Group no. 4

3121 : Disha Athani
3122 : Divya Gavane
3154 : Samruddhi Nabriya
3245 : Samruddhi Raut

Covid19 Data Analysis and Prediction

April 27, 2021

Overview:

Coronavirus disease (COVID-19) is an inflammation disease from a new virus. The disease causes respiratory ailment (like influenza) with manifestations, for example, cold, cough and fever, and in progressively serious cases, the problem in breathing. COVID-2019 has been perceived as a worldwide pandemic and a few examinations are being led utilizing different numerical models to anticipate the likely advancement of this pestilence. These numerical models dependent on different factors and investigations are dependent upon potential inclination. Here, we presented a model that could be useful to predict the spread of COVID-2019. We have performed Support Vector Regression method on the COVID-19 data, collected from "**covid19india.org**" to anticipate the epidemiological example of the ailment and pace of COVID-2019 cases in India.

Introduction :

As of date confirmed COVID-19 cases across the globe are 1,498,833 and mortality approximately 5.8%. Gradually the mortality rate is increasing and it's an alarming factor for the whole world. Transmission is categorized into 4 stages based on the mode of spread and time. Every nation imposed different methodologies starting from staying in-home, using masks, travel restrictions, avoiding social gatherings, frequently washing

hands and sanitizing the places often in the case of a common effort to combat the outbreak of this disease. Many countries imposed a lockdown state that prevents the movement of the citizens unnecessarily. Due to this social distancing factor and movement restrictions, the wellbeing and economy of the various nations are being under jeopardy. GDP of the entire world dropped drastically. When the person is found infected, he is isolated and treatment is given for recovery. But based on the severity it will cause death and also people left with a higher level of depression.

Objectives and Scope :

- The main objective of this paper is to predict and forecast COVID-19 cases, deaths, and recoveries through predictive modelling, and to decipher patterns on public sentiment related to health information dissemination. At the same time, assess the political and economic impact of the virus spread.
- This project proposes a machine learning model that can predict the number of cases well in advance very effectively and also suggest some key inputs.
- An analysis on COVID-19 datasets to understand which age group is mostly affected due to COVID-19. A prediction model is built using machine learning algorithms and their performances are computed and evaluated.

Literature Survey :

ARTICLES AND RESEARCH PAPERS	AUTHOR	ALGORITHMS USED
1.https://towardsdatascience.com/covid-19-outbreak-prediction-using-machine-learning-algorithm-ce5641bd55bf	Wie Kiang H	Support vector regression and polynomial regression
2.https://towardsdatascience.com/tracking-corona-covid-19-spread-in-india-using-python-40ef8ffa7e31	Pratik Nabriya	Python Based Analysis

3.https://www.researchgate.net/publication/341778862_Analysis_Prediction_and_Evaluation_of_COVID-19_Datasets_using_Machine_Learning_Algorithms	Kolla Bhanu Prakash, S. Sagar Imambi, Mohammed Ismail, T Pavan Kumar, YVR Naga Pawan	Random forest regressor, random forest classifier and SVM
4.https://onlinelibrary.wiley.com/doi/10.1002/pa.2537	Ritanjali Majhi, Rahul Thangeda, Renu Prasad Sugasi, Niraj Kumar	Random forest algorithm, Decision tree and non-linear regression

Dataset and Specifications :

Dataset selection:

The dataset for our project was taken from :- https://www.covid19india.org/. This site provides the current status and numbers for the covid 19 pandemic situation in India and also separate files of data for various months in json format. The datasets are also available in csv format.This is based data available at particular time.

Code for merging of all dataset:

- Import all the required libraries like numpy, panda (and json if dataset is in json format).
- Get data from API and store it in a variable to create a dataframe.
- Similarly read all the datasets ranging from Jan to November from the site mentioned above onto the jupyter notebook
- Create different dataframes for all of them so that it becomes easier to merge later into a single dataframe

- Next step is to retain the columns wherein the unnecessary columns can be removed (such as columns with NaN)
- Features selected for analysis of covid19 in India are number of cases, date, age bracket, gender, detected city and state and current status
- Final step is to merge all the dataframes into one by appending them into a single dataframe and storing into a new variable.
- Save all the data in csv format.

```
In [3]: df1=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data1.csv')
        df2=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data2.csv')
        df3=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data3.csv')
        df4=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data4.csv')
        df5=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data5.csv')
        df6=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data6.csv')
        df7=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data7.csv')
        df8=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data8.csv')
        df9=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data9.csv')
        df10=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data10.csv')
        df11=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data11.csv')
        df12=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data12.csv')
        df13=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data13.csv')
        df14=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data14.csv')
        df15=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data15.csv')
        df16=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data16.csv')
        df17=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data17.csv')
        df18=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data18.csv')
        df19=pd.read_csv('https://api.covid19india.org/csv/latest/raw_data19.csv')

In [6]: df12.columns

Out[6]: Index(['Entry_ID', 'State Patient Number', 'Date Announced', 'Age Bracket',
               'Gender', 'Detected City', 'Detected District', 'Detected State',
               'State code', 'Num Cases', 'Current Status',
               'Contracted from which Patient (Suspected)', 'Notes', 'Source_1',
               'Source_2', 'Source_3', 'Nationality', 'Type of transmission',
               'Status Change Date', 'Patient Number'],
              dtype='object')
```

```
In [7]: df1=df1.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Curre
        df2=df2.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Curre
        df3=df3.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Curre
        df4=df4.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Curre
        df5=df5.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Curre
        df6=df6.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Curre
        df7=df7.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Curre
        df8=df8.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Curre
        df9=df9.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Curre
        df10=df10.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Cur
        df11=df11.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Cur
        df12=df12.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Cur
        df13=df13.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Cur
        df14=df14.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Cur
        df15=df15.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Cur
        df16=df16.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Cur
        df17=df17.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Cur
        df18=df18.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Cur
        df19=df19.loc[:,['Num Cases','Date Announced','Age Bracket','Gender', 'Detected City', 'Detected District', 'Detected State','Cur
```

Merge all datasets

In [8]:
```
df=df1.append([df2,df3,df4,df5,df6,df7,df8,df9,df10,df11,df12,df13,df14,df15,df16,df17,df18,df19])
df
```

Out[8]:

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered
...	...	...	...	...	...	...	...	...
25379	1.0	21/11/2020	NaN	NaN	NaN	Thiruvananthapuram	Kerala	Migrated_Other
25380	1.0	24/11/2020	NaN	NaN	NaN	Kannur	Kerala	Migrated_Other
25381	1.0	25/11/2020	NaN	NaN	NaN	Thiruvananthapuram	Kerala	Migrated_Other
25382	1.0	28/11/2020	NaN	NaN	NaN	Thrissur	Kerala	Migrated_Other
25383	1.0	30/11/2020	NaN	NaN	NaN	Thiruvananthapuram	Kerala	Migrated_Other

430339 rows × 8 columns

In [9]:
```
Date=df['Date Announced'].str.split('/',expand=True)
Date.columns=['Day','Month','Year']
df=pd.concat([df,Date],axis=1)
df
```

Out[9]:

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	01	2020
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	02	02	2020
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	03	02	2020
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	02	03	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	02	03	2020
...	...	...	...	...	...	...	...	...	...	...	...
25379	1.0	21/11/2020	NaN	NaN	NaN	Thiruvananthapuram	Kerala	Migrated_Other	21	11	2020
25380	1.0	24/11/2020	NaN	NaN	NaN	Kannur	Kerala	Migrated_Other	24	11	2020
25381	1.0	25/11/2020	NaN	NaN	NaN	Thiruvananthapuram	Kerala	Migrated_Other	25	11	2020
25382	1.0	28/11/2020	NaN	NaN	NaN	Thrissur	Kerala	Migrated_Other	28	11	2020
25383	1.0	30/11/2020	NaN	NaN	NaN	Thiruvananthapuram	Kerala	Migrated_Other	30	11	2020

430339 rows × 11 columns

In [10]:
```
df.to_csv('Covid19India.csv')
```

Algorithm brief :

- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This

best decision boundary is called a hyperplane in N-Dimensional space [N=no. of features or independent variables] .

- There are 2 marginal planes parallel to the hyperplane such that the marginal distance is max. Support vectors are those points that cross through the marginal plane.
- Usually there are 2 types of separation i.e. linearly separable and non-linear separable.
- SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine.
- **SVR :**

  SVM kernels:- The main aim is to convert low dimension(2-D) to HD(3-D) so we are easily able to classify particular points by a hyperplane.

Code :

- The coding part is divided into two phases namely
  - ➢ Covid 19 Data Analysis and Visualisation
  - ➢ Covid 19 Data Prediction using Support Vector Machines (SVM) algorithm

1. Covid 19 Data Analysis and Visualisation code :
2. Covid 19 Data Prediction using Support Vector Machines (SVM) algorithm :

Covid 19 Data Analysis and Visualisation

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
df=pd.read_csv('Covid19Indiafinal.csv')
df
```

```
D:\Anaconda\lib\site-packages\IPython\core\interactiveshell.py:3071: DtypeWarning: Columns (3) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Out[3]:

	Unnamed: 0	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	1	2020
1	1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	2	2	2020
2	2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	3	2	2020
3	3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	2	3	2020
4	4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	2	3	2020
...	...	...	...	...	...	...	...	...	...	...	...	...
145844	22790	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized	7	7	2020
145845	22791	-9.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	7	7	2020
145846	22792	-6.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	7	7	2020
145847	22793	-1.0	07/07/2020	NaN	NaN	NaN	Kozhikode	Kerala	Recovered	7	7	2020
145848	22794	1.0	07/07/2020	NaN	NaN	NaN	Wayanad	Kerala	Recovered	7	7	2020

145848	22794	1.0	07/07/2020	NaN	NaN	NaN	Wayanad	Kerala	Recovered	7	7	2020

145849 rows × 12 columns

In [4]:
```
data=df.iloc[:,1:]
data
```

Out[4]:

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	1	2020
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	2	2	2020
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	3	2	2020
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	2	3	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	2	3	2020
...	...	...	...	...	...	...	...	...	...	...	...
145844	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized	7	7	2020
145845	-9.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	7	7	2020
145846	-6.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	7	7	2020
145847	-1.0	07/07/2020	NaN	NaN	NaN	Kozhikode	Kerala	Recovered	7	7	2020
145848	1.0	07/07/2020	NaN	NaN	NaN	Wayanad	Kerala	Recovered	7	7	2020

145849 rows × 11 columns

In [6]:
```
##Inspect data
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145849 entries, 0 to 145848
Data columns (total 11 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Num Cases          145846 non-null  float64
 1   Date Announced     145849 non-null  object 
 2   Age Bracket        60013 non-null   object 
 3   Gender             62808 non-null   object 
 4   Detected City      10949 non-null   object 
 5   Detected District  137451 non-null  object 
 6   Detected State     145840 non-null  object 
 7   Current Status     145847 non-null  object 
 8   Day                145849 non-null  int64  
 9   Month              145849 non-null  int64  
 10  Year               145849 non-null  int64  
dtypes: float64(1), int64(3), object(7)
memory usage: 12.2+ MB
```

In [8]:
```
#Inspect Null Values in each Column
```

In [9]:
```
data.isnull().sum(axis=0).sort_values()
```

Out[9]:
```
Date Announced            0
Day                       0
Month                     0
Year                      0
Current Status            2
Num Cases                 3
Detected State            9
Detected District      8398
Gender                83041
Age Bracket           85836
Detected City        134900
dtype: int64
```

In [10]:
```
data.isnull().sum(axis=0).sort_values(ascending=False)/len(data)*100 #gives the percentage of values that are  missing
```

Out[10]:
```
Detected City        92.492921
Age Bracket          58.852649
Gender               56.936283
Detected District     5.758010
Detected State        0.006171
Num Cases             0.002057
Current Status        0.001371
Year                  0.000000
Month                 0.000000
Day                   0.000000
Date Announced        0.000000
dtype: float64
```

```
In [12]: data.isnull().sum(axis=1).sort_values(ascending=False)
```

```
Out[12]: 28451     6
         69009     6
         69008     6
         4375      5
         28398     5
                  ..
         108695    0
         108694    0
         108693    0
         108692    0
         0         0
         Length: 145849, dtype: int64
```

```
In [13]: ##Total Covid-19 cases month wise
```

```
In [15]: data.groupby('Month')['Num Cases'].sum()
```

```
Out[15]: Month
         1         1.0
         2         2.0
         3      1635.0
         4     36078.0
         5    242853.0
         6    663178.0
         7    270185.0
         Name: Num Cases, dtype: float64
```

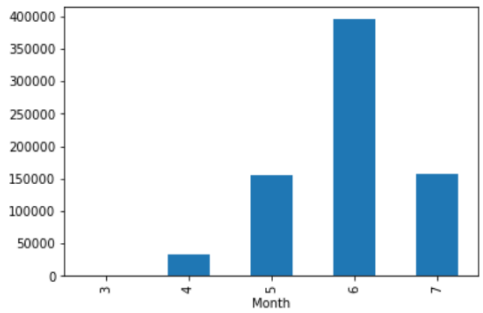
```
In [16]: data['Current Status']=='Hospitalized'
```

```
Out[16]: 0         False
         1         False
         2         False
         3         False
         4         False
                   ...
         145844     True
         145845    False
         145846    False
         145847    False
         145848    False
         Name: Current Status, Length: 145849, dtype: bool
```

```
In [17]: data[data['Current Status']=='Hospitalized'].groupby('Month')['Num Cases'].sum()
```

```
Out[17]: Month
         3      1431.0
         4     33209.0
         5    155781.0
         6    395144.0
         7    157701.0
         Name: Num Cases, dtype: float64
```

```
In [19]: M=data[data['Current Status']=='Hospitalized'].groupby('Month')['Num Cases'].sum()
         M.plot.bar()
         plt.show()
```




```
In [20]: ##Total Male/Female infected with coronavirus
```

```
In [21]: data.groupby('Gender')['Num Cases'].sum()
```

```
Out[21]: Gender
         F              21294.0
         M              42795.0
         M                  1.0
         Non-Binary        12.0
         Name: Num Cases, dtype: float64
```

In [25]:
```
data.groupby('Age Bracket')['Num Cases'].sum()
```

Out[25]:
```
Age Bracket
0.1       3.0
0.6       1.0
0.9       3.0
1.0      20.0
1.5       1.0
         ...
97        1.0
97.0      1.0
98.0      2.0
99        2.0
99.0      1.0
Name: Num Cases, Length: 325, dtype: float64
```

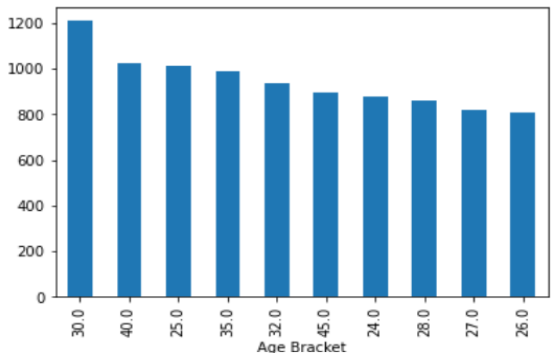
In [26]:
```
data.groupby('Age Bracket')['Num Cases'].sum().sort_values(ascending=False)
```

Out[26]:
```
Age Bracket
30.0        1209.0
40.0        1027.0
25.0        1015.0
35.0         992.0
32.0         936.0
             ...
1.6            1.0
29.6           1.0
5 Months       1.0
54.9           1.0
99.0           1.0
Name: Num Cases, Length: 325, dtype: float64
```

In [27]:
```
#if age is between 30 and 40 max chances of corona
```

In [30]:
```
M=data.groupby('Age Bracket')['Num Cases'].sum().sort_values(ascending=False).head(10)
M
```

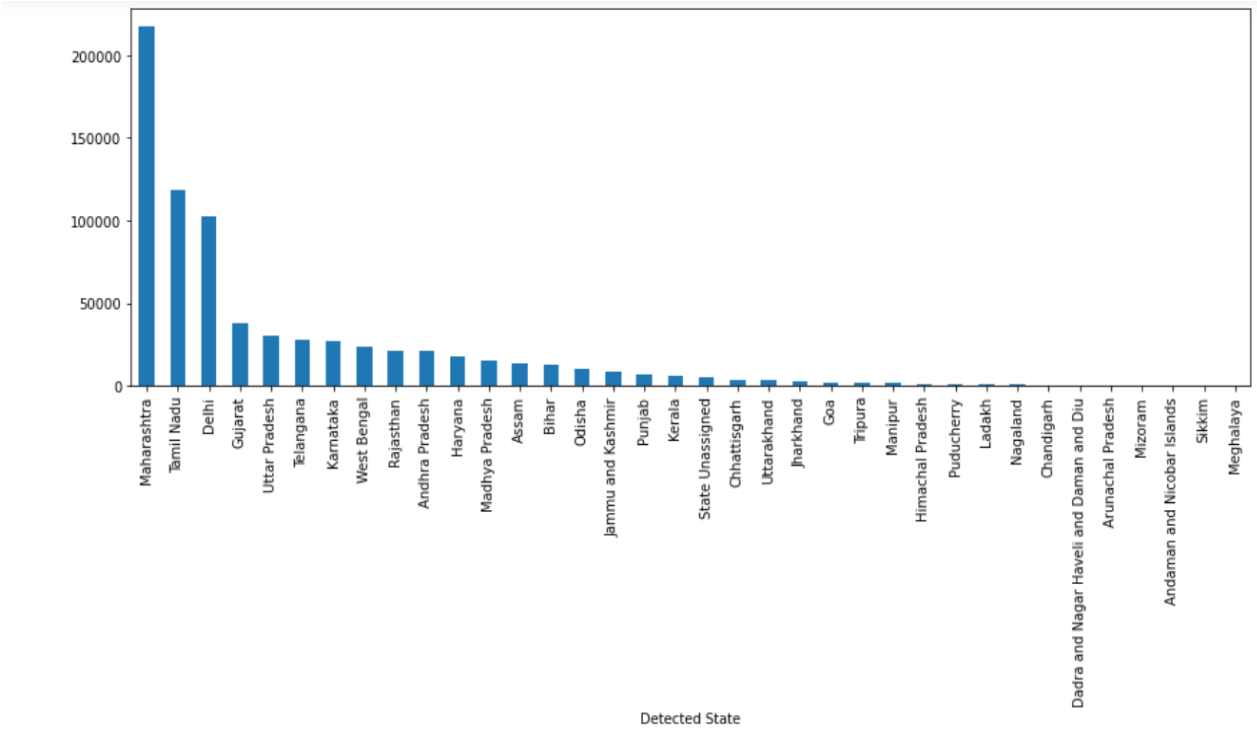
Out[30]:
```
Age Bracket
30.0    1209.0
40.0    1027.0
25.0    1015.0
35.0     992.0
32.0     936.0
45.0     893.0
24.0     880.0
28.0     858.0
27.0     818.0
26.0     810.0
Name: Num Cases, dtype: float64
```



In [39]:
```
M=data[data['Current Status']=='Hospitalized'].groupby('Detected State')['Num Cases'].sum().sort_values(ascending=False)
M
```

Out[39]:

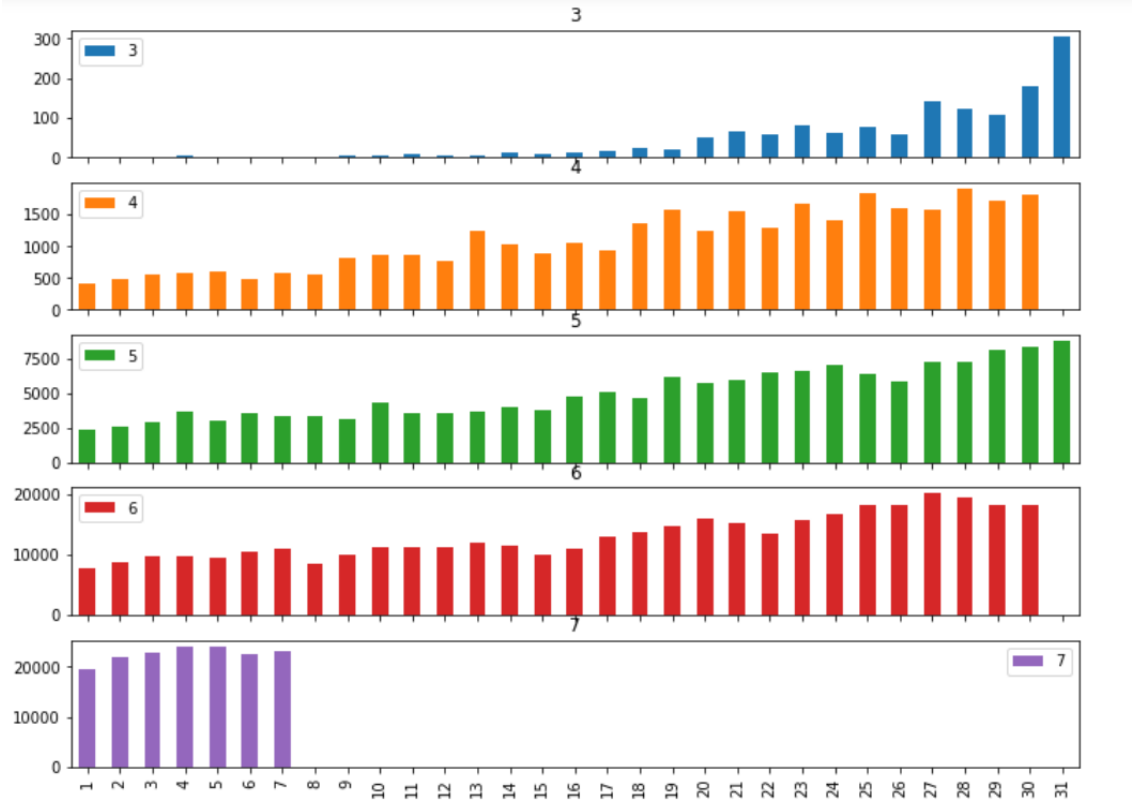
Detected State	
Maharashtra	217107.0
Tamil Nadu	118587.0
Delhi	102827.0
Gujarat	37631.0
Uttar Pradesh	29959.0
Telangana	27610.0
Karnataka	26743.0
West Bengal	23831.0
Rajasthan	21400.0
Andhra Pradesh	21195.0
Haryana	17987.0
Madhya Pradesh	15625.0
Assam	13337.0
Bihar	12524.0
Odisha	10096.0
Jammu and Kashmir	8930.0
Punjab	6747.0
Kerala	5834.0
State Unassigned	5034.0
Chhattisgarh	3407.0
Uttarakhand	3229.0
Jharkhand	3018.0
Goa	1903.0
Tripura	1716.0
Manipur	1429.0
Himachal Pradesh	1081.0
Puducherry	1043.0
Ladakh	1041.0



In [44]:
```
##Cases each day
```

In [46]:
```
Day=data[data['Current Status']=='Hospitalized'].groupby(['Month','Day'])['Num Cases'].sum()
Day
```

Out[46]:
```
Month  Day
3      4          5.0
       5          1.0
       7          2.0
       9          4.0
       10         4.0
                 ...
7      3      22718.0
       4      24018.0
       5      23942.0
       6      22500.0
       7      23147.0
Name: Num Cases, Length: 124, dtype: float64
```

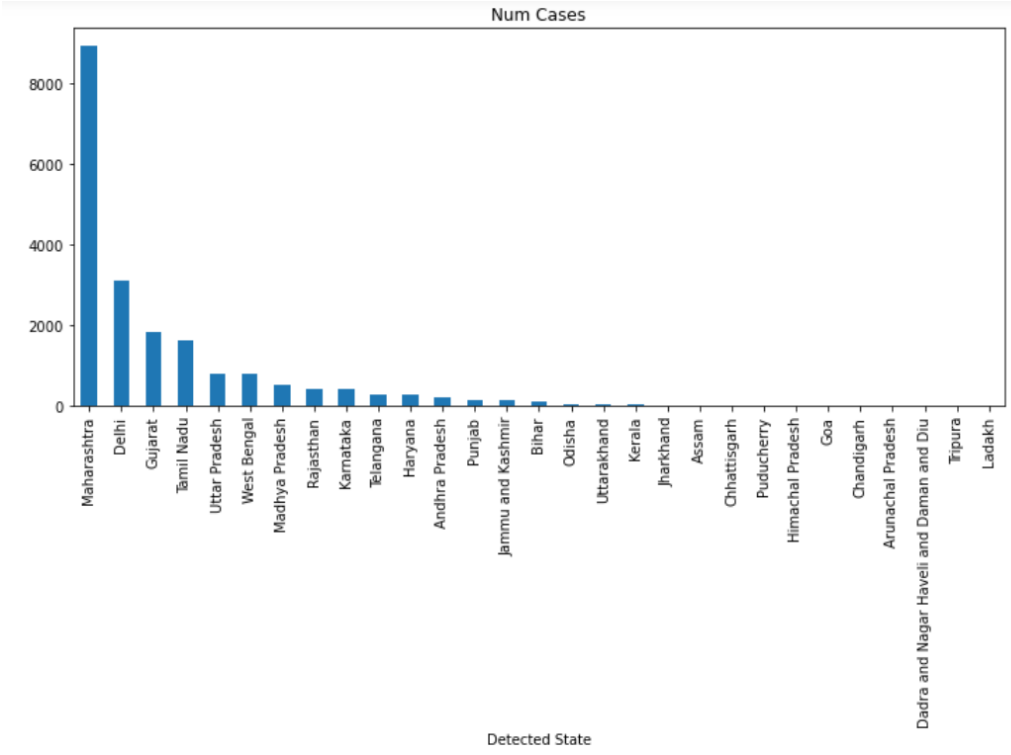


```
In [51]: data[data['Current Status']=='Deceased']['Num Cases'].sum()
```

```
Out[51]: 19817.0
```

```
In [44]: data[data['Current Status']=='Deceased'].groupby('Detected State')['Num Cases'].sum().sort_values(ascending=False)
```

```
Out[44]: Detected State
         Maharashtra                 8919.0
         Delhi                       3112.0
         Gujarat                     1831.0
         Tamil Nadu                  1613.0
         Uttar Pradesh                798.0
         West Bengal                  787.0
         Madhya Pradesh               521.0
         Rajasthan                    432.0
         Karnataka                    409.0
         Telangana                    289.0
         Haryana                      276.0
         Andhra Pradesh               221.0
         Punjab                       159.0
         Jammu and Kashmir            138.0
         Bihar                         97.0
         Odisha                        54.0
         Uttarakhand                   43.0
         Kerala                        27.0
         Jharkhand                     19.0
         Assam                         15.0
         Chhattisgarh                  14.0
         Puducherry                    14.0
         Himachal Pradesh               9.0
         Goa                            8.0
```



Covid 19 Data Prediction using SVM :

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
df1 = pd.read_csv('https://api.covid19india.org/csv/latest/raw_data1.csv')
df1
```

Out[2]:

State Patient Number	Date Announced	Estimated Onset Date	Age Bracket	Gender	Detected City	Detected District	Detected State	State code	...	Notes	Contracted from which Patient (Suspected)	Nationality	Type of transmission	Status Change Date
KL-TS-P1	30/01/2020	NaN	20	F	Thrissur	Thrissur	Kerala	KL	...	Travelled from Wuhan	NaN	India	Imported	14/02/2020
KL-AL-P1	02/02/2020	NaN	NaN	NaN	Alappuzha	Alappuzha	Kerala	KL	...	Travelled from Wuhan	NaN	India	Imported	14/02/2020
KL-KS-P1	03/02/2020	NaN	NaN	NaN	Kasaragod	Kasaragod	Kerala	KL	...	Travelled from Wuhan	NaN	India	Imported	14/02/2020
DL-P1	02/03/2020	NaN	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	DL	...	Travelled from Austria, Italy	NaN	India	Imported	15/03/2020
										Travelled				

DL-P1	02/03/2020	NaN	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	DL	...	Travelled from Austria, Italy	NaN	India	Imported	15/03/2020
TS-P1	02/03/2020	NaN	24	M	Hyderabad	Hyderabad	Telangana	TG	...	Travelled from Dubai to Bangalore on 20th Feb,...	NaN	India	Imported	02/03/2020
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
NaN	16/04/2020	NaN	NaN	NaN	NaN	NaN	Maharashtra	MH	...	Correction for district count	NaN	NaN	NaN	NaN
NaN	18/04/2020	NaN	NaN	NaN	NaN	Nagpur	Maharashtra	MH	...	Correction for district count	NaN	NaN	NaN	NaN
NaN	18/04/2020	NaN	NaN	NaN	NaN	NaN	Maharashtra	MH	...	Correction for district count	NaN	NaN	NaN	NaN
NaN	19/04/2020	NaN	NaN	NaN	NaN	Nagpur	Maharashtra	MH	...	Correction for district count	NaN	NaN	NaN	NaN
NaN	19/04/2020	NaN	NaN	NaN	NaN	NaN	Maharashtra	MH	...	Correction for district count	NaN	NaN	NaN	NaN

In [15]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 456648 entries, 0 to 456647
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Num Cases         456629 non-null  float64
 1   Date Announced    456641 non-null  object
 2   Age Bracket       117132 non-null  object
 3   Gender            119407 non-null  object
 4   Detected City     14422 non-null   object
 5   Detected District 444493 non-null  object
 6   Detected State    456626 non-null  object
 7   Current Status    456635 non-null  object
 8   Day               456641 non-null  float64
 9   Month             456641 non-null  float64
 10  Year              456641 non-null  float64
dtypes: float64(4), object(7)
memory usage: 38.3+ MB
```

Match all the Columns

In [5]:
```
df1 = df1.rename(columns={"Num cases":"Num Cases"})
df2 = df2.rename(columns={"Num cases":"Num Cases"})
```

In [6]:
```
df1.columns
```

Out[6]:
```
Index(['Patient Number', 'State Patient Number', 'Date Announced',
       'Estimated Onset Date', 'Age Bracket', 'Gender', 'Detected City',
       'Detected District', 'Detected State', 'State code', 'Current Status',
       'Notes', 'Contracted from which Patient (Suspected)', 'Nationality',
       'Type of transmission', 'Status Change Date', 'Source_1', 'Source_2',
       'Source_3', 'Backup Notes', 'Num Cases'],
      dtype='object')
```

Retain Necessary Columns

In [7]:
```
df1 = df1.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df2 = df2.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df3 = df3.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df4 = df4.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df5 = df5.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df6 = df6.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df7 = df7.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
```

Retain Necessary Columns

In [7]:
```
df1 = df1.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df2 = df2.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df3 = df3.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df4 = df4.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df5 = df5.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df6 = df6.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df7 = df7.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df8 = df8.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df9 = df9.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State', 'Cu
df10 = df10.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State',
df11 = df11.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State',
df12 = df12.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State',
df13 = df13.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State',
df14 = df14.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State',
df15 = df15.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State',
df16 = df16.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State',
df17 = df17.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State',
df18 = df18.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State',
df19 = df19.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State',
df20 = df20.loc[:,['Num Cases','Date Announced','Age Bracket', 'Gender', 'Detected City','Detected District', 'Detected State',
```

Merge all DataFrames

In [8]:
```
df = df1.append([df2,df3,df4,df5,df6,df7,df8,df9,df10,df11,df12,df13,df14,df15,df16,df17,df18,df19,df20])
df
```

Out[8]:

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered
...	...	...	...	...	...	...	...	...
26304	1.0	19/12/2020	NaN	NaN	NaN	Chitrakoot	Uttar Pradesh	Recovered
26305	30.0	19/12/2020	NaN	NaN	NaN	Jhansi	Uttar Pradesh	Hospitalized
26306	19.0	19/12/2020	NaN	NaN	NaN	Jhansi	Uttar Pradesh	Recovered
26307	4.0	19/12/2020	NaN	NaN	NaN	Ambedkar Nagar	Uttar Pradesh	Hospitalized
26308	3.0	19/12/2020	NaN	NaN	NaN	Ambedkar Nagar	Uttar Pradesh	Recovered

456648 rows × 8 columns

Making Seperate Columns for Day,Month,Year

In [9]:
```
DATE = df['Date Announced'].str.split('/',expand=True)
DATE.columns = ['Day','Month','Year']
DATE
```

Out[9]:

	Day	Month	Year
0	30	01	2020
1	02	02	2020
2	03	02	2020
3	02	03	2020
4	02	03	2020
...	...	...	...
26304	19	12	2020
26305	19	12	2020
26306	19	12	2020
26307	19	12	2020
26308	19	12	2020

456648 rows × 3 columns

Concatinate both the DataFrames along Axis=1

In [10]:
```
df = pd.concat([df,DATE],axis=1)
df
```

Out[10]:

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	01	2020
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	02	02	2020
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	03	02	2020
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	02	03	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	02	03	2020
...	...	...	...	...	...	...	...	...	...	...	...
26304	1.0	19/12/2020	NaN	NaN	NaN	Chitrakoot	Uttar Pradesh	Recovered	19	12	2020
26305	30.0	19/12/2020	NaN	NaN	NaN	Jhansi	Uttar Pradesh	Hospitalized	19	12	2020
26306	19.0	19/12/2020	NaN	NaN	NaN	Jhansi	Uttar Pradesh	Recovered	19	12	2020
26307	4.0	19/12/2020	NaN	NaN	NaN	Ambedkar Nagar	Uttar Pradesh	Hospitalized	19	12	2020

26307	4.0	19/12/2020	NaN	NaN	NaN	Ambedkar Nagar	Uttar Pradesh	Hospitalized	19	12	2020
26308	3.0	19/12/2020	NaN	NaN	NaN	Ambedkar Nagar	Uttar Pradesh	Recovered	19	12	2020

456648 rows × 11 columns

Save in CSV format

```
In [11]: df.to_csv('Covid19India.csv')
```

Final DataSet

```
In [12]: df
```

Out[12]:

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	01	2020
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	02	02	2020
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	03	02	2020
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	02	03	2020

```
In [12]: df
```

Out[12]:

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	01	2020
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	02	02	2020
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	03	02	2020
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	02	03	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	02	03	2020
...	...	...	...	...	...	...	...	...	...	...	...
26304	1.0	19/12/2020	NaN	NaN	NaN	Chitrakoot	Uttar Pradesh	Recovered	19	12	2020
26305	30.0	19/12/2020	NaN	NaN	NaN	Jhansi	Uttar Pradesh	Hospitalized	19	12	2020
26306	19.0	19/12/2020	NaN	NaN	NaN	Jhansi	Uttar Pradesh	Recovered	19	12	2020
26307	4.0	19/12/2020	NaN	NaN	NaN	Ambedkar Nagar	Uttar Pradesh	Hospitalized	19	12	2020
26308	3.0	19/12/2020	NaN	NaN	NaN	Ambedkar Nagar	Uttar Pradesh	Recovered	19	12	2020

456648 rows × 11 columns

Prediction and Analysis using Support Vector Regression

In [13]:
```
data = pd.read_csv('Covid19India.csv')
data
```

```
C:\Users\divya\Anaconda\lib\site-packages\IPython\core\interactiveshell.py:3071: DtypeWarning: Columns (3) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Out[13]:

	Unnamed: 0	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30.0	1.0	2020.0
1	1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	2.0	2.0	2020.0
2	2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	3.0	2.0	2020.0
3	3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	2.0	3.0	2020.0
4	4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	2.0	3.0	2020.0
...	...	...	...	...	...	...	...	...	...	...	...	...
456643	26304	1.0	19/12/2020	NaN	NaN	NaN	Chitrakoot	Uttar Pradesh	Recovered	19.0	12.0	2020.0
456644	26305	30.0	19/12/2020	NaN	NaN	NaN	Jhansi	Uttar Pradesh	Hospitalized	19.0	12.0	2020.0
456645	26306	19.0	19/12/2020	NaN	NaN	NaN	Jhansi	Uttar Pradesh	Recovered	19.0	12.0	2020.0
456646	26307	4.0	19/12/2020	NaN	NaN	NaN	Ambedkar Nagar	Uttar Pradesh	Hospitalized	19.0	12.0	2020.0
456647	26308	3.0	19/12/2020	NaN	NaN	NaN	Ambedkar Nagar	Uttar Pradesh	Recovered	19.0	12.0	2020.0

456648 rows × 12 columns

In [14]:
```
data = data.iloc[:,1:]
data
```

Out[14]:

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30.0	1.0	2020.0
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	2.0	2.0	2020.0
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	3.0	2.0	2020.0
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	2.0	3.0	2020.0
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	2.0	3.0	2020.0
...	...	...	...	...	...	...	...	...	...	...	...
456643	1.0	19/12/2020	NaN	NaN	NaN	Chitrakoot	Uttar Pradesh	Recovered	19.0	12.0	2020.0
456644	30.0	19/12/2020	NaN	NaN	NaN	Jhansi	Uttar Pradesh	Hospitalized	19.0	12.0	2020.0
456645	19.0	19/12/2020	NaN	NaN	NaN	Jhansi	Uttar Pradesh	Recovered	19.0	12.0	2020.0
456646	4.0	19/12/2020	NaN	NaN	NaN	Ambedkar Nagar	Uttar Pradesh	Hospitalized	19.0	12.0	2020.0
456647	3.0	19/12/2020	NaN	NaN	NaN	Ambedkar Nagar	Uttar Pradesh	Recovered	19.0	12.0	2020.0

456648 rows × 11 columns

In [15]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 456648 entries, 0 to 456647
Data columns (total 11 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Num Cases          456629 non-null  float64
 1   Date Announced     456641 non-null  object 
 2   Age Bracket        117132 non-null  object 
 3   Gender             119407 non-null  object 
 4   Detected City      14422 non-null   object 
 5   Detected District  444493 non-null  object 
 6   Detected State     456626 non-null  object 
 7   Current Status     456635 non-null  object 
 8   Day                456641 non-null  float64
 9   Month              456641 non-null  float64
 10  Year               456641 non-null  float64
dtypes: float64(4), object(7)
memory usage: 38.3+ MB
```

In [16]:
```
Day = data[data['Current Status']=='Hospitalized'].groupby(['Month','Day'])['Num Cases'].sum()
Day
```

Out[16]:
```
Month  Day
3.0    4.0         5.0
       5.0         1.0
       7.0         2.0
       9.0         4.0
       10.0        4.0
                  ...
12.0   15.0    26251.0
       16.0    18172.0
       17.0    26754.0
       18.0    26991.0
       19.0    26834.0
Name: Num Cases, Length: 289, dtype: float64
```

In [17]:
```
x = np.arange(len(Day))
x
```

Out[17]:
```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
```

In [349]:
```
y = Day.values
y
```

Out[349]:
```
array([5.0000e+00, 1.0000e+00, 2.0000e+00, 4.0000e+00, 4.0000e+00,
       8.0000e+00, 4.0000e+00, 6.0000e+00, 1.1000e+01, 8.0000e+00,
       1.2000e+01, 1.4000e+01, 2.2000e+01, 2.1000e+01, 5.2000e+01,
       6.7000e+01, 5.9000e+01, 8.2000e+01, 6.3000e+01, 7.5000e+01,
       5.8000e+01, 1.4000e+02, 1.2300e+02, 1.0600e+02, 1.7800e+02,
       3.0600e+02, 4.2300e+02, 4.8500e+02, 5.5600e+02, 5.7600e+02,
       6.0600e+02, 4.8500e+02, 5.7000e+02, 5.6300e+02, 8.1200e+02,
       8.7000e+02, 8.5300e+02, 7.5800e+02, 1.2430e+03, 1.0310e+03,
       8.8400e+02, 1.0610e+03, 9.2200e+02, 1.3700e+03, 1.5790e+03,
       1.2390e+03, 1.5370e+03, 1.2920e+03, 1.6670e+03, 1.4080e+03,
       1.8350e+03, 1.6070e+03, 1.5680e+03, 1.9020e+03, 1.7050e+03,
       1.8020e+03, 2.3960e+03, 2.5640e+03, 2.9520e+03, 3.6560e+03,
       2.9710e+03, 3.6020e+03, 3.3440e+03, 3.3390e+03, 3.1750e+03,
       4.3110e+03, 3.5920e+03, 3.5620e+03, 3.7260e+03, 3.9910e+03,
       3.8080e+03, 4.7940e+03, 5.0490e+03, 4.6280e+03, 6.1540e+03,
       5.7200e+03, 6.0230e+03, 6.5360e+03, 6.6650e+03, 7.1110e+03,
       6.4140e+03, 5.9070e+03, 7.2460e+03, 7.2540e+03, 8.1380e+03,
       8.3640e+03, 8.7890e+03, 7.7240e+03, 8.8120e+03, 9.6880e+03,
       9.8470e+03, 9.4720e+03, 1.0408e+04, 1.0882e+04, 8.5360e+03,
       9.9810e+03, 1.1156e+04, 1.1135e+04, 1.1306e+04, 1.2039e+04,
       1.1404e+04, 1.0032e+04, 1.1085e+04, 1.3108e+04, 1.3829e+04,
       1.4740e+04, 1.5918e+04, 1.5151e+04, 1.3560e+04, 1.5656e+04,
       1.6868e+04, 1.8205e+04, 1.8255e+04, 2.0142e+04, 1.9610e+04,
       1.8339e+04, 1.8255e+04, 1.9430e+04, 2.1947e+04, 2.2718e+04,
       2.4018e+04, 2.3942e+04, 2.2500e+04, 2.3148e+04, 2.5561e+04,
```

In [350]:
```
x = x.reshape(-1,1)
y = y.reshape(-1,1)
```

In [351]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.30,shuffle = True)
print(X_train.shape,X_test.shape,y_train.shape,y_test.shape)
```

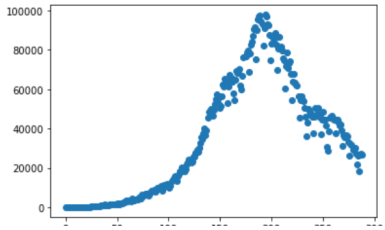
```
(202, 1) (87, 1) (202, 1) (87, 1)
```

In [352]:
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
Sx = sc_X.fit_transform(X_train)
Sy = sc_y.fit_transform(y_train)
```

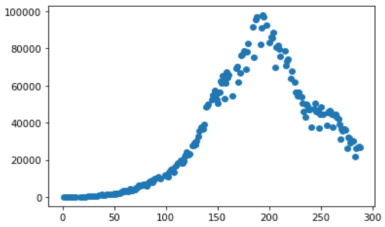
In [353]:
```
print(Sx)
print("*****************************************************************************************************")
print(Sy)
```

```
[[-1.20566204]
 [ 0.12258582]
 [ 0.6918349 ]
 [-0.50596004]
 [-0.20947615]
 [ 0.84600652]
 [-1.01591235]
```

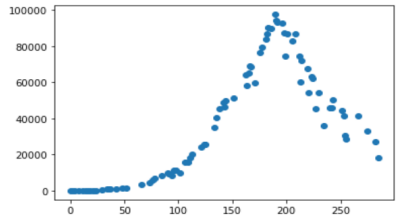
```
In [354]: from sklearn.svm import SVR
          reg = SVR(kernel='rbf') #RBF => Radial Biased Function
          print(reg.fit(Sx,Sy.ravel())) #ravel() converts 2d to 1d

          SVR()

In [355]: reg.score(Sx,Sy)*100

Out[355]: 97.0293414681747

In [356]: plt.scatter(x,y)
          plt.show()
```




```
In [358]: plt.scatter(X_train,y_train)
          plt.show()
```




```
In [361]: plt.scatter(X_test,y_test)
          plt.show()
```



Outcomes of analysis :

From the analysis of available data, following are the outcomes:

1. Highest number of cases recorded in the month of June 2020.
2. Age group most affected by Covid 19 : 30-40 years.
3. Maharashtra recorded maximum number of cases.
4. Highest number of cases recorded every day were maximum in June 2020.

5. Highest number of deceased were recorded in Maharashtra.

Outcomes of prediction :

The score of prediction comes out to be around 97.029%. The above curve is obtained when the train test size ratio is taken as 70-30.

This project helped us to analyse and predict the spread of Covid 19 disease using machine learning algorithm.Such predictions aid authorities, heath workers and task forces for making informed decisions and take measures to control the spread.

References :

1. https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/#:~:text=A%20support%20vector%20machine%20(SVM,able%20to%20categorize%20new%20text.
2. https://en.wikipedia.org/wiki/Support-vector_machine
3. Dataset-https://www.covid19india.org/