Covid-19 Pre-Treatment Recovery Analysis

Prepared by:

Tanmay Pardhi,

Government College of Engineering, Nagpur

Abstract

In this report, we address the problem of Covid-19 Pre-Treatment Recovery Analysis on patients dataset provided by the Ministry of Health and Family Welfare (MoHFW). We use a number of machine learning methods to perform the analysis. In the end, we use a majority vote ensemble method with 5 of our best models to achieve the classification accuracy of 93.33% using ensemble classifier score method.

1 Problem Statement:

The coronavirus COVID-19 pandemic is the defining global health crisis of our time and the greatest challenge we have faced since World War Two. But the pandemic is much more than a health crisis, it's also an unprecedented socio-economic crisis. Stressing every one of the countries it touches, it has the potential to create devastating social, economic and political effects that will leave deep and longstanding scars. Every day, people are losing jobs and income, with no way of knowing when normality will return. Small island nations, heavily dependent on tourism, have empty hotels and deserted beaches. The International Labor Organization estimates that 195 million jobs could be lost.

In this report we will attempt to conduct Pre-Treatment Recovery Analysis on Covid-19 Patients using different data Visualization and Machine Learning algorithms. We attempt to classify whether a Patient will recover from the Covid-19 disease prior to treatment based on the types and severity of his symptoms. The more dominant symptom being gradually given higher importance in determining whether the patient will recover or not.

We use the dataset of the patients provided by the Ministry of Health and Family Welfare and the World Health Organization which showed whether a patient has recovered and the symptoms he faced. The data provided comes with missing data too which are required to be processed and also converted into a standard form. We also need to extract useful insights from the data that will help us to find which are the most important factors necessary in order to recover as well as can be used for further research purposes.

We use various machine learning algorithms to conduct the prediction for recovery using the extracted features. However, just relying on individual models did not give accuracy so we pick the top few models to generate a model ensemble. Ensembling is a form of Meta learning algorithm technique where we combine different classifiers in order to improve the prediction accuracy.\Finally we report our experimental results and findings at the end.

2 Data Description

The data we have is in the form of comma-separated values files with the patient's symptoms and the information about the patient. The training dataset is a csv file of type "covid_19_data, covid_19_india, ICMR Testing Labs, Individual Details, State-wise Testing Details". covid_19_data consists of covid-19 related date-wise data of patients, covid_19_india is made up of covid-19 related date-wise data of individual patients, ICMR Testing Labs consists of details about the private and Government Testing labs set-up in India, Individual Details show the details of patients and how they got covid-19 syndrome.

The dataset consists of the following feature-sets

	Sno	Date	Time	State	ConfirmedIndianNational	ConfirmedForeignNational	Recovered	Deaths	Confirmed	Active
35	36	2020-03-03	6:00 PM	Telengana	1	0	0	0	1	1
36	37	2020-03-03	6:00 PM	Rajasthan	0	1	0	0	1	1
37	38	2020-03-03	6:00 PM	Kerala	3	0	3	0	3	0
38	39	2020-03-03	6:00 PM	Delhi	1	0	0	0	1	1
39	40	2020-03-04	6:00 PM	Uttar Pradesh	6	0	0	0	6	6

Table 1: Sample of data used for visualizing the growth of Covid-19

So, the dataset consists of 4671 rows of data consisting of data from 30-01-2020 to 30-07-2020 and the data from 28 states and 8 Union Territories. We've used the **pandas** library and the **Plotly** *Open-source* Library to visualize our data. So from these data we visualized the Reported cases in India over the above-specified time.

3 Utility:

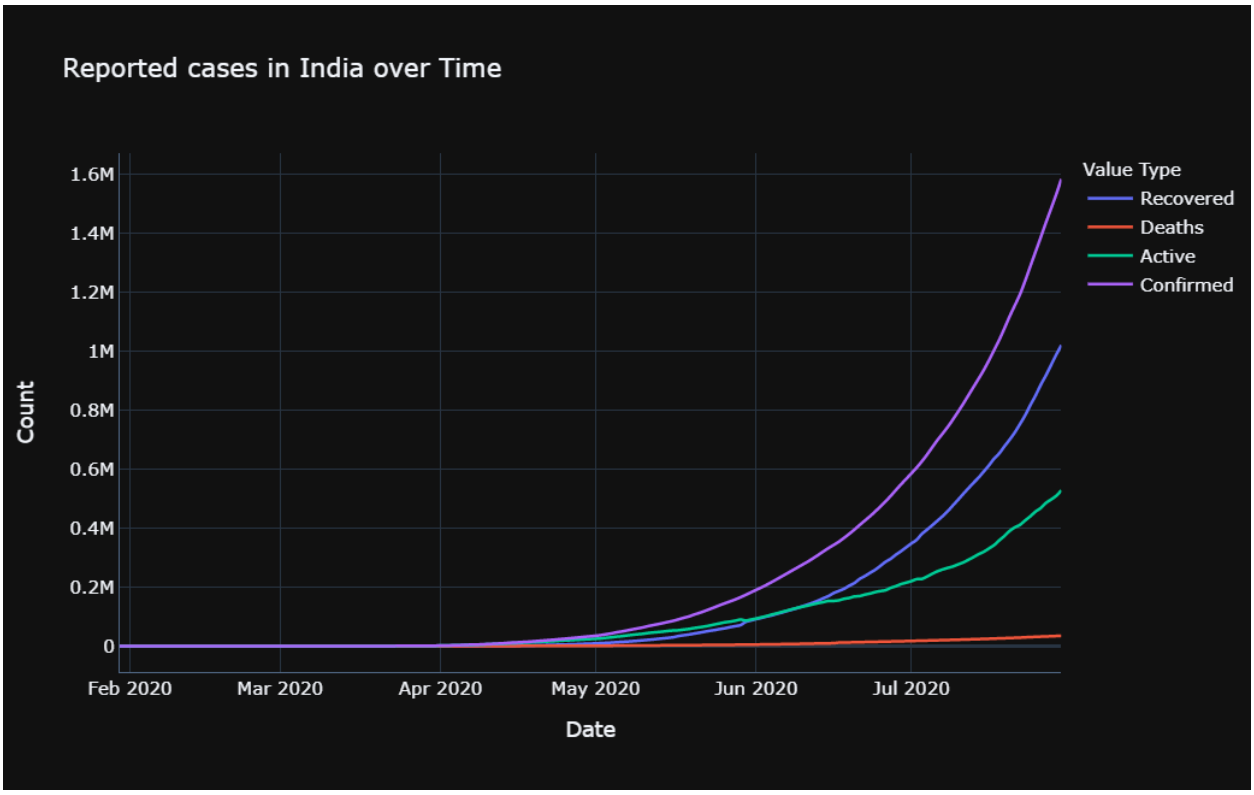

Figure 1: Confirmed, Active, Recovered & Deaths over Time in India

We can see from the data the above visuals that except for death the active, confirmed and recovered cases surged at a very fast rate from mid-April till the current date. The Positive insight from the above data is that Deaths haven't rise at the same speed as that of other features. The other insight that we can get from this is that from **10-Jun** the Recovered patients overtooked the Active cases.

The above graph is a live model where you can analyze the data in real-time and you can hover, add spike-lines, auto-scale, pan, zoom-in, zoom-out, reset axes and capture images in real time.

Now, let's use the data to find the percentage of the cases in the specified feature cases. We create a Pie-chart of the data using **graph_objects** and **pandas.**

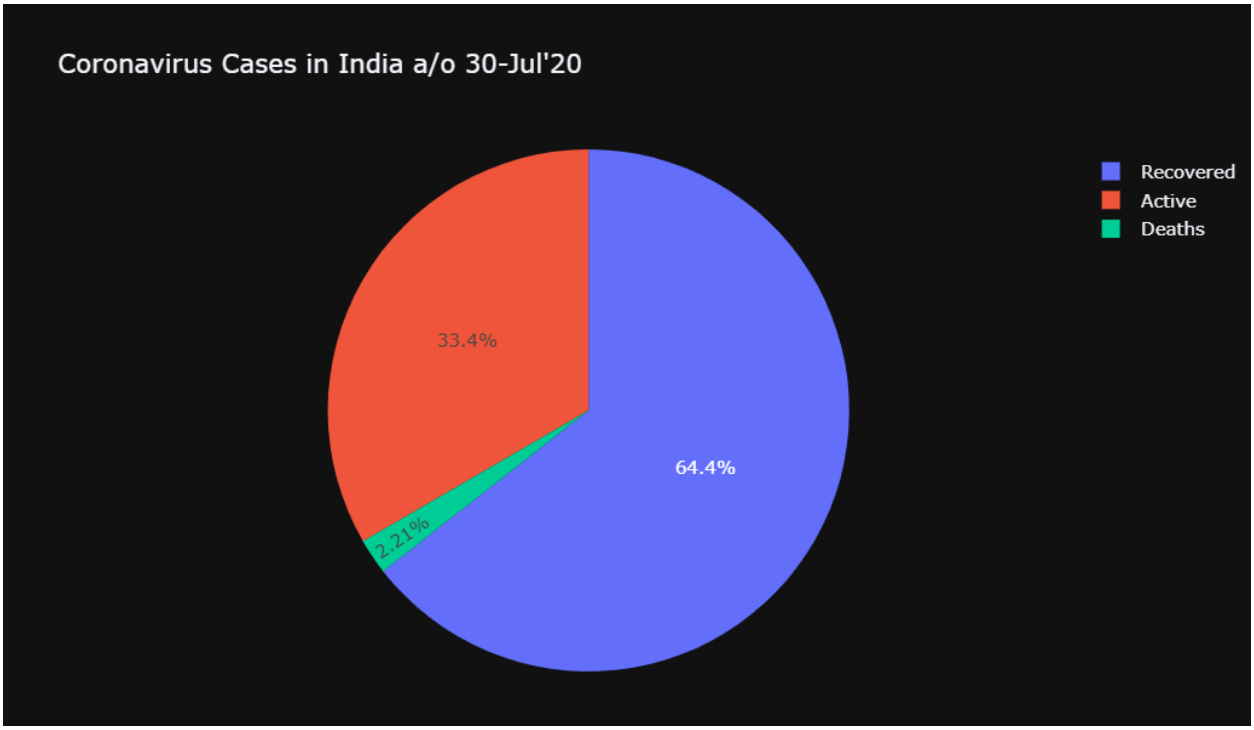

Figure 2: Pie-chart of Coronavirus recorded till 30-07-2020

Recovered Patients - 64.4%

Active Patients – 33.4%

Deaths – 2.21%

So the above visuals show how the cases and their share in the total cases exist and how the government machinery needs to perform on the basis of the percentage share of respective patients. The Government can even plan on how many more beds they will need for proper management of the patients, plus can pre-predict the upcoming challenges in the near future.\

The next thing which we have done is created a mp4 file which shows live modeled video consisting of bar chart race among the Active, Recovered and Deaths Date-wise of the patients. Considering it is a back-end model the video can't be shown in the report but we've attached a still from 13-06-2020.

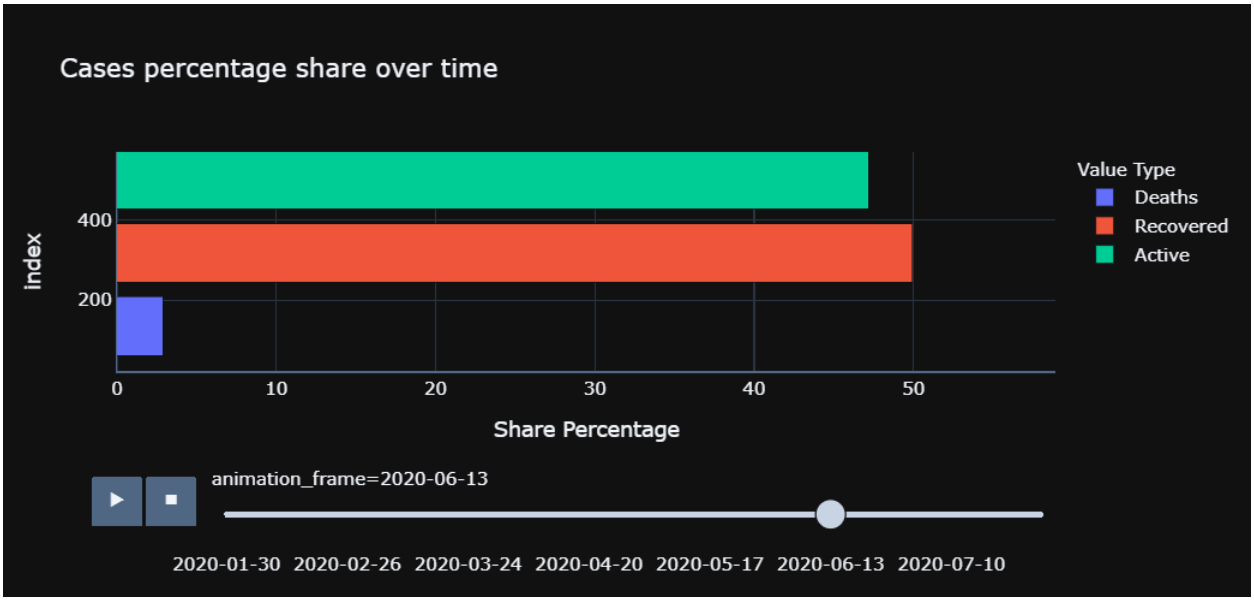

Figure 3: A still from the Bar-chart race among the feature sets.

The above video shows how the cases have fared up and how the recovery rate and active cases fare up with each passing day. From this video we get the insight that from **10-Jun** the recovery rate overtakes the active cases.

The above video can be very useful for research purpose as it can show the percentage share with each passing days.

The below shown figure shows cases recorded per day, we can correctly analyze how the cases increase with each passing day easily, As can be seen from the data that right from 03-07-2020 i.e. 3 July 2020.

While Plotting the above data no standardizing technique was used to better analyze the data. To get a smooth curve for the data we can use standardizing technique like logarithmic or scaler scaling.

Continued below is the exponentially standardized graph cumulatively of the past 14 days mean of the Coronavirus Patients. The exponential data is selected as if any new patient who is diagnosed by Covid-19 syndrome mostly gets fine in 14 days from the day he/she gets affected with Covid-19 syndrome. The Exponential data may not be that useful for research purpose but is easier for analyzing purposes. Viewer can get a clear idea how the graph of increase in covid related patients is increasing/decreasing as well as in what manner the graph is showing it's pattern (Exponential in our case).

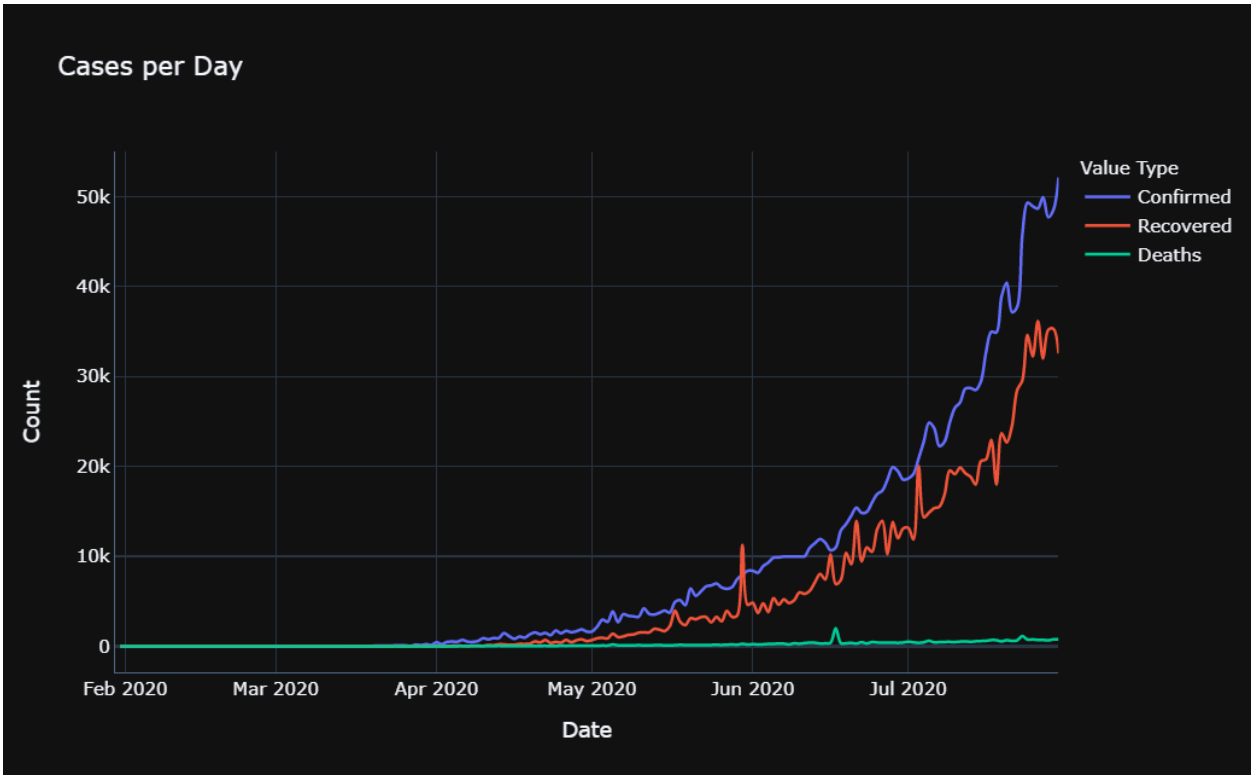

Figure 4: Unstandarsized Cases per Day

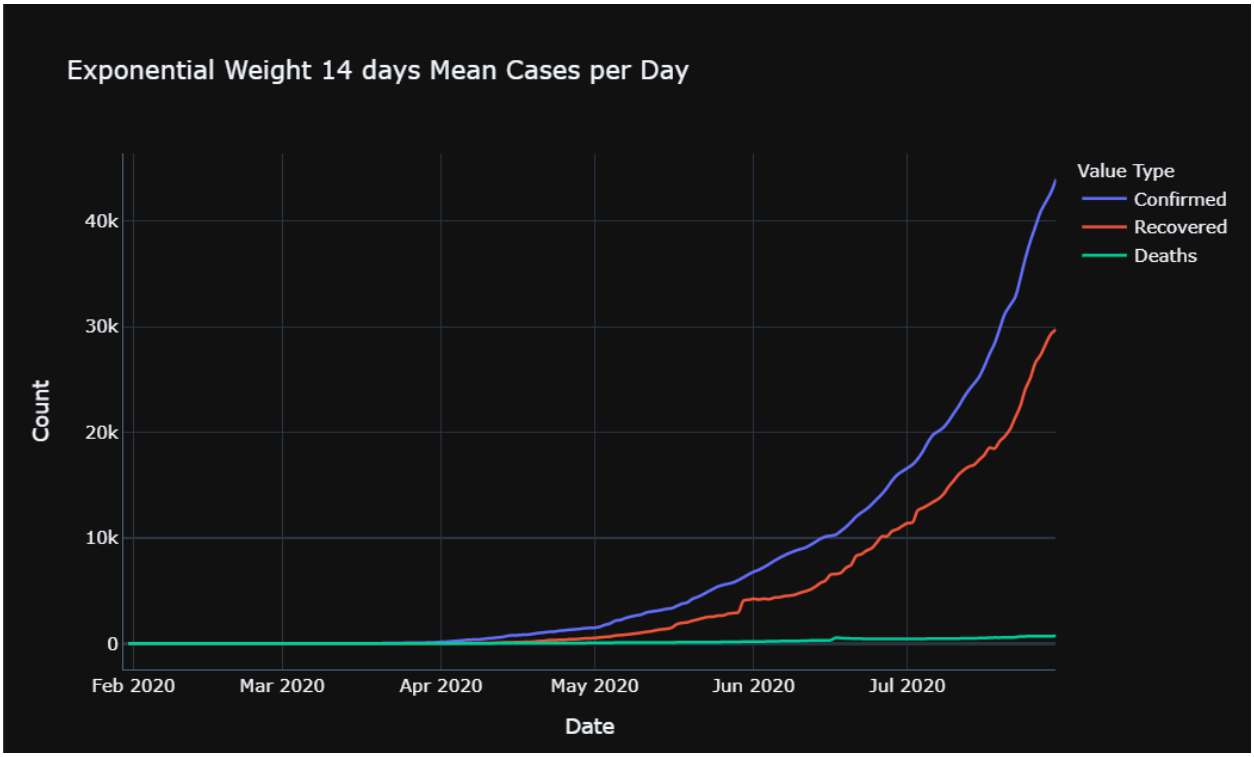

Figure 5: Exponentially Standardized Cases for 14 days mean

Now since we have talked this much about Covid-19 in India, A common Question that will arise is how the states and union-territories are being affected due to this Pandemic. The Union of India consists of 28 States and 8 Union Territories with each of varying size in area as well as population. So it is obvious that the impact in the states will also vary. In-fact in a state too, some areas are urban and some are rural areas so this will also affect the spread of Covid-19. As per the analysis cases in metropolitan cities are very much more in comparison to other cities.

So to analyze how cases are surging in Different states we use a tree-map to visualize the data in a single frame of screen. It is a live model which shows the count of cases in different states when we hover our cursor over it. Plus we can open to better visualize the data and we can also analyze the data color specifically, like we can give different colors for different range of active patients.

Ex: Since the cases have increased above 140K in Maharashtra that specific block of it will be shown in Yellow color

Similarly the cases in Tamil Nadu are close to 60K so the specific block of it will be shown in crimson color.

When a user will hover over a particular state the active cases and the state name is going to show as the data associated with the tree. For the states which can't be seen because of very less cases can also be seen with just a click on the particular block.

There are some inaccurate data provided in the dataset too like Telangana is misspelled as Telengana/Telangana as well as since some data are provided by the state government later after post-analysis then they are provided with the state name and asterisk i.e Telengana***/Telangana***

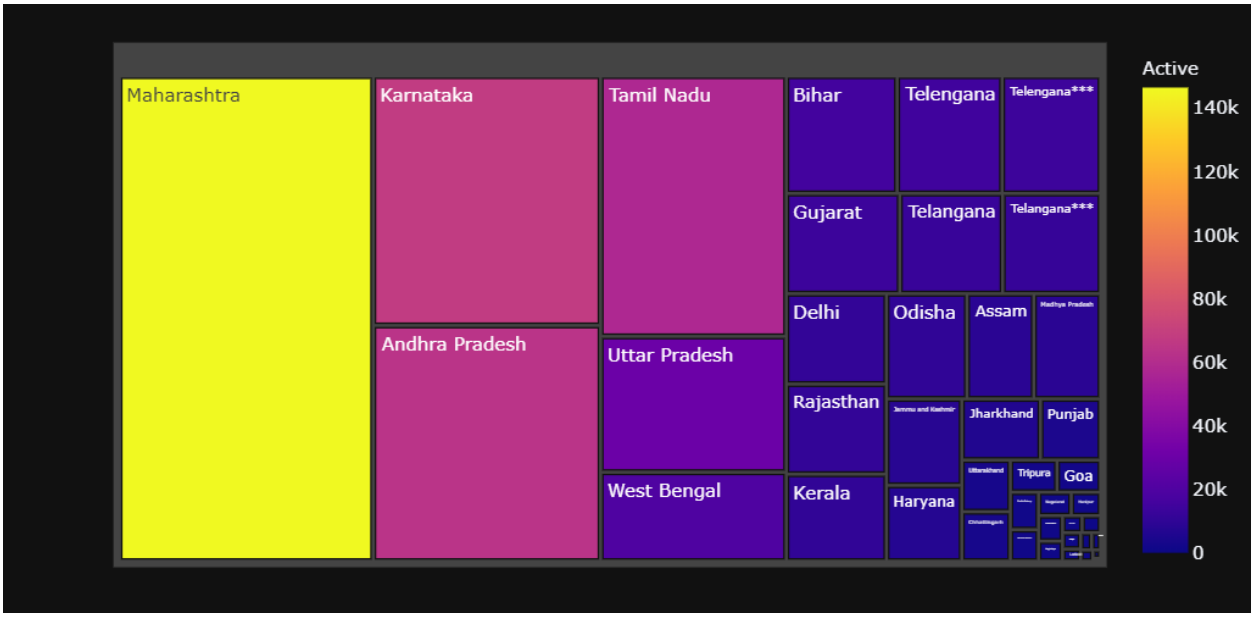

Figure 6: State-wise Covid-19 Hovering Dashboard

Next we create a Bar chart race from the state Covid-19 database. We take the data right from when the pandemic started in India on 30-01-2020 right till 30-07-2020.

This bar chart provided it's first insight that the recovery rate of Kerala was 100% until 07-03-2020 and then hit a very low 18% on 12-03-2020 showing a sudden increase in the Covid positive cases of that state.

As we watch further watch the mp4 file at the end we notice that currently the highest recovery percentage is in the state of Delhi i.e 89% and upon doing the further research we find that from being the second highest covid impacted state/union-territory to the highest recovery is due to the combined effort of the central as well as the state government and the plasma therapy and all the initiatives the governments took.

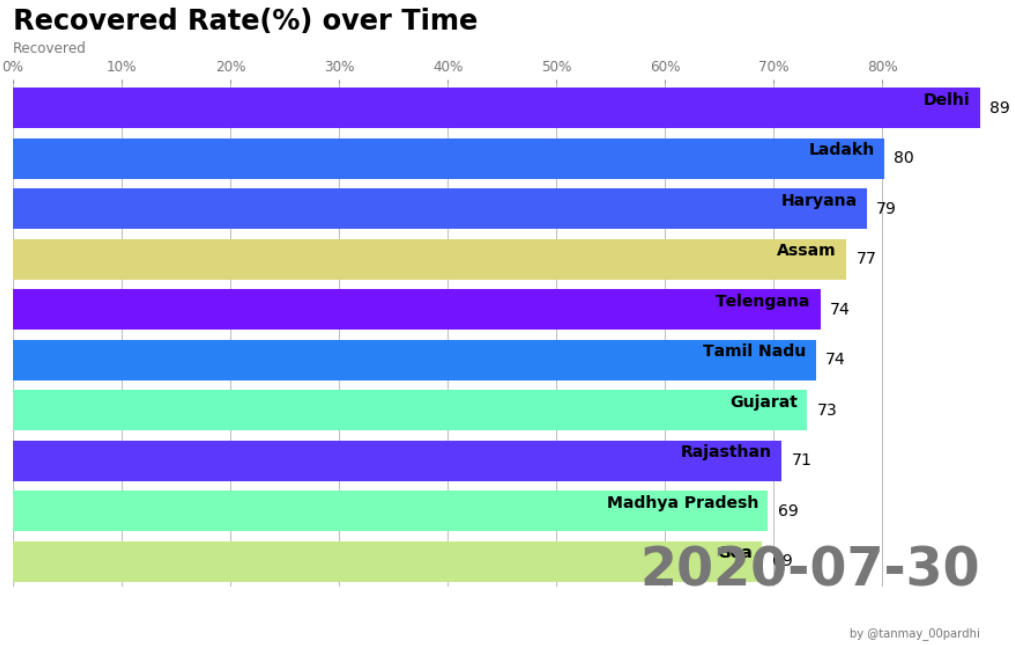

Figure 7: Still from the recovered rate over time video

Indian Council of Medical Research (ICMR) has provided the data on testing Labs present in each state. This data consists of Government Lab as well as privately owned labs too.

So we create a tree-map for this data which is further more subdivided into different states and further branched into cities which get furthermore divided into Government Labs and Private Labs which open up into themselves showing the count of the respective feature into the selected city and state.

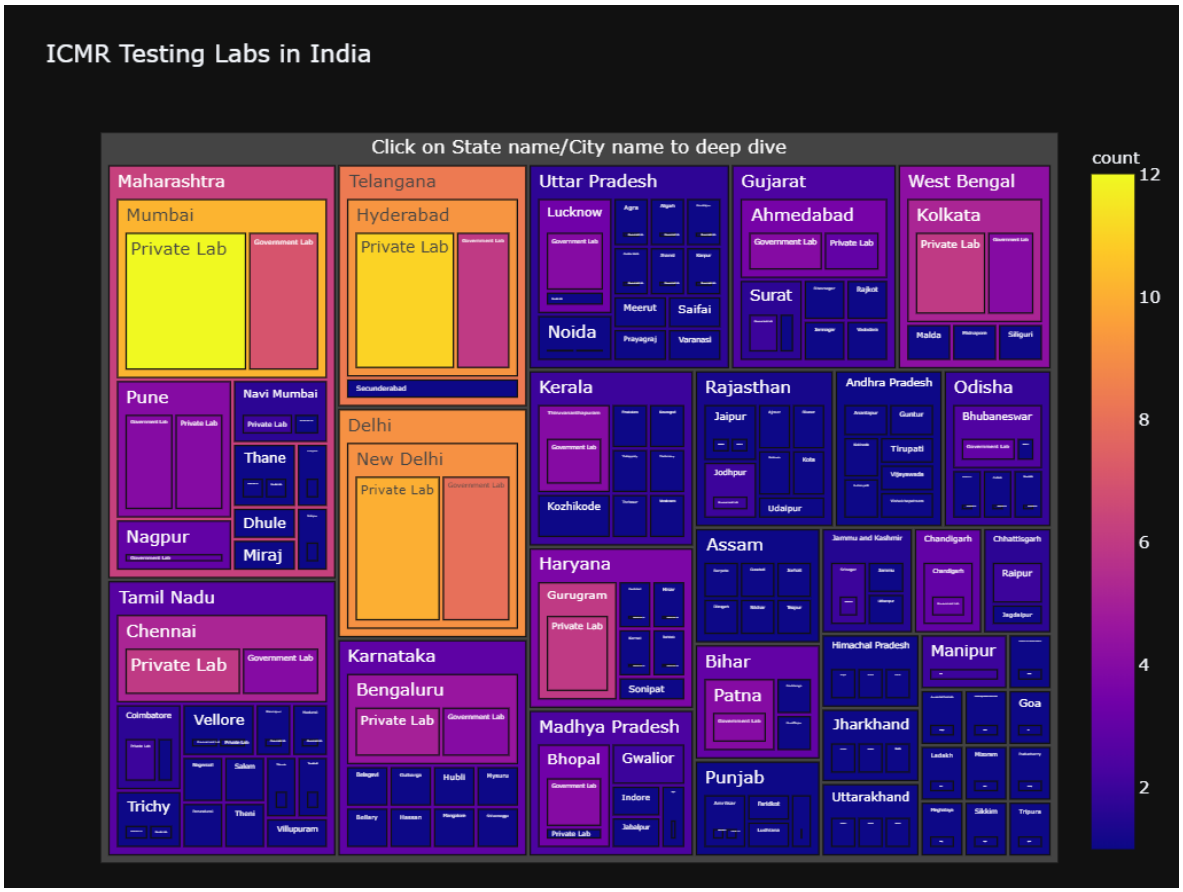

Figure 7: State-wise Covid Testing Labs

Now on hovering above the state blocks and on entering a particular state we can see the different cities and the number of different private labs and government labs.

As we can see from the below data that the most no. of labs in the countries are located in the state of Maharashtra followed by the state of Tamil-Nadu and the state of Telangana.

Each state is then subsequently ordered as per the total number of labs present in the state. This can prove to be a great feature in modeling the growth of Covid-19 as the more number of testing labs the more the discovery of corona-virus and the less the chances of transmission.

Figure 8: City-wise Covid Testing Centre

Now if the cases are surging in the countries at such a faster rate a gradual thing is how the testing rate in the countries increasing with respect to the rising number of cases.

As per WHO, the more the number of testing in the country, the more the number of cases and lesser the transmission.

So let's check how our country is faring in our country in case of testing capabilities.

For checking the Testing ratio with active cases we use the **plotly library's express feature.**

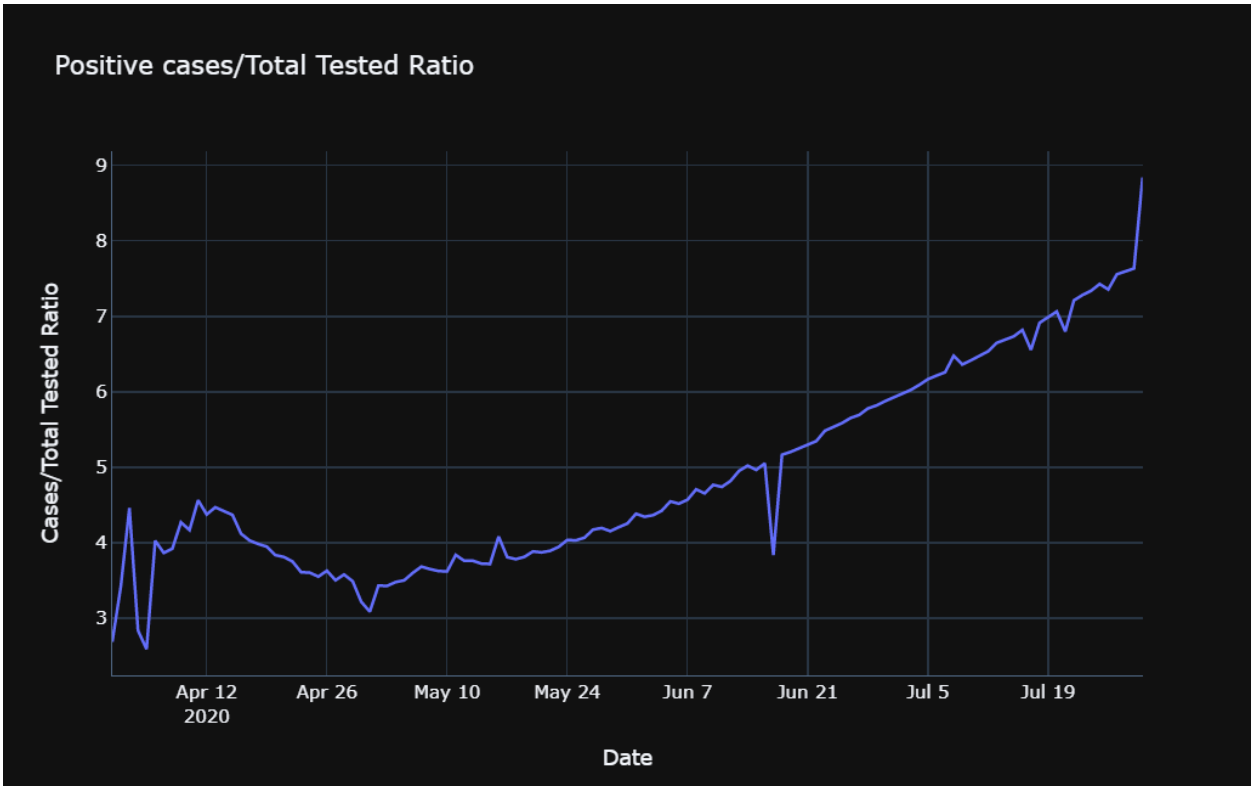

Figure 9: Date-wise Positive cases / Total Tested Ratio

The upward trend of graph shows that cases are increasing much faster than the testing capabilities of the country.

$$\text{Ratio} \propto \text{Rising Cases}$$

$$\text{Ratio} \propto 1 / \text{Total Testing}$$

The above equation indicates that as the ratio will increase the number of cases will also increase whereas as the ratio increases the testing is not up to the mark and is proving to be insufficient with respect to what is needed at the current time.

So, this visualization shows that India is currently lagging in terms of testing and it needs to increase the testing capabilities at its earliest in order to contain the virus at its earliest.

Now we use the Plotly's Histogram feature to extract gender-wise data of patient's suffering/suffered from Covid-19.

It would be great to know what's the minimum and maximum age of patients in each frame as well as the median of age group of each gender.

The data provided by the Ministry of Health and Family Welfare consists of lots of data without gender classification and filling data with approximation is not that precise so we create another set of the name N/A.

From this data we conclude that:

Females diagnosed with Covid have a median age of 35.

Minimum age recorded of female diagnosed patient is 3 months.

Maximum age recorded of female diagnosed patient is 89 years.

Males diagnosed with Covid have a median age of 35.

Minimum age recorded of male diagnosed patient is 4 months.

Maximum age recorded of male diagnosed patient is 98 years.

Figure 10: Age/Gender share of confirmed case

Recovery time distribution is found to be quite different in both the genders. While males usually have a recovery time of 14-15 days, females usually take 16-17 days to recover.

The shortest and longest duration of recovery in females is 8 & 22 respectively.

The shortest and longest duration of recovery in males is 7 & 29 respectively.

Figure 11: Gender-wise Recovery Time Distribution

So now basically what we are going to do is to compare the graphs of covid cases in China over time with the covid cases in India over time. What we understand from the graph of China is that they too had an exponential growth during the initial phase of spread and the inflection point of Chinese Mainland spread of Corona-virus was 12-02-2020 and the graph started to get a straight line from 03-03-2020 and since then the graph has mostly been a straight line which shows that the spread of coronavirus in China has been under control now.

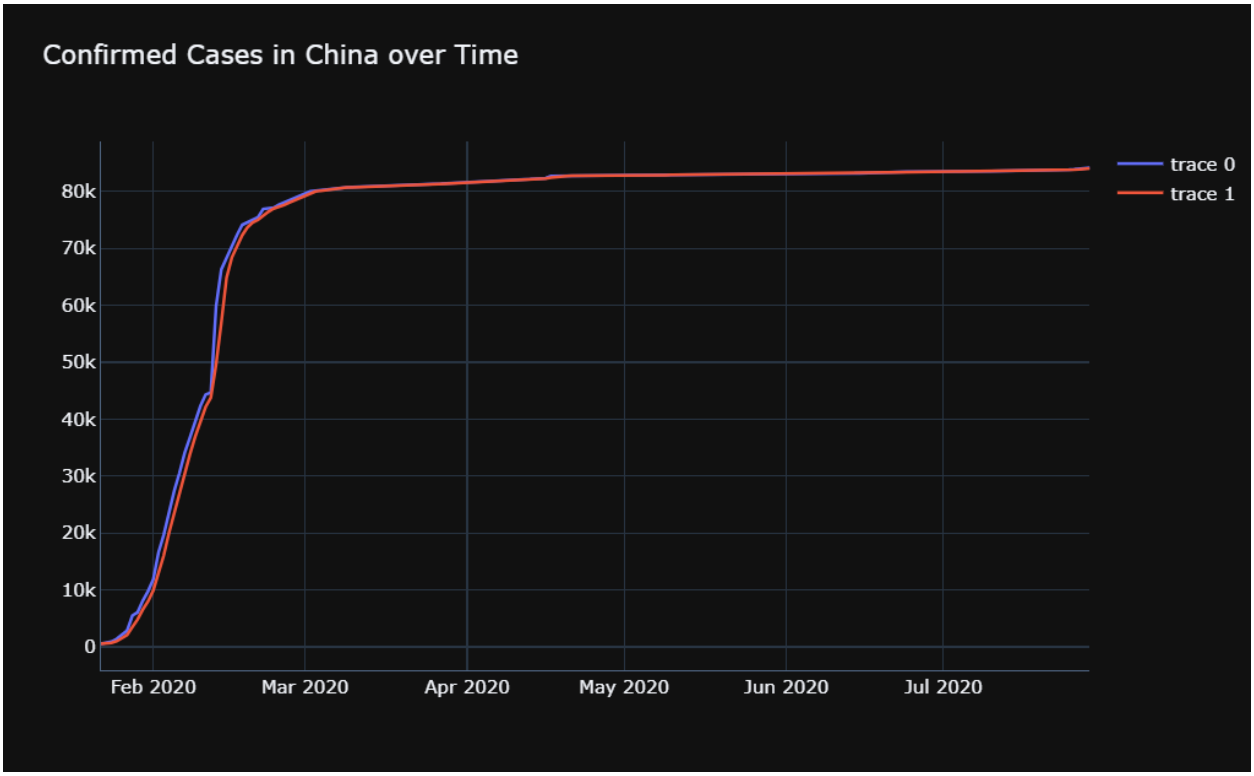

Figure 12: Confirmed Cases in China over time

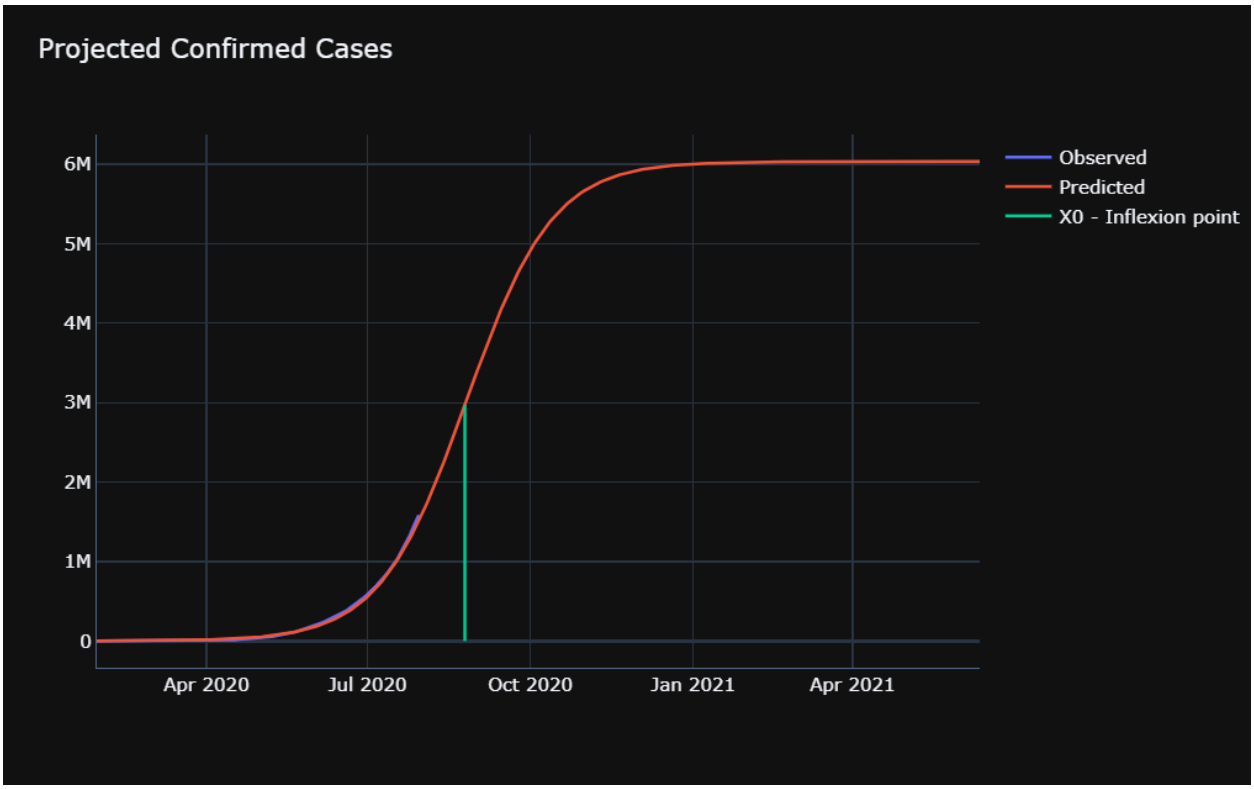

Figure 13: Projected Confirmed Cases in Time

So from the above graph we can see that the cases in India are following a sigmoidal pattern of increasing and have just started to increase and are yet to reach the inflection point. If the cases keep on increasing and the situation remains same and the testing keeps on increasing as it is increasing currently, the covid 19 pandemic is going to stay here for a long time.

So what our data analytics model predicts is that the Covid-19 pandemic will reach its inflexion point on 25-Aug-2020 when the cases will reach 2.96 Million cases and will keep on increasing till 21 Jan 2021 when the cases will reach the value of 6.02 Million.

The cases will keep on increasing even after that but since the Graph would have became a straight line it won't be that hard to keep the things under control.

So, if the testing rates are increased at a very high rate the graph will become steeper and the number of daily cases recorded will become high but will also result in an increased efficiency since the cases will come in control at a much faster rate.

Conclusion (for analyzing tool): So, this analyzing tool is a deployable model which can provide real time analysis of data and would prove quite useful in research as well as Containment of the Pandemic.

4) Methodology and Implementation

4.1) Pre-Processing:

Our data comprises of data in different data-types and usually categorical data as well as missing data, Both this can lead to an inefficient model and can prove to be over-fitted model so we have to encode and fill the missing values. So the three steps in preprocessing are:

4.1.1) Filling Missing Values:

So in some data the Death column is mostly unfilled so what we can do here is if the patient has recovered it means the patient hasn't died.

Our data contains the data in binary form as 0 or 1 so if a patient's recovery state is 1 then we fill the death as 0.

4.1.2) Label Encoding:

To have a model that performs efficiently it is advised to have all the data type in the same format, since we have lots of categorical data like symptoms ranging from 1 to 6 as well as the gender with their respective values.

We use the sklearn library's preprocessing model's LabelEncoder( ) function to Label Encode our categorical Data.

4.1.3) Converting date to date-time format:

Since the date in dd/mm/yy is in object format, we've to convert the data to integer data-type. For this we use the datetime library of python which is used as pd.to_datetime( ) function.

4.2) Algorithms Used:

To find the best model with the best accuracy we have to check lots of Algorithms to find the right fit for our data.

4.2.1) Logistic Regression:

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1".

This linear relationship can be written in the following mathematical form (where ℓ is the log-odds, b is the base of the logarithm, and **β** are parameters of the model):

$$\ell = \log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

We can recover the odds by exponentiating the log-odds:

$$\frac{p}{1-p} = b^{\beta_0+\beta_1 x_1+\beta_2 x_2}.$$

By simple algebraic manipulation, the probability that $Y = 1$ is

$$p = \frac{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}+1} = \frac{1}{1+b^{-(\beta_0+\beta_1 x_1+\beta_2 x_2)}}.$$

4.2.2) Decision Tree :

Decision trees are a classifier model in which each node of the tree represents a test on the attribute of the data set, and its children represent the outcomes. The leaf nodes represents the final classes of the data points. It is a supervised classifier model which uses data with known labels to form the decision tree and then the model is applied on the test data. For each node in the tree the best test condition or decision has to be taken. We use the GINI factor to decide the best split.

$$\text{Variance} = \frac{\Sigma(X-\overline{X})^2}{n}$$

4.2.3) Support Vector Machines:

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

$$\text{maximize } f(c_1 \dots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i (x_i \cdot x_j) y_j c_j,$$

$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i.$$

4.2.4) Naive Bayes:

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\ &= p(x_1 \mid x_2, \dots, x_n, C_k)\, p(x_2, \dots, x_n, C_k) \\ &= p(x_1 \mid x_2, \dots, x_n, C_k)\, p(x_2 \mid x_3, \dots, x_n, C_k)\, p(x_3, \dots, x_n, C_k) \\ &= \cdots \\ &= p(x_1 \mid x_2, \dots, x_n, C_k)\, p(x_2 \mid x_3, \dots, x_n, C_k) \cdots p(x_{n-1} \mid x_n, C_k)\, p(x_n \mid C_k)\, p(C_k) \end{aligned}$$

4.2.5) Boosted Random Forest:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees habit of overfitting to their training set.

$$\hat{y} = \frac{1}{m} \sum_{j=1}^{m} \sum_{i=1}^{n} W_j(x_i, x')\, y_i = \sum_{i=1}^{n} \left(\frac{1}{m} \sum_{j=1}^{m} W_j(x_i, x') \right) y_i.$$

4.3) Accuracy (Multiple Models):

To know which model performed the best we use the metrics library of python and use the recall, precision, f1, balanced accuracy and confusion matrix on our models.

On performing Accuracy tests we come to know that:

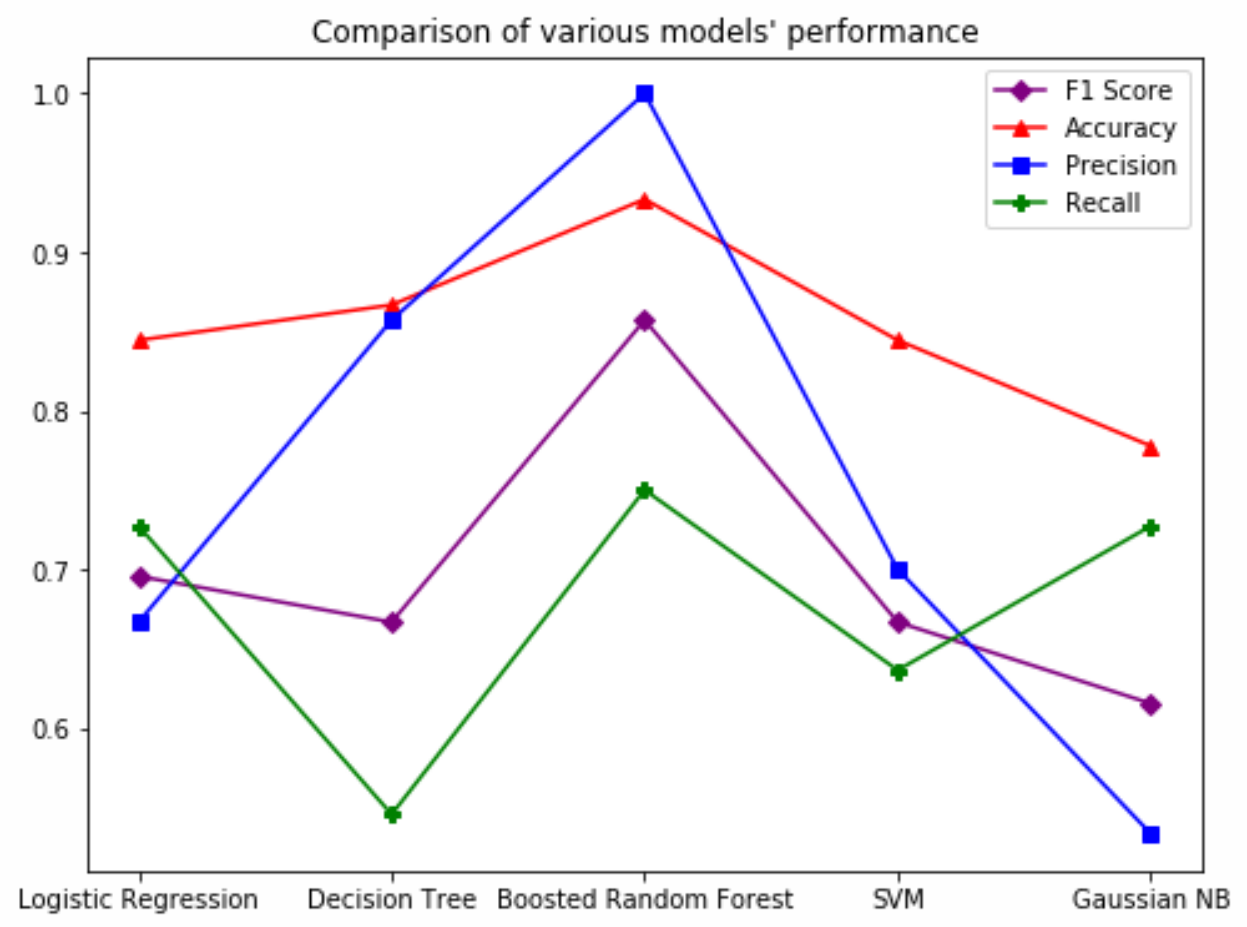

Figure 14: Comparison of models performance

4.4) Feature Selection:

To perform feature selection on the data we use library GridSearchCV in python. We divide the dataset into 80% data as training dataset and the rest 20% as the testing dataset. The model trains using the given dataset and while using the 4 CPU Cores the PC is having and predicts whether a patient will recover or not based on the data in the testing data and get the following parameters for performing the machine Learning Algorithm using the GridSearchCV function by dividing the training dataset into 5 parts, it performs training in 5 sets while considering a different sets as the training set and one among them as validation datset. The parameters obtained are:

```
GridSearchCV(cv=5, error_score=nan,
             estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                              class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              max_samples=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators=100, n_jobs=None,
                                              oob_score=False,
                                              random_state=None, verbose=0,
                                              warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'max_depth': [1, 2, 5, 6],
                         'min_samples_leaf': [2, 3, 4, 5],
                         'min_samples_split': [1, 2, 6, 7],
                         'n_estimators': [100, 200, 300, 400, 500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=5)
```

4.5) Prediction:

Using the features extracted plus the dataset we are using we predict whether a particular patient will recover from the Covid-19 or not. The model predicts whether a patient will recover or not with an accuracy of

```
Recall Score:  0.75
Precision Score:  1.0
F1 Score:  0.8571428571428571
Accuracy:  0.9333333333333333
```

Example of predictions are:

```
0      1
1      0
2      0
3      0
4      0
      ..
858    1
859    0
860    0
861    0
862    0
Name: result, Length: 863, dtype: int64
```