle=90)
220 ax1.axis('equal')
221 plt.show()
222 fig1.savefig('C:/Users/pjail/OneDrive/Desktop/IBM/images/verifiedpie.png',dpi=100, bbox_inches='tight', pad_inches=0.0)
223
224 #Pie Chart for Non-Verified Tweets
225
226 labels = 'Positive','Neutral','Negative'
227 sizes=nlist
228 explode = (0.1, 0, 0)
229 fig2, ax1 = plt.subplots()
230 ax1.pie(sizes, explode=explode,
        labels=labels,colors=colors, autopct='%1.1f%%',shadow=True,
        startangle=90)
231 ax1.axis('equal')
232 plt.show()
233 fig2.savefig('C:/Users/pjail/OneDrive/Desktop/IBM/images/nonverifiedpie.png',dpi=100, bbox_inches='tight',
        pad_inches=0.0)
234
235 #LANGUAGE DOMINANCE
236

```

```

237 #Finding number of tweets in each language
238
239 Y=dataset.iloc[:,12].values
240 X=[]
241 Z=[]
242 langs=['es','zh','en',
        'fr','nl','el','ja','th','hi','ar','pt','tr','tl','fa','et'
        , 'ta','de','it','in','ru','bn','gu','kn','or','te','sl','ne'
        , 'ro','lt','mr','pl','ur','ml','ko','ca','pa','vi','da','n
        o','si','sv','cs','fi','ht','iw','eu','bg','cy','hy','am','s
        r','is','hu','lv','ps','sd','dv','uk','km','lo','ckb','ka']
243 la=[]
244 for i in langs:
245     c=0
246     for j in range(463418):
247         if(Y[j]==i):
248             c=c+1
249     X.append(c)
250
251 #Creating a dictionary
252
253 dic={}
254 lan={}
255 for key in langs:
256     for value in X:
257         lan[key]=value
258         X.remove(value)
259         break
260 Z=[]
261 dic= sorted(lan.items(), key=lambda x: x[1], reverse=True)
262 for i in range(62):
263     Z.append(dic[i][0])
264 for j in range(62):
265     la.append(dic[j][1])
266
267 #Finding top five languages count

```

```

268
269 Z=Z[:5]
270 la=la[:5]
271 fig,ax=plt.subplots()
272 ax.bar(Z[0],la[0],color="#000000")
273 ax.bar(Z[1],la[1],color="#0000CD")
274 ax.bar(Z[2],la[2],color="#00FFFF")
275 ax.bar(Z[3],la[3],color="#ADD8E6")
276 ax.bar(Z[4],la[4],color="#A9A9A9")
277 ax.legend(labels=la)
278 plt.xlabel('Languages')
279 plt.ylabel('Tweets')
280 plt.show()
281 fig.savefig("C:/Users/pjail/OneDrive/Desktop/IBM/images/lang
    uages.png",dpi=100, bbox_inches='tight', pad_inches=0.0)
282
283 #VERIFIED AND NON-VERIFIED
284
285 posvalue=[0,0]
286 negvalue=[0,0]
287 neuvalue=[0,0]
288 for i in range(463418):
289     if(data['compound'][i]>0):
290         if(data['verified'][i]==1):
291             posvalue[0]+=1
292         else:
293             posvalue[1]+=1
294     elif(data['compound'][i]<0):
295         if(data['verified'][i]==1):
296             negvalue[0]+=1
297         else:
298             negvalue[1]+=1
299     else:
300         if(data['verified'][i]==1):
301             neuvalue[0]+=1
302         else:

```



```

303         neuvalue[1]+=1
304
305 #Donut for Positive Sentiments
306
307 labels =['Verified','Non Verified']
308 sizes=posvalue
309 colors = ['#1515B4','#3BB9FF']
310 my_circle=plt.Circle( (0,0), 0.5, color='white')
311 patches, texts=plt.pie(sizes,colors=colors)
312 plt.rcParams['text.color'] = 'black'
313 plt.legend(patches, labels, loc="upper right")
314 plt.title('Positive')
315 p=plt.gcf()
316 plt.pie(sizes,colors=colors, wedgeprops = { 'linewidth':7,
        'edgecolor' : 'white' },autopct='%1.1f%%',pctdistance=0.7)
317 p.gca().add_artist(my_circle)
318 p.savefig("C:/Users/pjail/OneDrive/Desktop/IBM/images/posve
        rnon.png")
319 plt.show()
320
321 #Donut for Negative Sentiments
322
323 labels =['Verified','Non Verified']
324 sizes=negvalue
325 colors = ['#6F4E37','#C85A17']
326 my_circle=plt.Circle( (0,0), 0.5, color='white')
327 patches, texts=plt.pie(sizes,colors=colors)
328 plt.rcParams['text.color'] = 'black'
329 plt.legend(patches, labels, loc="upper right")
330 plt.title('Negative')
331 p=plt.gcf()
332 plt.pie(sizes,colors=colors, wedgeprops = { 'linewidth':7,
        'edgecolor' : 'white' },autopct='%1.1f%%',pctdistance=0.7)
333 p.gca().add_artist(my_circle)
334 p.savefig("C:/Users/pjail/OneDrive/Desktop/IBM/images/negve
        rnon.png")

```

```

335 plt.show()
336
337 #Donut for Neutral Sentiments
338
339 labels = ['Verified', 'Non Verified']
340 sizes=neuvalue
341 colors = ['#EC83D9', '#ED1FC4']
342 my_circle=plt.Circle( (0,0), 0.5, color='white')
343 patches, texts=plt.pie(sizes,colors=colors)
344 plt.rcParams['text.color'] = 'black'
345 plt.legend(patches, labels, loc="upper right")
346 plt.title('Neutral')
347 p=plt.gcf()
348 plt.pie(sizes,colors=colors, wedgeprops = { 'linewidth':7,
    'edgecolor' : 'white' }, autopct='%1.1f%%',pctdistance=0.7)
349 p.gca().add_artist(my_circle)
350 p.savefig("C:/Users/pjail/OneDrive/Desktop/IBM/images/neuve
    rnon.png")
351 plt.show()
352
353 #TOTAL SENTIMENTS
354
355 value=[0,0,0]
356 for i in range(num):
357     if(data['compound'][i]>0):
358         value[0]+=1
359     elif(data['compound'][i]==0):
360         value[1]+=1
361     else:
362         value[2]+=1
363
364 #Plotting overall sentiments
365
366 labels = 'Positive','Neutral','Negative'
367 sizes=value
368 fig1, ax1 = plt.subplots()

```

```

369 my_circle=plt.Circle( (0,0), 0.5, color='white')
370 plt.pie(sizes,labels=labels,labeldistance=1.2,autopct='%1.
    1f%%',pctdistance=0.7,wedgeprops = { 'linewidth' : 7,
    'edgecolor' : 'white' })
371 plt.rcParams['text.color'] = 'black'
372 p=plt.gcf()
373 p.gca().add_artist(my_circle)
374 ax.legend(labels=['Positive','Neutral','Negative'])
375 p.savefig("C:/Users/pjail/OneDrive/Desktop/IBM/images/posne
    g.png")
376 plt.show()
377
378 #IMPACT CLUSTERING
379
380 #Elbow Method
381 '''
382 wcss = []
383 for i in range(1, 11):
384     kmeans = KMeans(n_clusters = i, init = 'k-means++',
        random_state = 42)
385     kmeans.fit(Y)
386     wcss.append(kmeans.inertia_)
387 plt.plot(range(1, 11), wcss)
388 plt.title('The Elbow Method')
389 plt.xlabel('Number of clusters')
390 plt.ylabel('WCSS')
391 plt.show()
392 '''
393 #K Means Clustering for followers vs favourite count
394
395 Y=dataset.iloc[:,[8,6]].values
396 kmeans = KMeans(n_clusters =2, init = 'k-means++',
    random_state = 0)
397 y_kmeans = kmeans.fit_predict(Y)
398 fig,ax=plt.subplots()
399 ax.scatter(Y[y_kmeans == 0, 0],Y[y_kmeans == 0, 1], s =

```

```

30, c = 'red', label = 'Cluster 1')
400 ax.scatter(Y[y_kmeans == 1, 0],Y[y_kmeans == 1, 1], s =
30, c = 'blue', label = 'Cluster 2')
401 '''
402 plt.scatter(Y[y_kmeans == 2, 0],Y[y_kmeans == 2, 1], s =
100, c = 'green', label = 'Cluster 3')
403 plt.scatter(Y[y_kmeans == 3, 0],Y[y_kmeans == 3, 1], s =
100, c = 'cyan', label = 'Cluster 4')
404 plt.scatter(arr[y_kmeans == 4, 0], arr[y_kmeans == 4, 1],
s = 100, c = 'magenta', label = 'Cluster 5')
405 plt.scatter(arr[y_kmeans == 5, 0], arr[y_kmeans == 5, 1],
s = 100, c = 'black', label = 'Cluster 6')
406 plt.scatter(arr[y_kmeans == 6, 0], arr[y_kmeans == 6, 1],
s = 100, c = 'violet', label = 'Cluster 7')
407 plt.scatter(arr[y_kmeans == 7, 0], arr[y_kmeans == 7, 1],
s = 100, c = 'purple', label = 'Cluster 8')
408 plt.scatter(arr[y_kmeans == 8, 0], arr[y_kmeans == 8, 1],
s = 100, c = 'grey', label = 'Cluster 9')
409 plt.scatter(arr[y_kmeans == 9, 0], arr[y_kmeans == 9, 1],
s = 100, c = 'pink', label = 'Cluster 10')
410 '''
411 ax.scatter(kmeans.cluster_centers_[0, 0],
kmeans.cluster_centers_[0, 1], s = 100, c = 'yellow', label
= 'Centroids')
412 plt.title('Clusters of people')
413 plt.ylabel('No. of Followers')
414 plt.xlabel('Favourites count')
415 plt.legend()
416 plt.show()
417 fig.savefig("C:/Users/pjail/OneDrive/Desktop/IBM/images/impa
ctcluster.png",dpi=100, bbox_inches='tight',
pad_inches=0.0)
418
419 corpus = []
420 for i in range(0,num):
421     rev=re.sub(r'http\S+', '',data['text'][i] )
422     review = re.sub('[^a-zA-Z0-9]', ' ',rev)

```

```

423     review = review.lower()
424     review = review.split()
425     ps = PorterStemmer()
426     sw=stopwords.words('english')
427     sw.remove('not')
428     review = [ps.stem(word) for word in review if not word
in set(sw) ]
429     review = ' '.join(review)
430     corpus.append(review)
431
432 #CLASSIFICATION
433
434 i=100000
435 cv = CountVectorizer(max_features = 1500)
436 X = cv.fit_transform(corpus).toarray()
437 X_train=[]
438 X_test=[]
439 y=[]
440 while i<num:
441     y=data.iloc[:i,18].values
442
443     data=pd.read_csv('CORONA.CSV',nrows=i+10000,engine="python"
)
444     X_train=X[:i]
445     X_test=X[i:i+10000]
446     classifier = BernoulliNB()
447     classifier.fit(X_train, y)
448     y_pred = classifier.predict(X_test)
449     y=y.tolist()
450     if (num-i)>=10000):
451         for k in range(10000):
452             y.append(y_pred[k])
453     else:
454         for k in range(num-i):
455             y.append(y_pred[k])
456     data['labels']=pd.Series(y)

```

```

456     i+=10000
457
458 data['labels']=pd.Series(y)
459
460
461 #LABELS VS VERIFIED AND NON-VERIFIED
462
463 Y=data.iloc[:,18].values
464 X=data.iloc[:,11].values
465 labels=['sad','happy','angry','informative']
466 v=[0,0,0,0]
467 n=[0,0,0,0]
468 for i in range(num):
469     if(X[i]==1):
470         if(Y[i]=='sad'):
471             v[0]=v[0]+1
472         elif(Y[i]=='happy'):
473             v[1]=v[1]+1
474         elif(Y[i]=='angry'):
475             v[2]=v[2]+1
476         else:
477             v[3]=v[3]+1
478     else:
479         if(Y[i]=='sad'):
480             n[0]=n[0]+1
481         elif(Y[i]=='happy'):
482             n[1]=n[1]+1
483         elif(Y[i]=='angry'):
484             n[2]=n[2]+1
485         else:
486             n[3]=n[3]+1
487
488 #Labels for verified accounts
489
490 fig,ax=plt.subplots()
491 ax.bar(labels[0],v[0],color="#000000")

```

```

492 ax.bar(labels[1],v[1],color="#0000CD")
493 ax.bar(labels[2],v[2],color="#00FFFF")
494 ax.bar(labels[3],v[3],color="#ADD8E6")
495 ax.legend(labels=['sad', 'happy', 'angry', 'informative'])
496 plt.xlabel('Emotions')
497 plt.ylabel('No. of accounts')
498 plt.title('Emotions of Verified Users')
499 plt.show()
500 fig.savefig("../flask/images/labelsvsverified.png",dpi=100,
    bbox_inches='tight', pad_inches=0.0)
501
502 #Labels for non-verified accounts
503
504 fig,ax=plt.subplots()
505 ax.bar(labels[0],n[0],color="#000000")
506 ax.bar(labels[1],n[1],color="#0000CD")
507 ax.bar(labels[2],n[2],color="#00FFFF")
508 ax.bar(labels[3],n[3],color="#ADD8E6")
509 ax.legend(labels=['sad', 'happy', 'angry', 'informative'])
510 plt.xlabel('Emotions')
511 plt.ylabel('No. of accounts')
512 plt.title('Emotions of Non-Verified Users')
513 plt.show()
514 fig.savefig("../flask/images/labelsvsnv.png",dpi=100,
    bbox_inches='tight', pad_inches=0.0)
515
516
517 #LABELS VS SENTIMENTS
518
519 W=data.iloc[:,18].values
520
521 #Encoding data
522
523 labelencoder_X = LabelEncoder()
524 labelencoder_X.fit_transform(W)
525 data['new']=W

```

```

526 Z=data.iloc[:,[19,16]].values
527
528 #Elbow method
529 '''
530 wcss = []
531 for i in range(1, 11):
532     kmeans = KMeans(n_clusters = i, init = 'k-means++',
533         random_state = 42)
534     kmeans.fit(Z)
535     wcss.append(kmeans.inertia_)
536 plt.plot(range(1, 11), wcss)
537 plt.title('The Elbow Method')
538 plt.xlabel('Number of clusters')
539 plt.ylabel('WCSS')
540 plt.show()
541
542 # K Means for sentiment score vs labels
543 kmeans = KMeans(n_clusters = 4, init = 'k-means++',
544     random_state = 42)
545 y_kmeans = kmeans.fit_predict(Z)
546 plt.scatter(Z[y_kmeans == 0, 0], Z[y_kmeans == 0,
547     1],marker='*',s = 100, c = 'red', label = 'Angry')
548 plt.scatter(Z[y_kmeans == 1, 0], Z[y_kmeans == 1,
549     1],marker='P', s = 70, c = 'blue', label = 'Sad')
550 plt.scatter(Z[y_kmeans == 2, 0], Z[y_kmeans == 2,
551     1],marker='8', s = 30, c = 'green', label = 'Informative')
552 plt.scatter(Z[y_kmeans == 3, 0], Z[y_kmeans == 3,
553     1],marker='2', s = 500, c = 'black',label = 'Happy' )
554 #plt.scatter(kmeans.cluster_centers_[0, 0],
555     kmeans.cluster_centers_[0, 1], s = 30, c = 'yellow', label
556     = 'Centroids')
557 plt.title('Sentiments VS Emotions')
558 plt.xlabel('Emotion')
559 plt.ylabel('Sentiment score')
560 ax = plt.subplot(111)

```



```

554 box = ax.get_position()
555 ax.set_position([box.x0, box.y0, box.width*0.65,
    box.height])
556 legend_x = 1.5
557 legend_y = 0.5
558 plt.legend(loc="center right", bbox_to_anchor=(legend_x,
    legend_y))
559 plt.savefig("../flask/images/sentivsemo.png", dpi=100,
    bbox_inches='tight', pad_inches=0.0)
560
561 #ANALYSIS OF LABELS
562
563 #Label-SAD
564
565 Y=dataset.iloc[:,18].values
566 X=dataset.iloc[:,11].values
567 labels=['sad', 'happy', 'angry', 'informative']
568 Z=[]
569 for i in labels:
570     c=0
571     for j in range(num):
572         if(Y[j]==i):
573             c=c+1
574     Z.append(c)
575 svno=[0,0]
576 for i in range(num):
577     if(Y[i]=='sad'):
578         if(X[i]==1):
579             svno[0]=svno[0]+1
580         else:
581             svno[1]=svno[1]+1
582
583 #Pie Chart for Sad
584
585 labels = 'Verified', 'Non-Verified'
586 sizes=svno

```

```

587 explode = (0.1, 0)
588 colors=['#A74AC7', '#461B7E']
589 fig1, ax1 = plt.subplots()
590 ax1.pie(sizes, explode=explode,
        labels=labels, colors=colors, autopct='%1.1f%%', shadow=True,
        startangle=180)
591 ax1.axis('equal')
592 plt.title('Sad comments')
593 fig1.savefig("../flask/images/Sad.png")
594
595 #Label-HAPPY
596
597 hvno=[0,0]
598 for i in range(num):
599     if(Y[i]=='happy'):
600         if(X[i]==1):
601             hvno[0]=hvno[0]+1
602         else:
603             hvno[1]=hvno[1]+1
604
605 #Pie Chart for Happy
606
607 labels = 'Verified', 'Non-Verified'
608 sizes=hvno
609 explode = (0.1, 0)
610 colors=['#FFDB58', '#8B4513']
611 fig1, ax1 = plt.subplots()
612 ax1.pie(sizes, explode=explode, colors=colors,
        labels=labels, autopct='%1.1f%%', shadow=True,
        startangle=180)
613 ax1.axis('equal')
614 plt.title('Happy comments')
615 plt.show()
616 fig1.savefig("../flask/images/Happy.png")
617
618 #Label-ANGRY

```

```

619
620 avno=[0,0]
621 for i in range(num):
622     if(Y[i]=='angry'):
623         if(X[i]==1):
624             avno[0]=avno[0]+1
625         else:
626             avno[1]=avno[1]+1
627
628 #Pie Chart for Angry
629
630 labels = 'Verified','Non-Verified'
631 sizes=avno
632 colors=['#F75D59','#CC0000']
633 explode = (0.1, 0)
634
635 fig1, ax1 = plt.subplots()
636 ax1.pie(sizes, explode=explode,
        labels=labels,colors=colors, autopct='%1.1f%%',shadow=True,
        startangle=180)
637 ax1.axis('equal')
638 plt.title('Angry comments')
639 plt.show()
640 fig1.savefig("../flask/images/Angry.png")
641
642 #Label-INFORMATIVE
643
644 ivno=[0,0]
645 for i in range(num):
646     if(Y[i]=='informative'):
647         if(X[i]==1):
648             ivno[0]=avno[0]+1
649         else:
650             ivno[1]=avno[1]+1
651
652 #Pie Chart for Informative

```

```

653
654 labels = 'Verified','Non-Verified'
655 sizes=ivno
656 colors=['#FFB6C1','#FF1493']
657 explode = (0.1, 0)
658 fig1, ax1 = plt.subplots()
659 ax1.pie(sizes, explode=explode, colors=colors,
        labels=labels, autopct='%1.1f%%', shadow=True,
        startangle=180)
660 ax1.axis('equal')
661 plt.title('Informative comments')
662 plt.show()
663 fig1.savefig("../flask/images/Informative.png")
664
665 #OVERALL CHART FOR LABELS
666
667 #Pie Chart for Labels
668
669 labels = 'Angry','Happy','Sad','Informative'
670 sizes=Z
671 explode = (0, 0.1,0,0)
672 fig1, ax1 = plt.subplots()
673 ax1.pie(sizes, explode=explode, labels=labels,
        autopct='%1.1f%%', shadow=True, startangle=180)
674 ax1.axis('equal')
675 plt.title('Tweet emotions')
676 plt.show()
677 fig1.savefig("../flask/images/Labels.png")
678
679 #Donut Chart
680
681 labels =['Angry','Happy','Sad','Informative']
682 sizes=Z
683 colors = ['#EC83D9','#A74AC7','#461B7E','#ED1FC4']
684 my_circle=plt.Circle( (0,0), 0.5, color='white')
685 patches, texts=plt.pie(sizes,colors=colors)

```

```

686 plt.rcParams['text.color'] = 'black'
687 plt.legend(patches, labels, loc=1)
688 plt.title('Tweet Emotions')
689 p=plt.gcf()
690 plt.pie(sizes,colors=colors, wedgeprops = { 'linewidth':2,
        'edgecolor' : 'white' },autopct='%1.1f%%',pctdistance=0.7)
691 p.gca().add_artist(my_circle)
692 p.savefig("../flask/images/Labels1.png")
693 plt.show()
694
695
696 #Flask code
697 from flask import Flask ,render_template
698 from flask_fontawesome import FontAwesome
699 import stats
700 import titletophome
701 import tweetstop
702 import CovidAnalyzer
703 import sentitop
704 app = Flask(__name__)
705 fa = FontAwesome(app)
706
707
708 @app.route('/home')
709 def home():
710
711     return
    render_template("index.html",tottweet=titletophome.tottweet
    ,engtweet=titletophome.engtweet,pos=sentitop.pnn[0],info=ti
    tletophome.info,ver=titletophome.ver,date=titletophome.date
    )
712 @app.route('/positive')
713 def positive():
714
715     return
    render_template("index2.html",var1=stats.favb,var2=stats.fa

```

```

        vbfols, var3=stats.folbfavs)
716
717
718 @app.route('/tweets')
719 def tweets():
720     return
        render_template("index3.html", info0=tweetstop.info[0], info1
        =tweetstop.info[1], info2=tweetstop.info[2], sad0=tweetstop.s
        ad[0], sad1=tweetstop.sad[1], sad2=tweetstop.sad[2], angry0=tw
        eetstop.angry[0], angry1=tweetstop.angry[1], angry2=tweetstop
        .angry[2], happy0=tweetstop.happy[0], happy1=tweetstop.happy[
        1], happy2=tweetstop.happy[2])
721
722 @app.route('/sentiment')
723 def sentiment():
724     return
        render_template("index4.html", post=sentitop.pnn[0], negt=sen
        titop.pnn[1], neut=sentitop.pnn[2], posp=CovidAnalyzer.posp, n
        eup=CovidAnalyzer.neup, negp=CovidAnalyzer.negp)
725
726 @app.route('/data_visualization')
727 def data_visualisation():
728     return render_template("ChartJs.html")
729 if __name__ == '__main__':
730     app.run()
731
732

```