Data Science Project

COVID-19 ANALYSIS

Project By:

PRASFUR TIWARI
DIVYANSH TIWARI
SONAM KUMAR

INDEX

1. Introduction

Background

Currently, there are so many dashboards and statistics around the Corona virus spread available all over the internet. With so much of information and expert opinions, to see different nations adopting different strategies, from complete lockdown to social distancing to herd immunity. This project is an attempt of data modeling and analyzing Corona virus (COVID-19) spread in India with the help of data science and data analytics in python code.

Problem

Government of India is also taking all necessary steps to ensure that we are prepared well to face the challenge and threat posed by the growing pandemic of COVID-19 the Corona Virus. On **24 March 2020**, the Government of India under Prime Minister Narendra Modi ordered a **nationwide lockdown** for 21 days, limiting movement of the entire 1.3 billion population of India as a preventive measure against the COVID-19 pandemic in India which was appreciated as well as criticized. This project aims to predict and compare the growth of cases with or without lockdown.

2. Data Analysis

Data sources

The Datasets containing state-wise information of Confirmed Cases, Deaths and Cured Individuals was obtained from kaggle.com. Other datasets containing population data of states as well as Info Age Group Affected were also obtained from the same repository.

Data Preprocessing, Wrangling and Filtering:

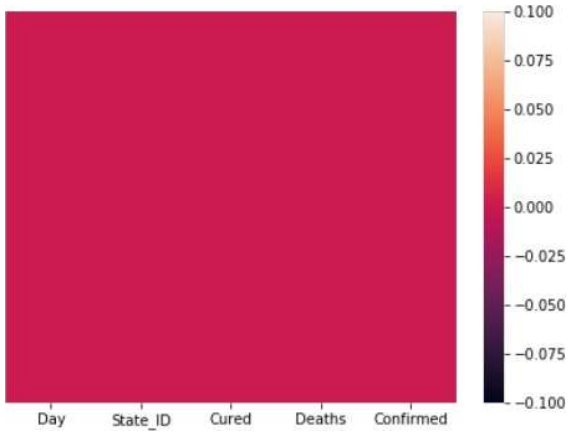
Using python library 'Pandas', the csv files were converted to data frame for further processing. Since we only needed "Cured", "Deaths" and "Confirmed" cases all the unnecessary columns were removed. The Dates were replaced with Day (30/01/20 as 1st Day) and State name by State-ID (Alphabetically).

	Day	State_ID	Cured	Deaths	Confirmed
0	1	1	0	0	0
1	1	2	0	0	0
2	1	3	0	0	0
3	1	4	0	0	0
4	1	5	0	0	0

Data Cleaning

The data frame needs to get rid of all the null values to fit the model. Hence, we need to identify the columns with Null/NaN values and print A count of the null values from each column. Our data frame has no Null values and hence we need not handle them. Also, a Heatmap was generated to check the uncertainties in the dataset.

```
Day           0
State_ID      0
Cured         0
Deaths        0
Confirmed     0
dtype: int64
```



3. Exploratory data analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

Descriptive Statistics

It is a good practice to know the columns and their corresponding data types, along with finding whether they contain null values or not.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3888 entries, 0 to 3887
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Day        3888 non-null   int64
 1   State_ID   3888 non-null   int64
 2   Cured      3888 non-null   int64
 3   Deaths     3888 non-null   int64
 4   Confirmed  3888 non-null   int64
dtypes: int64(5)
memory usage: 152.0 KB
```

The '**describe()**' function in pandas is very handy in getting various summary statistics. This function returns the count, mean, standard deviation, minimum and maximum values and the quantities of the data.

	Day	State_ID	Cured	Deaths	Confirmed
count	3888.000000	3888.000000	3888.000000	3888.000000	3888.000000
mean	54.500000	18.500000	94.222737	11.428498	353.127058
std	31.179588	10.389631	400.506480	62.268418	1592.878800
min	1.000000	1.000000	0.000000	0.000000	0.000000
25%	27.750000	9.750000	0.000000	0.000000	0.000000
50%	54.500000	18.500000	0.000000	0.000000	1.000000
75%	81.250000	27.250000	10.000000	1.000000	41.000000
max	108.000000	36.000000	6564.000000	1068.000000	29100.000000

Correlation

Pandas' **'dataframe.corr()'** is used to find the pair-wise correlation of all columns in a data frame. We also use Pearson method to find correlation of Day with Cured, Death and Confirmed to find if the relation is statistically significant **(p-value)** and how strong is the relationship.

	Day	State_ID	Cured	Deaths	Confirmed
Day	1.000000	0.000000	0.332959	0.244722	0.293391
State_ID	0.000000	1.000000	0.044153	0.024726	0.039588
Cured	0.332959	0.044153	1.000000	0.836089	0.926386
Deaths	0.244722	0.024726	0.836089	1.000000	0.931426
Confirmed	0.293391	0.039588	0.926386	0.931426	1.000000

Day Vs Confirmed Cases

```
The Pearson Correlation Coefficient is 0.29339149322005065  with a P-value of P =
4.835636472628199e-78
```

Observation:

Since the p-value is < 0.001, the correlation between Day and Confirmed Cases is statistically significant, and the **linear relationship isn't quite strong** (~0.293).

Day Vs Deaths

```
The Pearson Correlation Coefficient is 0.24472239126055217  with a P-value of P =
4.015735020158 98e-54
```

Observation

Since the p-value is < 0.001, the correlation between Day and Deaths is statistically significant, and the **linear relationship isn't quite strong** (~0.244).

Day Vs Cured

```
The Pearson Correlation Coefficient is 0.24472239126055217  with a P-value of P =
4.015735020158 98e-54
```

Observation

Since the p-value is < 0.001, the correlation between Day and Deaths is statistically significant, and the **linear relationship isn't quite strong** (~0.244).

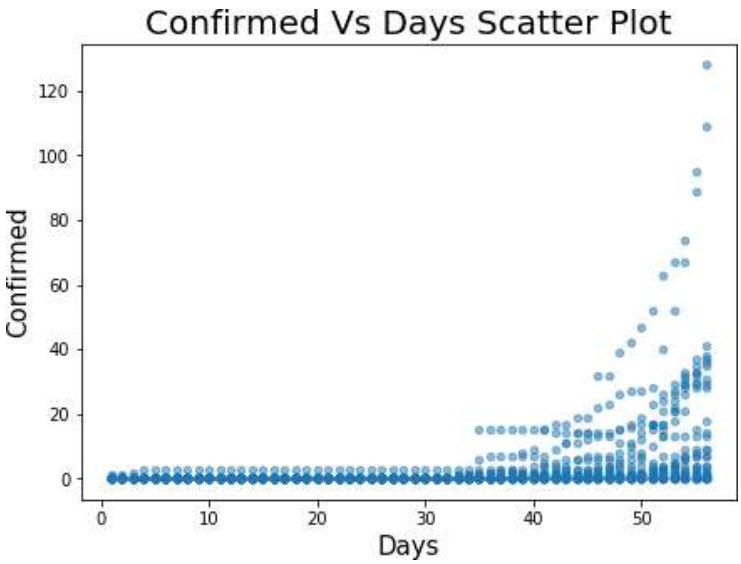
4. Data Visualization

Basic plotting

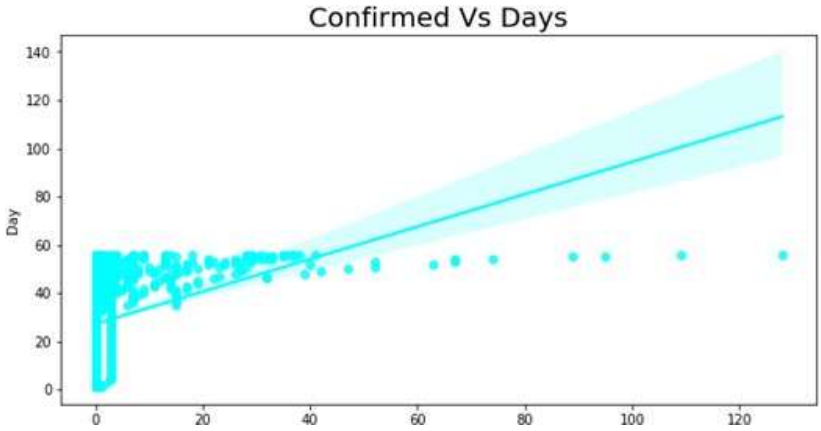
We take the data of only 56 days from the beginning, i.e. data till first lockdown, and keep the rest of the data for prediction.

Heat Map: Heat maps display numeric tabular data where the cells are colored depending upon the contained value

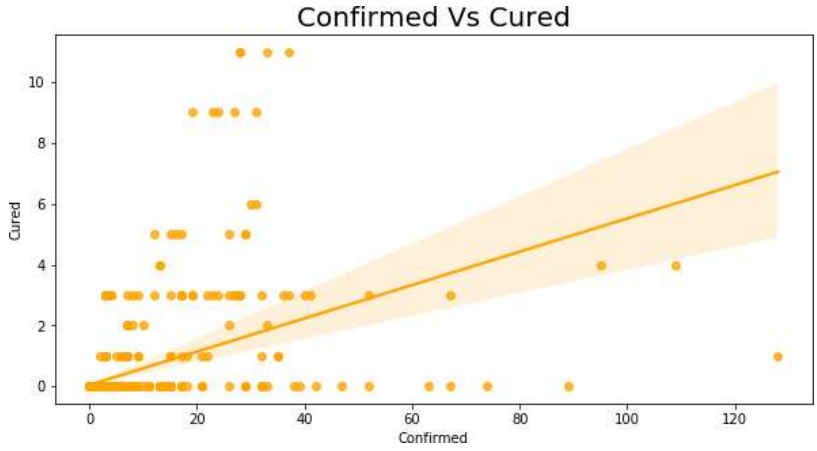
Confirmed Cases and Days Scatter Plot



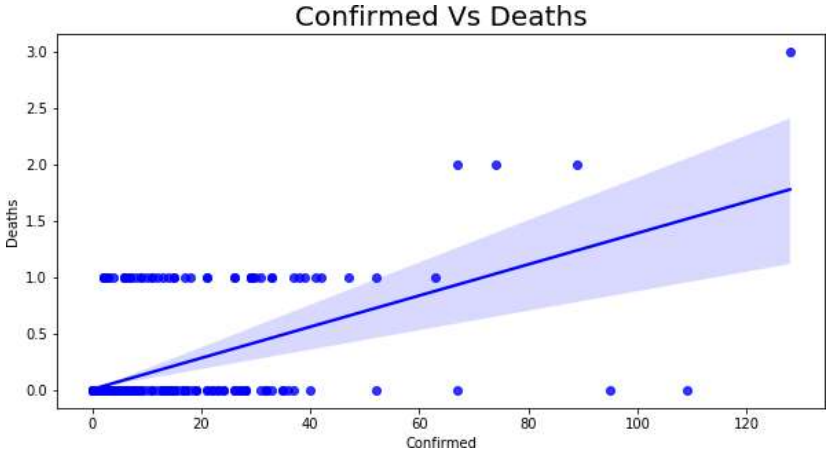
Confirmed Cases and Days Regression Plot



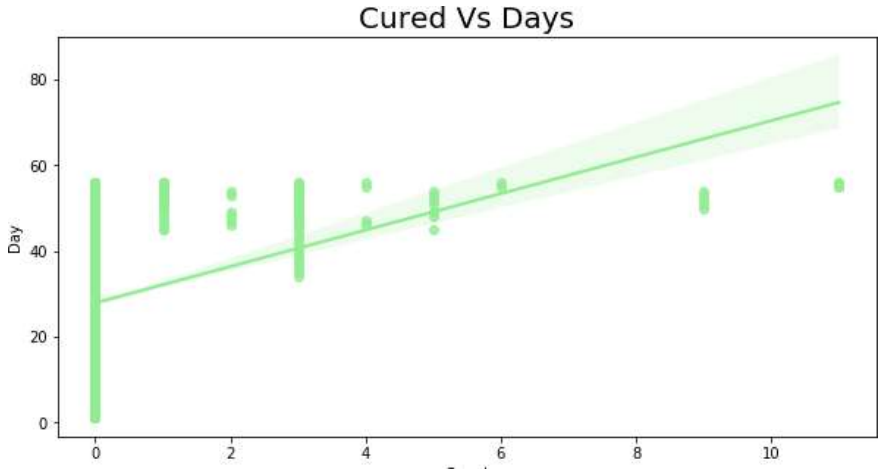
Confirmed Cases and Cured Cases Regression Plot



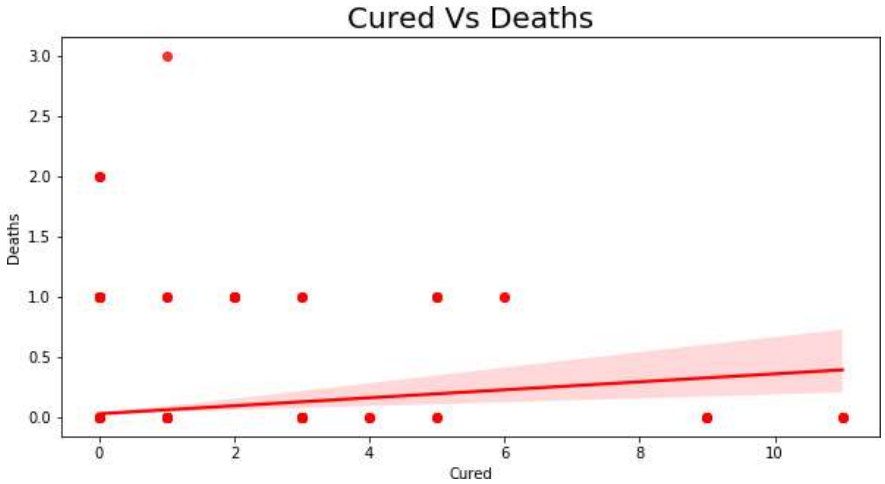
Confirmed Cases and Deaths Regression Plot



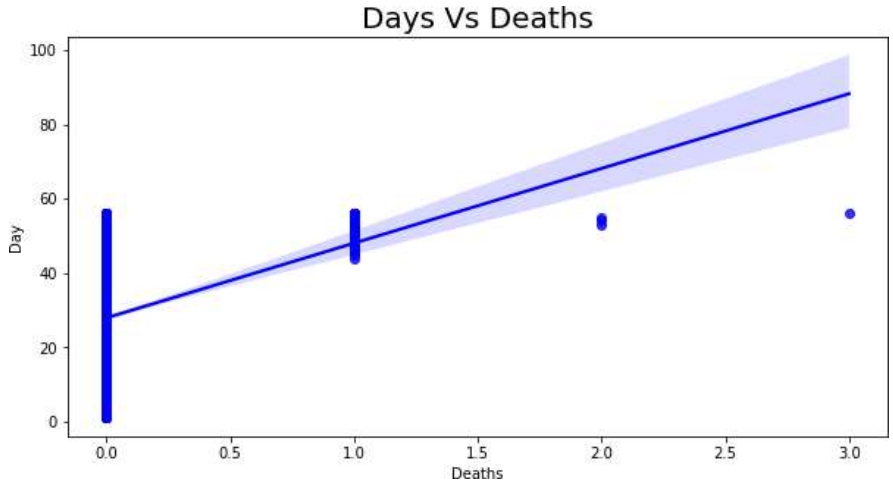
Cured Cases and Days Regression Plot



Cured Cases and Deaths Regression Plot

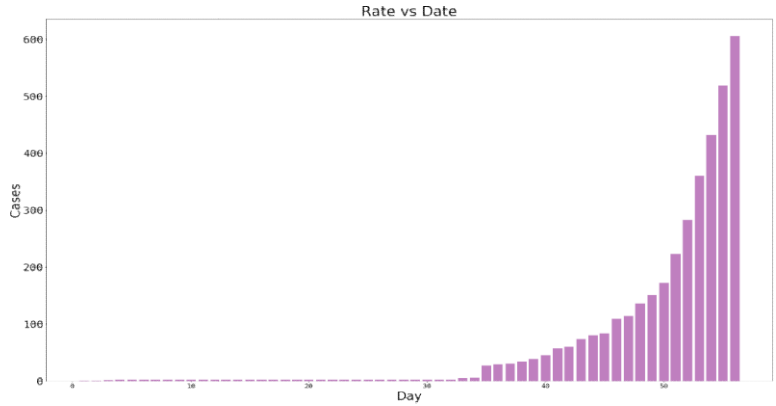


Deaths and Days Regression Plot

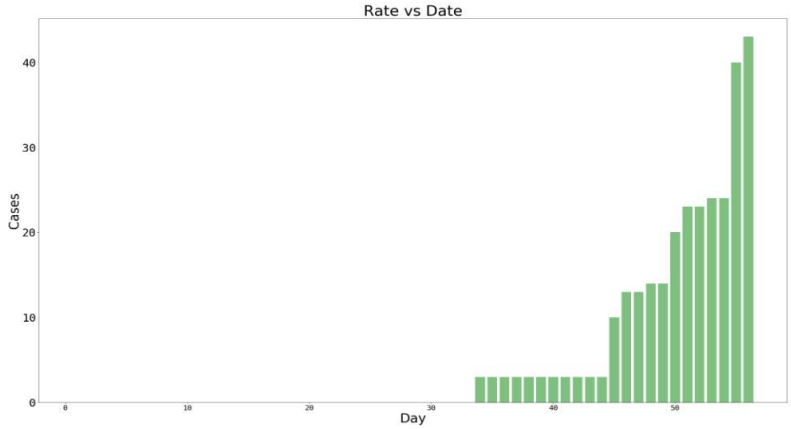


Day-wise analysis

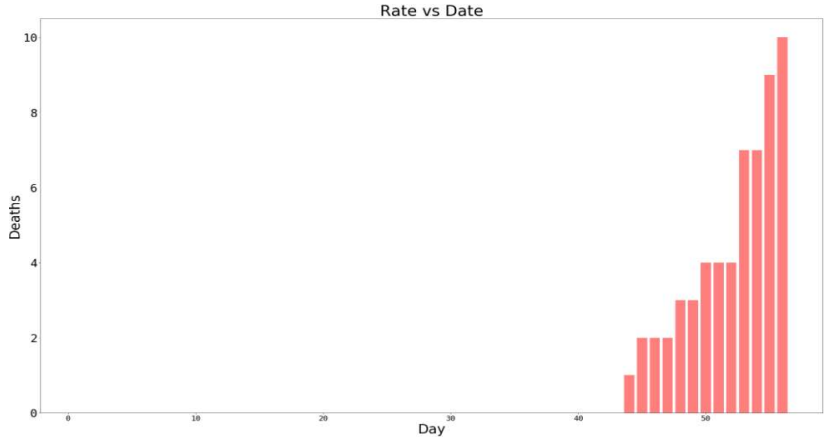
4.2.1. Confirmed Cases' Plots



Cured Rate's Plot

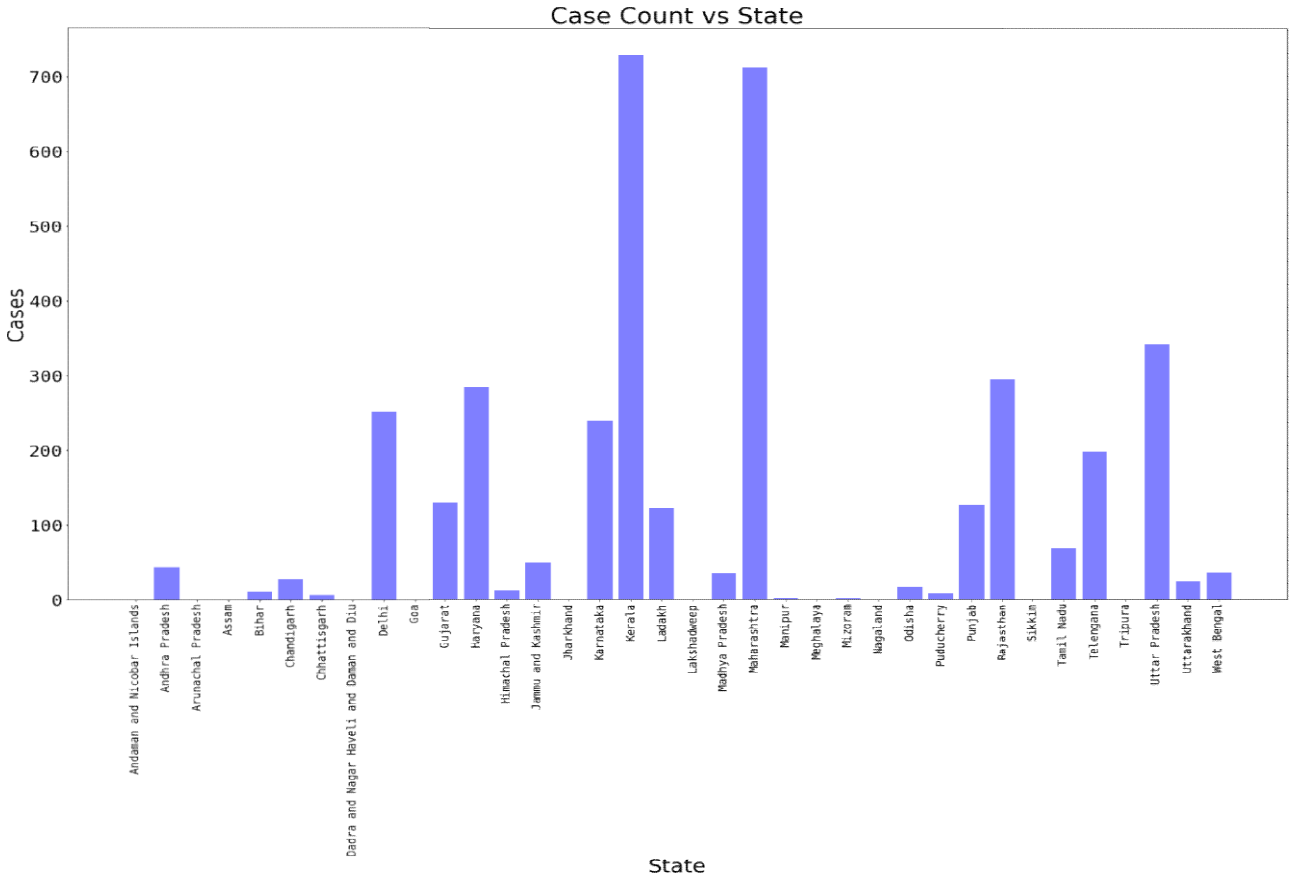


Death Rate's Plots

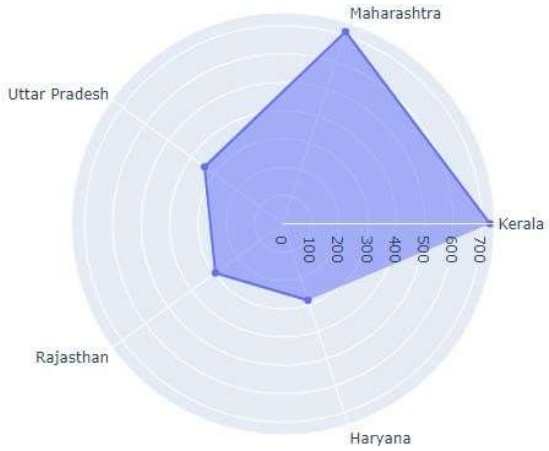


State-wise analysis

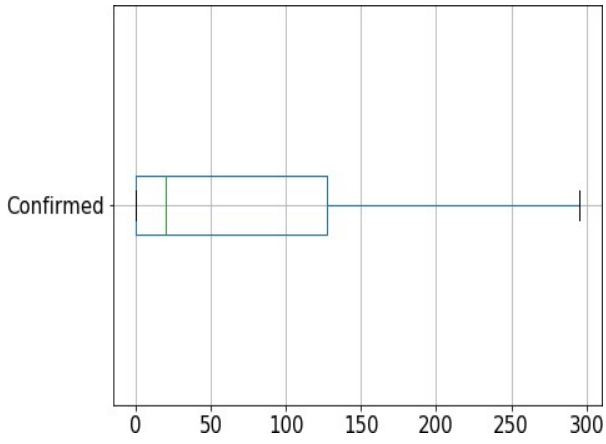
Confirmed Cases' Plots



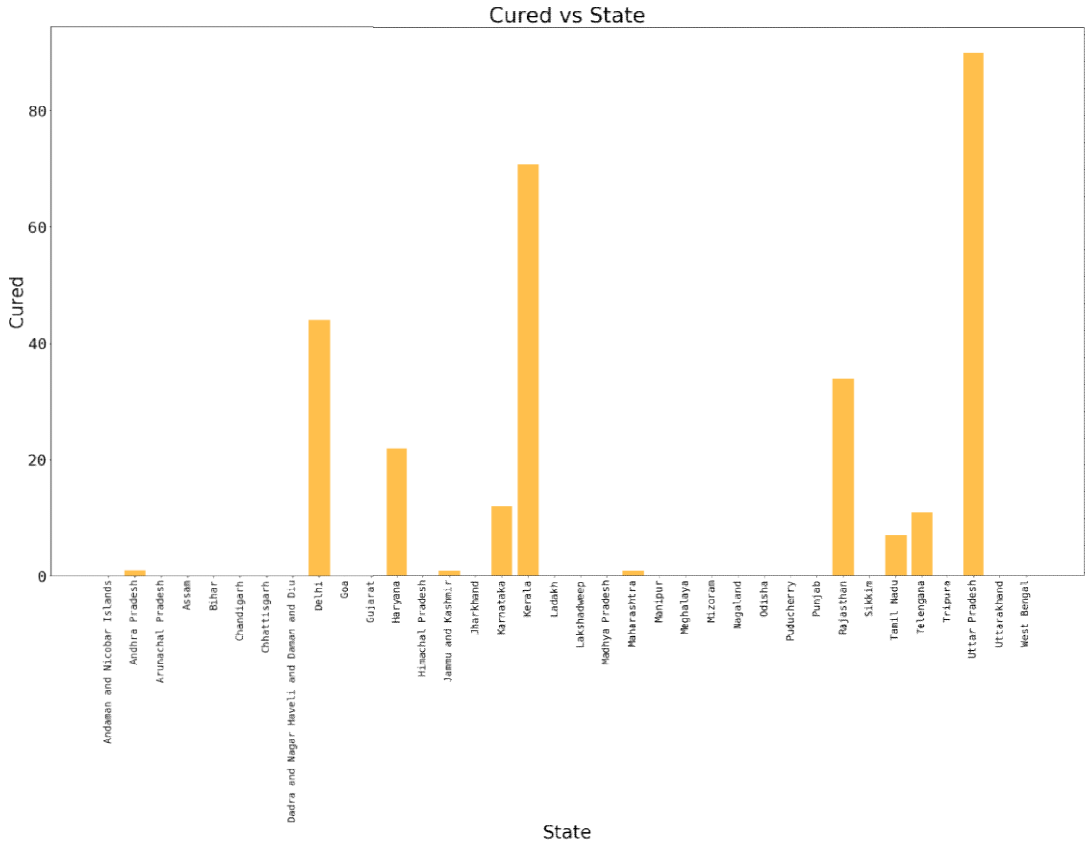
Confirmed Cases' Radar Plot



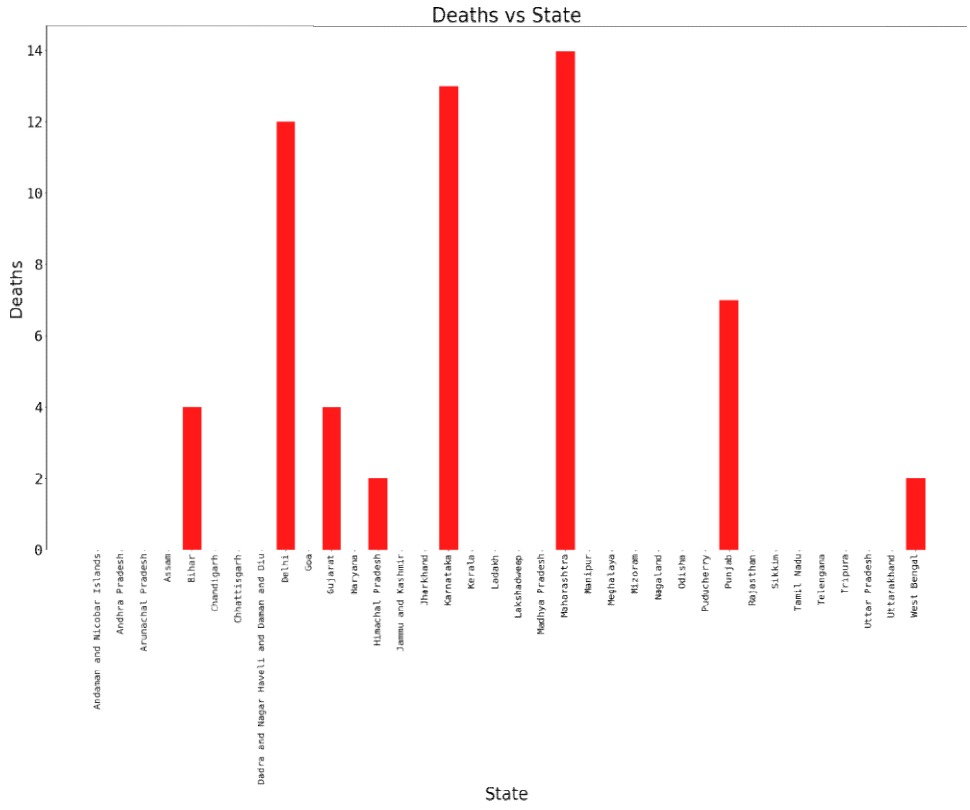
4.3.3 Confirmed Cases' Box Plot



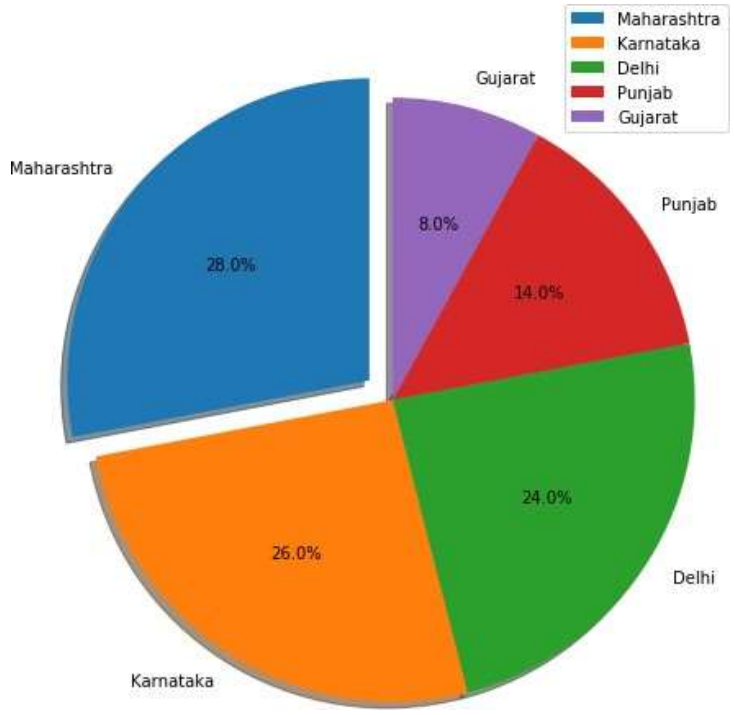
Cured Rate's Plot



Death Rate's Plot

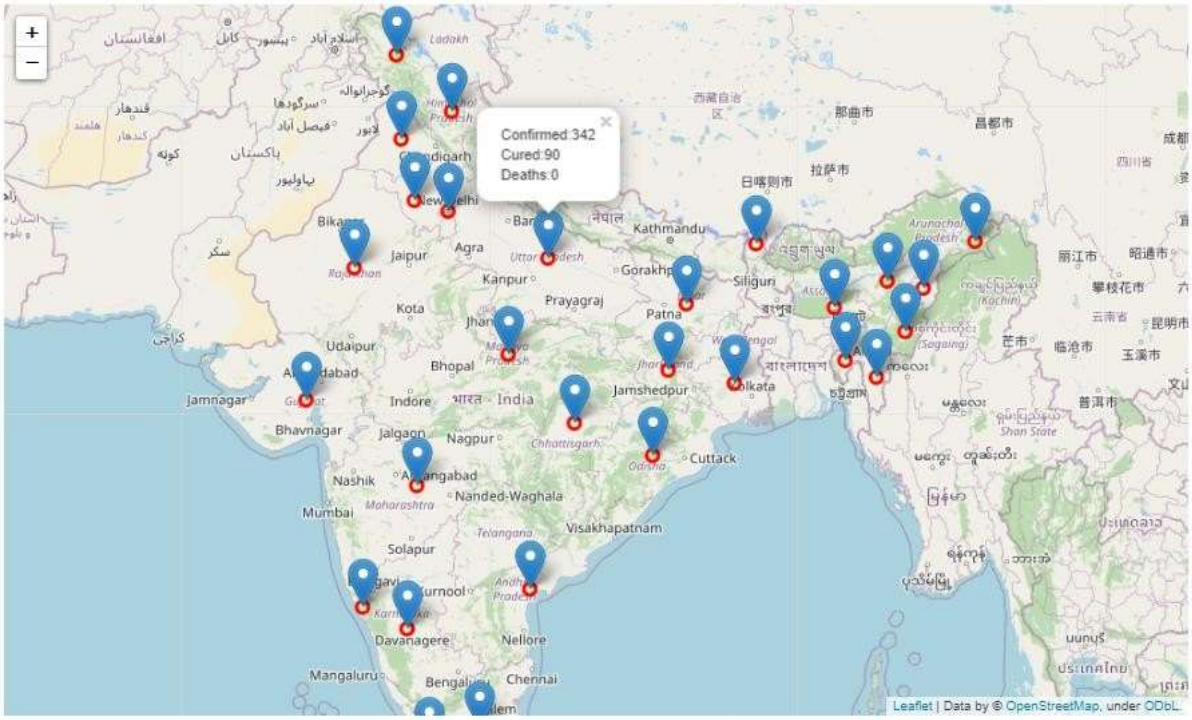


Death Rate's Pie Chart



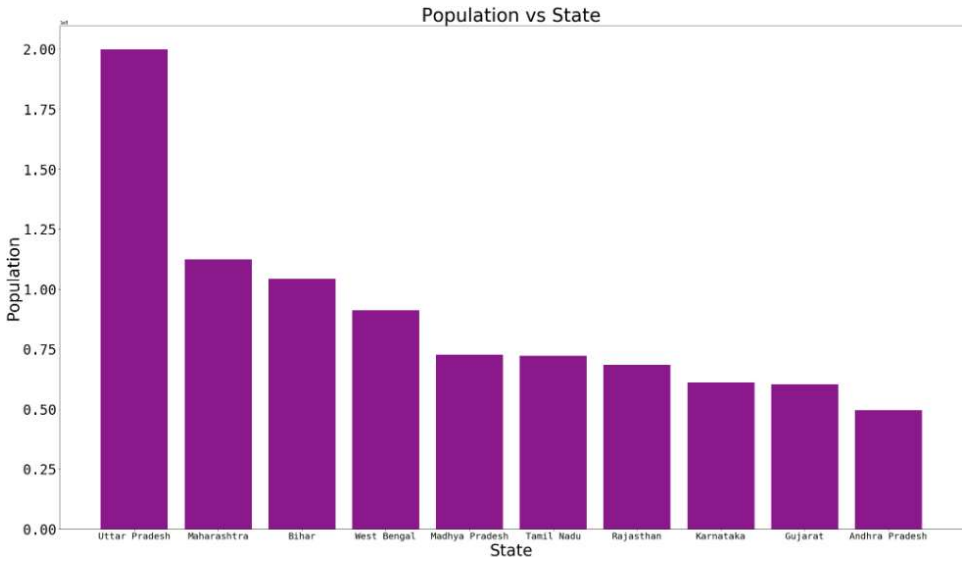
This indicated that Maharashtra had the most number of deaths. Karnataka and Delhi took the follow-up.

Overall Scenario

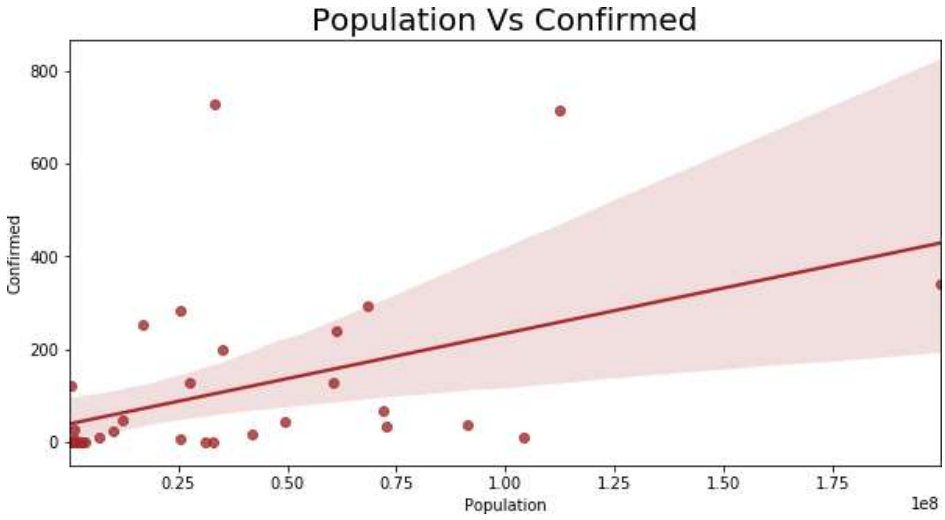


Population-wise Analysis

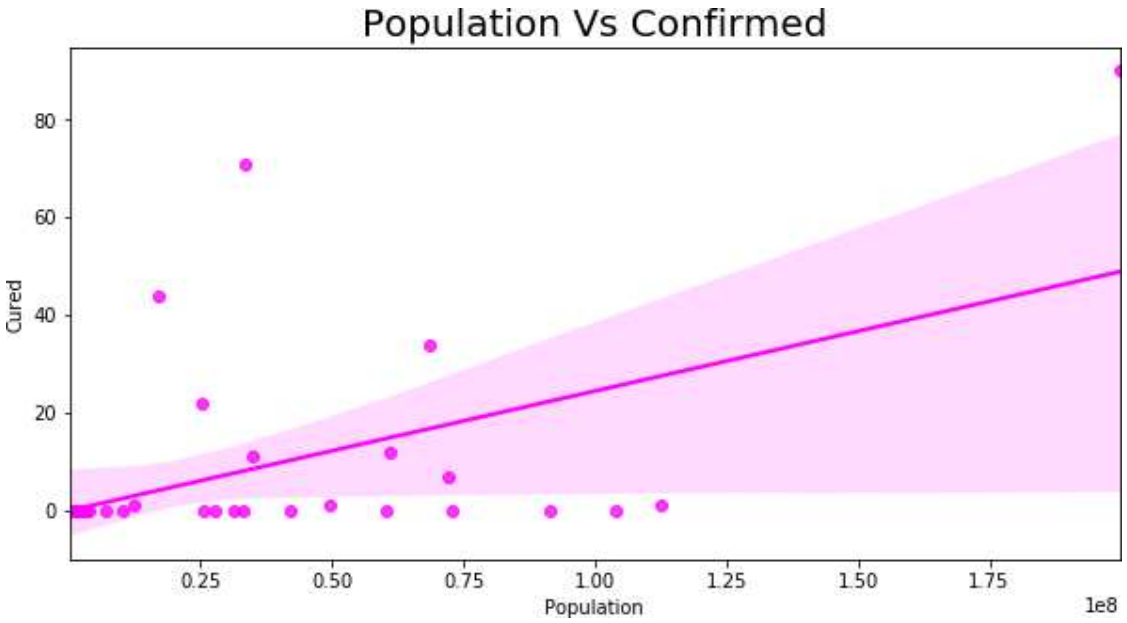
Most populous states in India (Population Vs State)



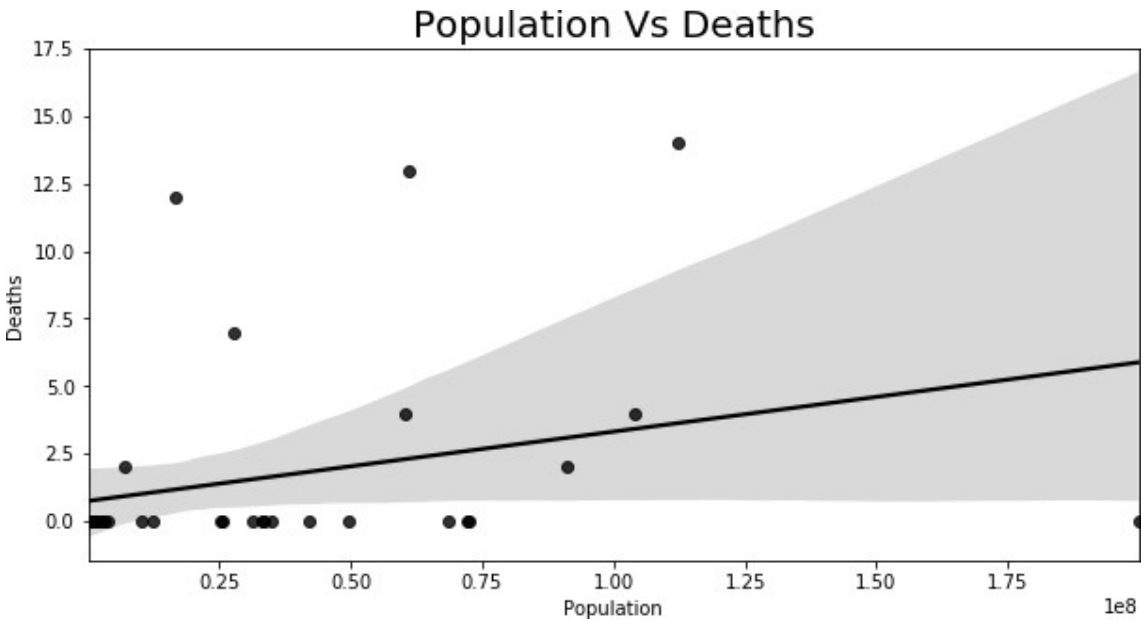
Population Vs Confirmed



Population Vs Cured

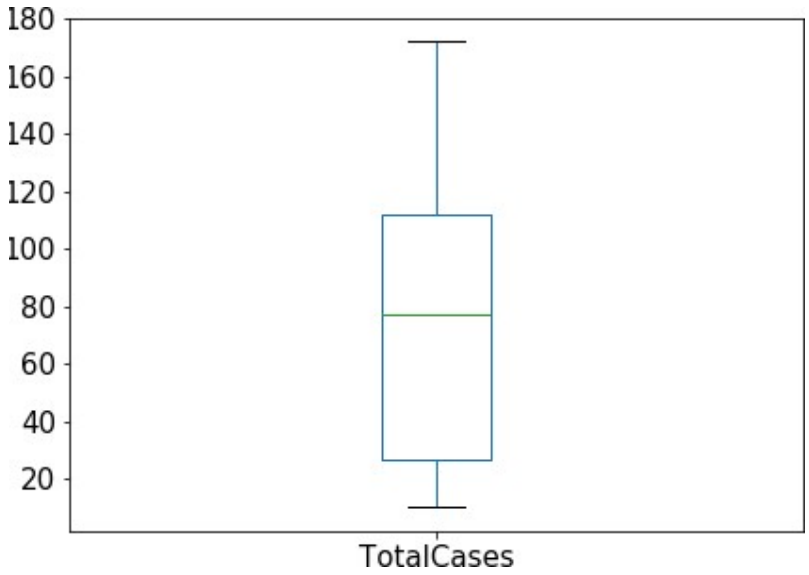


Population Vs Deaths

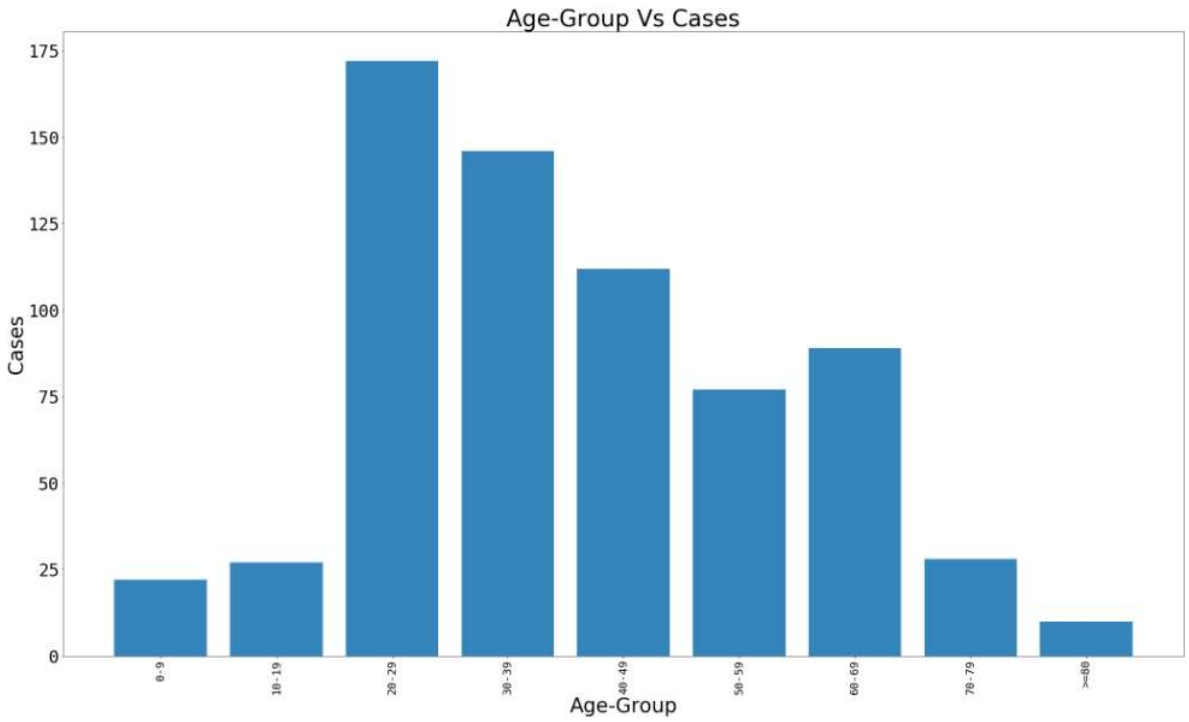


Age-Wise Analysis

Age-Group Vs Cases Box Plot



Age-Group Vs Cases Bar Graph



5. Data Modeling

Here, we use the **Exponential Regression** algorithm so as to predict the number of possible confirmed cases, if there was no lockdown implemented.

The reason for using the exponential model is based on the plots we observed during visualization. The number of confirmed cases increased exponentially, which impelled us to select the model accordingly.

Training and testing data set:

We split the data set into 2 parts:

1. **Training Set:** Used to fit the model (here, **75%** of whole data)

2. **Testing Set:** Used to validate model (here, **25%** of whole data)

We use the **'random'** function of python so as to select the dataset for training and testing purpose, randomly.

Transforming Data:

Here, we extract only the values of the columns so as to fit the regression model. The fitting is done using the **'scikit-learn'** module which only supports a list of values for fitting the model, not data sets.

The **'values'** method supported by python, helps us to extract and transform the data as required.

Defining the exponential function:

We define a function (here, with name **'sigmoid'**) which finds a relation between the number of days and confirmed, cured and death cases in those particular days.

The function takes 3 arguments: **x**, which is the number of days, and two parameters of the equation, '**a'** and '**b'**. It return **'y',** i.e. the number of confirmed cases in the particular day, based on the equation defined by us.

The equation used in this function is:

$$y=ae^{bx}$$

Finding optimized parameters:

To get the best values for ‘a’ and ‘b’, we use a function called **‘curve_fit()’** defined by scikit-learn. We input our sigmoid function, with training dataset and get the optimized parameters in result.

We do this for all, i.e. confirmed, cured and death cases. Hence, we obtain 6 results in total.

```
Optimized parameters are:

For Confirmed:
a: 0.024636474551803556
b: 0.1806252301712788

For Cured:
a: 0.006491019205331347
b: 0.15694857422845598

For Deaths:
a: -2.4800922367228083e-16
b: 0.9999999998702547
```

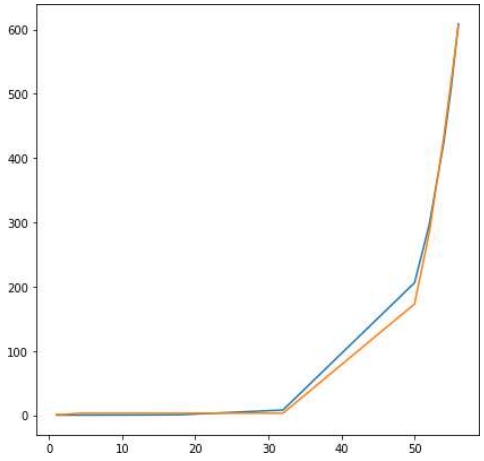
Test Predictions:

We use the ‘sigmoid’ function defined by us to find the predicted values and get the accuracy of model. We do it for all three columns, i.e. confirmed, cured and death rates. We use our test dataset this time.

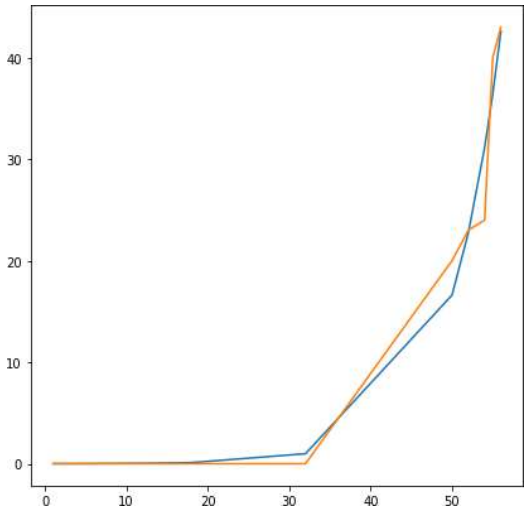
Comparisons:

After making predictions, we compare our results with the actual values to get an estimation of our model’s accuracy. We do this via visualization, where we plot the actual as well as predicted values on the same curve.

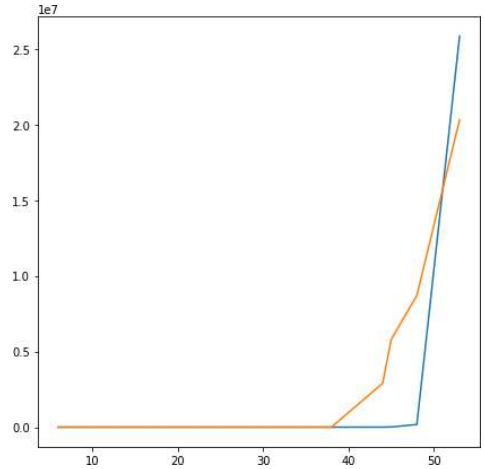
Confirmed Cases:



Cured Cases:



Death Cases:



Evaluation:

After comparison, we evaluate the error and the accuracy of our model for each column.

We use the following factors:

1. **Residual Sum of Squares**
2. **Mean Absolute Error**
3. **R^2 Score**

➔ The R^2 Score gives the probability of how similar the predicted as well as the actual values are. It can e used to measure the percentage accuracy of the model

Confirmed:

```
R-2 Score: 99.77 %
Residual Sum of Squares: 101.762
Mean Absolute Error: 6.263
```

Cured:

```
R-2 Score: 97.75 %
Residual Sum of Squares: 5.087
Mean Absolute Error: 1.075
```

Deaths:

```
R-2 Score: 76.54 %
Residual Sum of Squares: 47647784733679.734
Mean Absolute Error: 1858069.864
```

Evaluation Summary:

Our R^2 score comes out to be:

Factor	R2 %
Confirmed	99.77
Cured	97.75
Deaths	76.54

Since the R^2 score wasn't acceptable for the death cases, we only intended to predict the confirmed cases for further days.

Reason for this low R2 score is that the death column had most values as 0, which didn't allow our model to fit the best.

6. Predicting Cases after Lockdown:

Now, we predict the confirmed cases for days after lockdown. This will be done using our sigmoid function and the parameters we got while fitting the model.

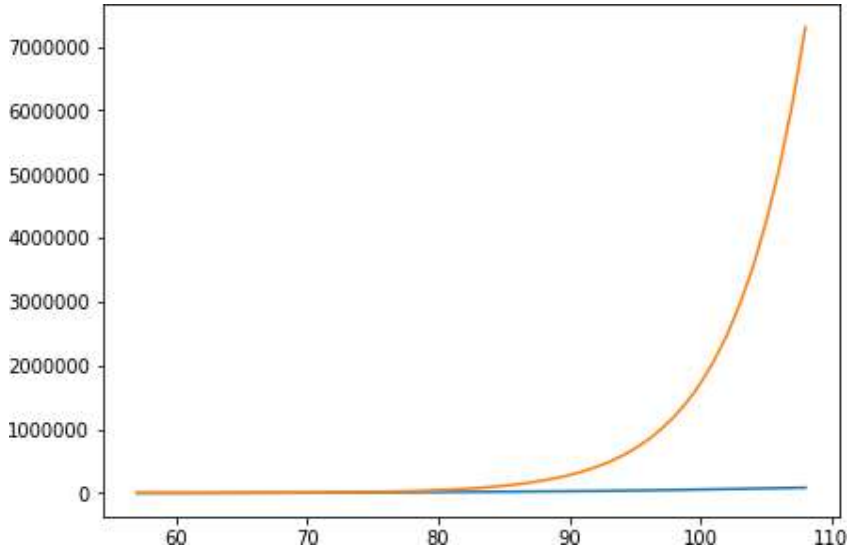
***Note: The data is restricted till first 108 days, i.e., till 16th May 2020.**

Prediction:

We predict the values using the sigmoid function. Since our previous data was based on the initial 56 days, we start making our prediction from the 57th day. Since we were able to collect data till 16th May 2020, we make our prediction till that day.

Comparison:

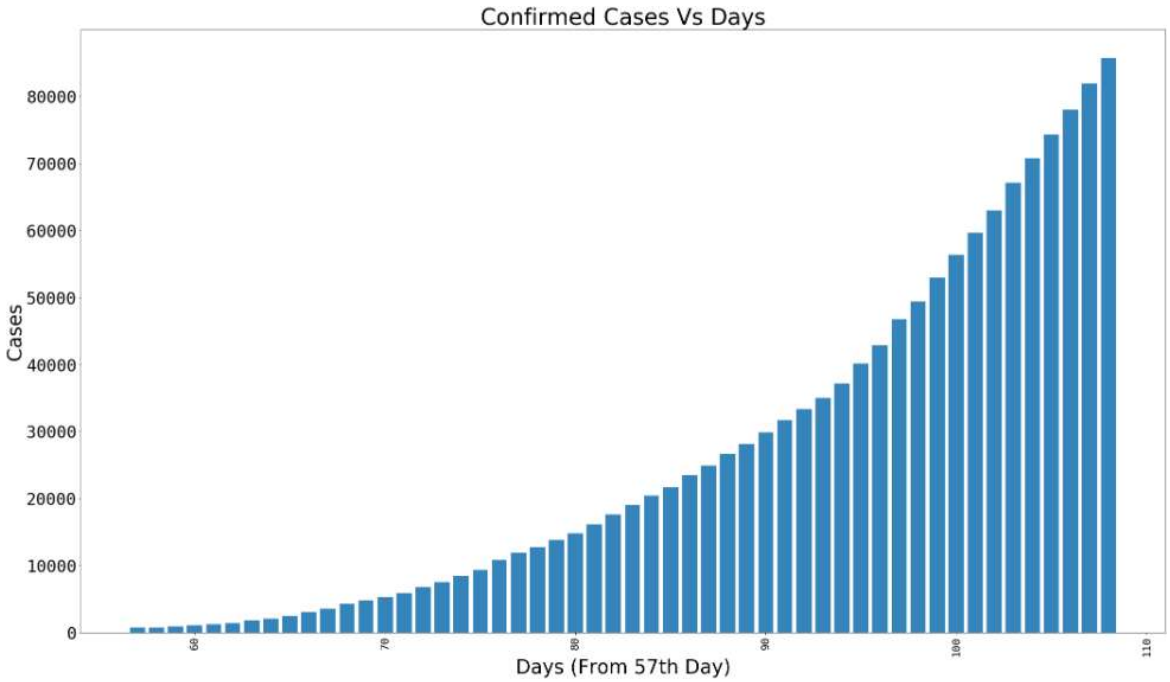
We now visualize our predicted and actual values in the form of a graph.



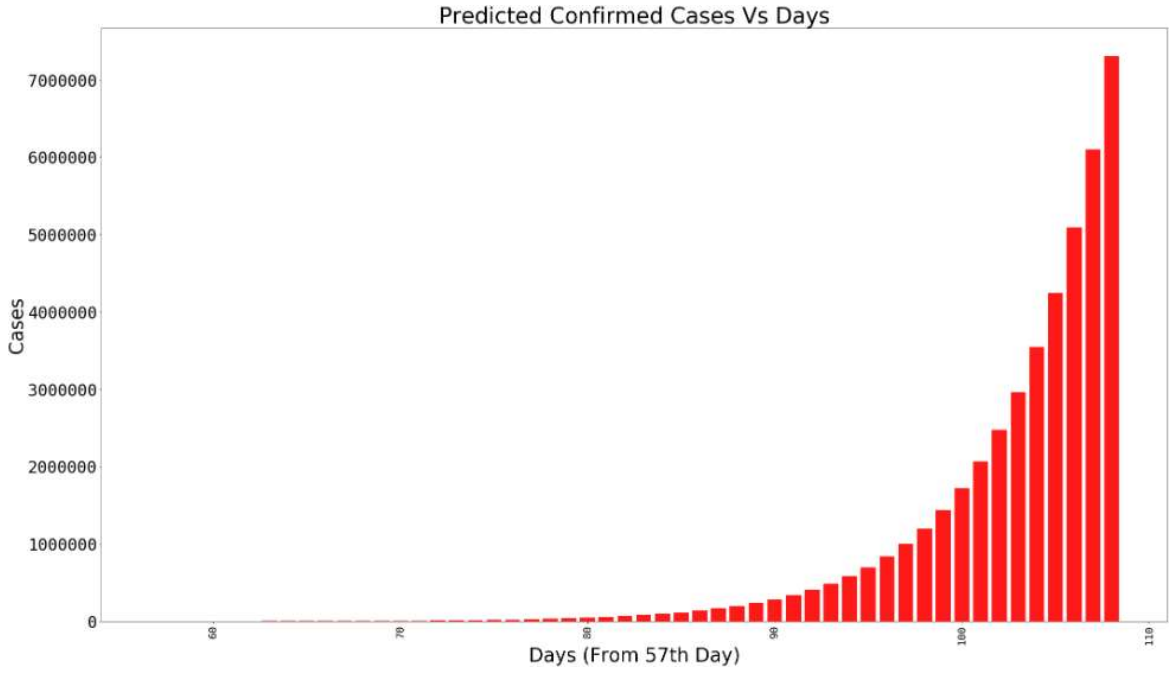
Here:

- **Orange** curve signifies the **predicted values**
- **Blue** curve signifies the **actual values**

Actual Values:



Predicted Values:



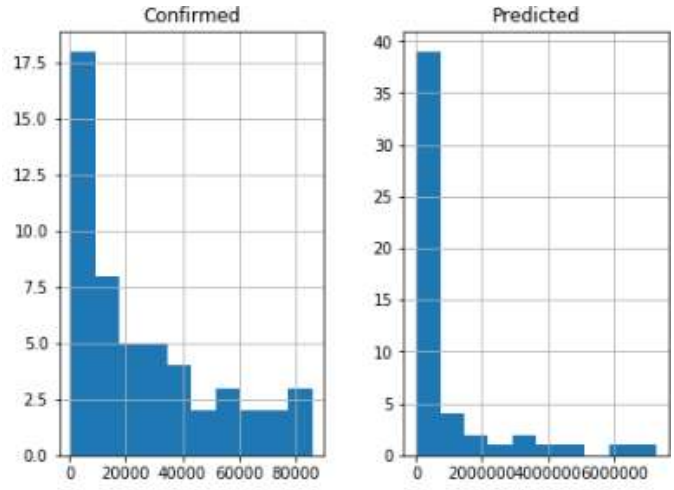
This clearly signifies that the predicted values are way higher than the actual ones.

7. Observations

Observations from Analysis and Visualization:

- Total confirmed cases before lockdown: **606**
- Total cured cases before lockdown: **43**
- Total deaths before lockdown: **10**
- Cured percentage before lockdown: **7%**
- Death percentage before lockdown: **1%**
- State with most confirmed cases before lockdown: **Kerala** (729 Cases)
- State with most cured cases before lockdown: **Uttar Pradesh** (90 Cases)
- State with most death cases before lockdown: **Maharashtra** (10 Cases)
- State with most infected/population rate before lockdown: **Ladakh**
- State with most cured/population rate before lockdown: **Delhi**
- State with most demise/population rate before lockdown: **Delhi**
- Most infected age-group before lockdown: **20-29** (172 Cases)

Observations from Regression:




```
Maximum cases during lockdown: 85709
Maximum cases if there was no lockdown: 7304474
Percentage Population Saved: 98.83 %
```

8. Result:

Our estimations show that due to lockdown, **98.83%** of India's population was saved from being infected. This clearly signifies that ***lockdown was successful, and proves out to be a correct decision!***

Project URL: https://nbviewer.jupyter.org/github/DivyT-03/Covid19Analysis/blob/master/COVID-19%20-%20Notebook.ipynb