

A Mini Project with Seminar On
Analysis and prediction of Covid-19 Cases using ML
Submitted in partial fulfillment of the requirements for the award of the
Bachelor of Technology

In
Department of Computer Science and Engineering

By

D.Mallesh	18241A05D0
G.Sree Sai Raghavendar	18241A05D7
Dadu Sharief	18241A05G8

Under the Esteemed guidance of
Dr. S.Govinda Rao ,
Professor



Department of Computer Science and Engineering
GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND
TECHNOLOGY
(Autonomous)
Bachupally,Kukatpally, Hyderabad-500090



**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND
TECHNOLOGY**

(Autonomous)

Bachupally,Kukatpally, Hyderabad-500090

CERTIFICATE

This is to certify that the major project entitled “**Analysis and prediction of Covid Cases Using ML**” is submitted by **Dadu Sharief Shaik(18241A05G8)**, **G.Sree Sai Raghavendar (18241A05D7)**, **D.mallesha(18241A05D0)**, in partial fulfillment of the award of degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during academic year **2020-2021**.

INTERNAL GUIDE

Dr. S.Govinda Rao

Professor

HEAD OF THE DEPARTMENT

Dr. K. MADHAVI

EXTERNAL EXAMINER

ACKNOWLEDGE

There are many people helped us ,directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First we would like to express our Deep gratitude towards our internal guide **Dr. S. Govinda Rao , Prof.** Department of CSE for his support in the completion of our dissertation. We wish to express our sincere thanks to **Dr. K. Madhavi, HOD, Department of CSE** and our principal **Dr. J. Praveen** for providing the facilities to complete the dissertation. We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

Dadu Sharief Shaik(18241A05G8)
G.Sree Sai Raghavendar (18241A05D7)
D.Mallesha (18241A05D0)

DECLARATION

We hereby declare that the industrial minor project entitled “**Analysis and Prediction of Covid Cases using ML**” is the work done during the period from **8th March 2021 to 8th July 2021** and is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technological University, Hyderabad).

The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

Dadu Sharief Shaik(18241A05G8)

G.Sree SaiRaghavendar(18241A05D7)

D.Mallesha (18241A05D0)

ABSTRACT

The main aim is to predict and analyze Covid-19 cases of next 30 days using Prophet Library and to bring awareness to the public from the predicted data .

Nowadays, we all are listening about only one thing “Covid -19” and recently one new word ‘Covirgin’ is added which means the person who has not been effected by the pandemic. It shows how much covid-19 has effected us.

This project is used to analyze the situation of covid-19 in every state in India. It Visualizes the value of attributes such as confirmed cases, recovered cases, Deaths in various plots. It is crucial to understand the impact of the pandemic to take significant decisions to decrease the death rate and increase the recovery rate. There are several predictons models in ML. One of them is the prophet model, which is used to predict the situation of the outbreak by analyzing the past data and to take standard measures for defending the worst situations affecting the citizens of their Geographic areas. This pandemic is forcing governments to introduce some extreme measures to fight against its spread. This project is used to understand how the cases rate will change in future and to bring awareness about next wave of covid-19 ahead to the public. This project helps to alert public about next wave and also helps in visualizing number of confirmed, recovered and deceased cases of states among India by simple user-friendly Interface (Dashboard).

TABLE OF CONTENTS

Contents	Page no.
Title Page	i
Declaration	ii
Certificate by the Supervisor	iii
Acknowledgement	4
Abstract	5
Chapter 1 Introduction	8
1.1 Rationale	8
1.2 Goal	8
1.3 Objective	9
1.4 Methodology	9
1.4.1 Data Collection	9
1.4.2 Data Preprocessing	9
1.4.3 Algorithms	17
1.5 Roles and Responsibilities	20
1.6 Contribution of Project	20
1.6.1 Market Potential	20
1.6.2 Innovativeness	20
1.6.3 Usefulness	21
1.7 Report Organization	21
Chapter 2: Requirement Engineering	22
2.1 Functional Requirements	22
2.2 Non Functional Requirements	22
Chapter 3: System Design	22

3.1 Use-case Diagrams	23
3.2 Activity Diagrams	24
3.3 Sequence Diagram	25
3.4 System architecture	26
Chapter 4: Construction	27
4.1 System Requirements	27
4.1.1 Software Requirements	27
4.1.2 Hardware Requirements	27
4.2 Implementation	27
4.3 Testing	29
4.2.1 Types of tests	29
4.2.2 Test cases and Results	30
4.4 Building Application	31
4.4.1 Screenshots	
Chapter 5: Conclusion	39
5.1 Conclusion	39
5.2 References	42

CHAPTER 1

INTRODUCTION

It is crucial to understand the impact of pandemic to take significant decisions to decrease the death rate and increase recovery rate. This project visualizes information about covid-19 cases among states of India. For this we should collect state-wise dataset (changing day-to-day) extracted automatically from covid19.org API, provided and maintained by Ministry of Health affairs and Welfare of India.

Every dataset contains attributes based on the Information, Our dataset contains information recorded according to time, hence, we must use Time-series analysis to treat our data.

One of the algorithm that may help us is Random forest. In random forest, we should create decision trees from datasets and predict for all trees. The final predicted value is taken from average of all predicted values of trees. It might become complex for this model, because ours is a large dataset and a single error or small change in decision trees could lead to large changes.

We may also use ARIMA model for this time series data, but ARIMA model needs pre-testing and various parameters need to be identified after testing past data, since our dataset is dynamic, we can't change parameters daily.

So, we have used prophet model to forecast the data.

RATIONALE

The analysis of cases depicts the impact of covid-19 on India and predicted data denotes future cases, which create awareness in people. By predicted data visualization, The model is used to understand how the cases rate is changing in future and to bring awareness about next wave of covid-19 ahead to the public. This project helps to alert public about next wave and also helps in

visualizing number of confirmed, recovered and deceased cases of states among India by simple user-friendly Interface.

Only 4% people of our country are fully vaccinated till now. This Analysis helps to make ourselves attentive and understand the spread of Pandemic in our location.

GOAL:

In this study, we have mainly focused on:

- Analyze the present situation of covid-19. Visualize the state wise data everyday.
- Comparison of situation among all states.
- Prediction of cases using ML Library called “fbprophet”.

OBJECTIVE:

Objective of our work is to:

- Predict the confirmed cases, recovered cases and no of deaths.
- Analyse the state wise data using interactive plots and user-friendly Interface(Dashboard) which automatically updates with respect to time.

METHODOLOGY:

Data Collection:

Data collection is gathering sufficient information to build Dashboard. Government of India will release day-to-day update on covid-19 cases. Collect that data from govt resources. The extraction of data is from <https://api.covid19india.org> which updates the data daily.

Data Preprocessing:

We should convert collected data to useful data. First we should perform data cleaning. It is done by truncating unnecessary attributes and renaming the attributes for ease of access.

Data Cleaning:

Remove all duplicate data from dataset which is not useful and remove null values

from it or replace null values using previous data. After data cleaning classify the data among states and create dataframes for every individual state with their state names.

```

Out[1]: Click here to toggle on/off the raw code.

In [2]: import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.graphics.tsaplots as sgt
import statsmodels.tsa.stattools as sts
from statsmodels.tsa.seasonal import seasonal_decompose
import numpy as np
import seaborn as sns
import pylab
import scipy.stats
sns.set()
%matplotlib inline

In [3]: df_o = pd.read_csv('C:/Users/khadar/Desktop/covid_data.csv')
df = df_o.copy()
df = df.drop(['Unnamed: 0'], axis=1)
df.Date = pd.to_datetime(df.Date, dayfirst=True)
df.set_index('Date', inplace=True)
df.head()

Out[3]:
State  Confirmed  Active  Death  Discharged
Date

```

Data Visualization :

It is a graphical representation of data. It is used to study and analyze data easily.

There are different types of plotting.

In our dashboard, we need to select type of analysis. Past analysis will visualize the past data and prediction analysis will compare the past and predicted data.

State-wise Analysis

➤ Scatter Plot:

- It is a type of plot using Cartesian coordinates to display values to a set of data. We can select any state from the top-down menu and change the plot for desired time period by changing points of the slider.
- When we hover the mouse pointer on one particular date then it shows all the values of it as shown in below diagram. (To experience it, we must select ‘compare data on hover’ option present at the top right menu of the plot)



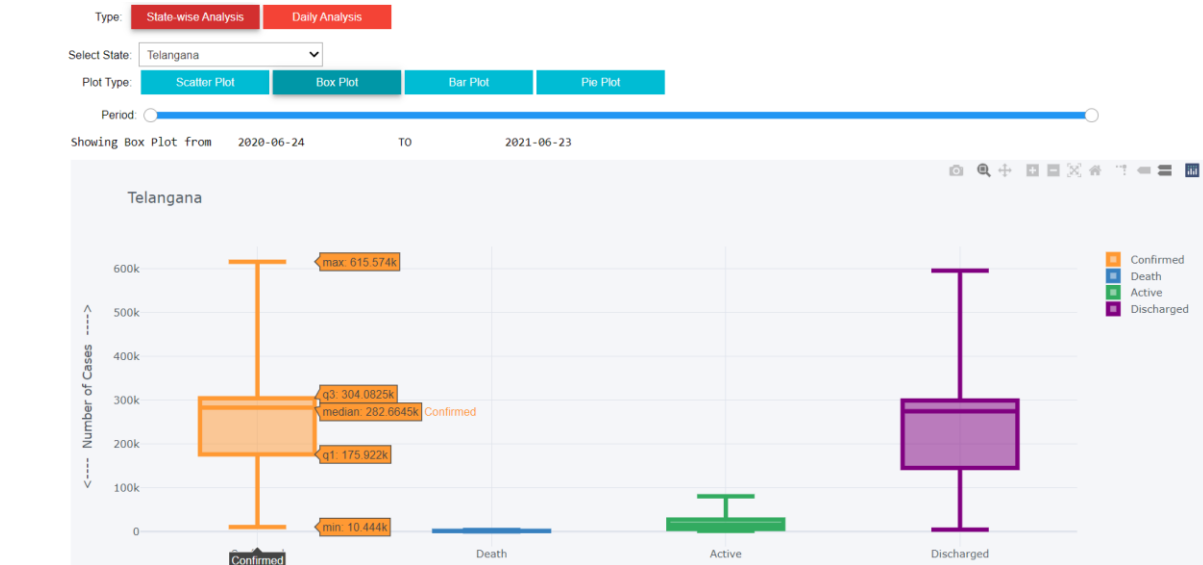
➤ Bar Plot :

A bar plot is also known as bar chart which is represented by rectangular bars and they are proportional to the values which they represent. In this plot confirmed, recovered, active and death cases are represented by different colored bars.



➤ Box plot :

It is one of the graphical representation of numerical values based on quartiles. It is represented as 5 measures. They are min, median, max, 1st quartile(25%) 2nd quartile(50%) and 3rd quartile(75%).



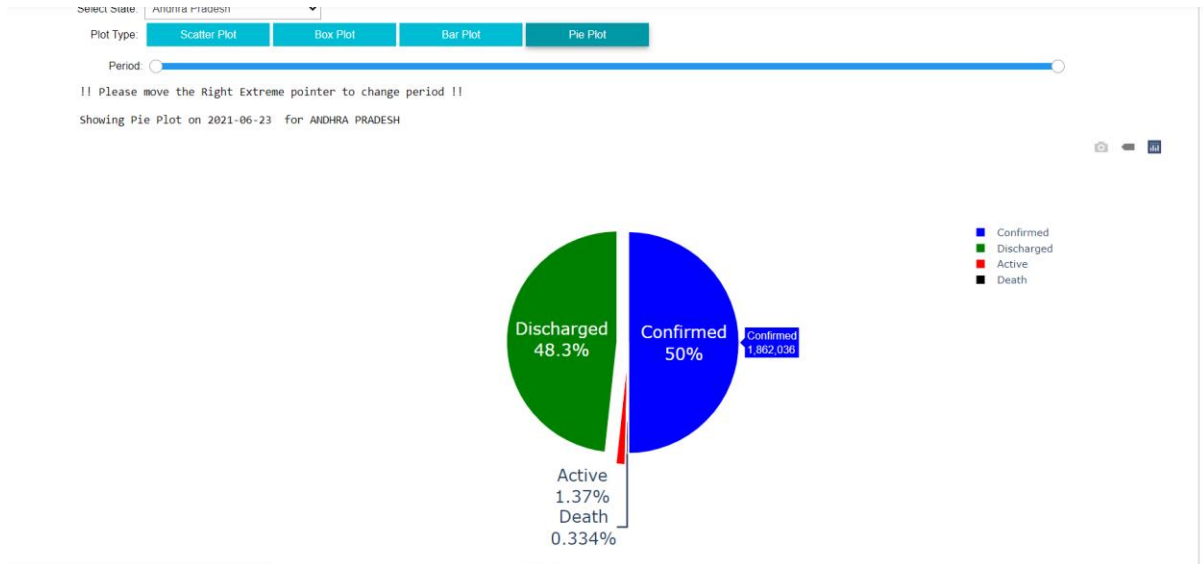
Elements of box plot:

Box plot represented as 5 parts.They are:

- Q0 or minimum :
 - It gives the minimum value from dataset.
- Q1 :
 - It is also known as first quartile or lower quartile.It is median value for lower half of dataset.
- Q2:
 - It gives the median (middle value) of dataset.
- Q3:
 - It is also known as 3rd quartile or higher quartile.It gives median value of higher half of the dataset.
- Q4:
 - It gives the maximum value from dataset.

➤ Pie Chart:

Pie chart is a pictorial representation of Data. It represents the data in percentage.



Conclusion from above chart, In AP 48.3 in 50 of covid-19 affected are recovered and current active cases are 1.37%.

Daily Analysis:

In Daily analysis, we have plotted the bar graph and maps for required date (could be selected using calendar). We need to select before Today's date. If we selected wrong date then it shows warning message in dashboard as shown in the below diagram.

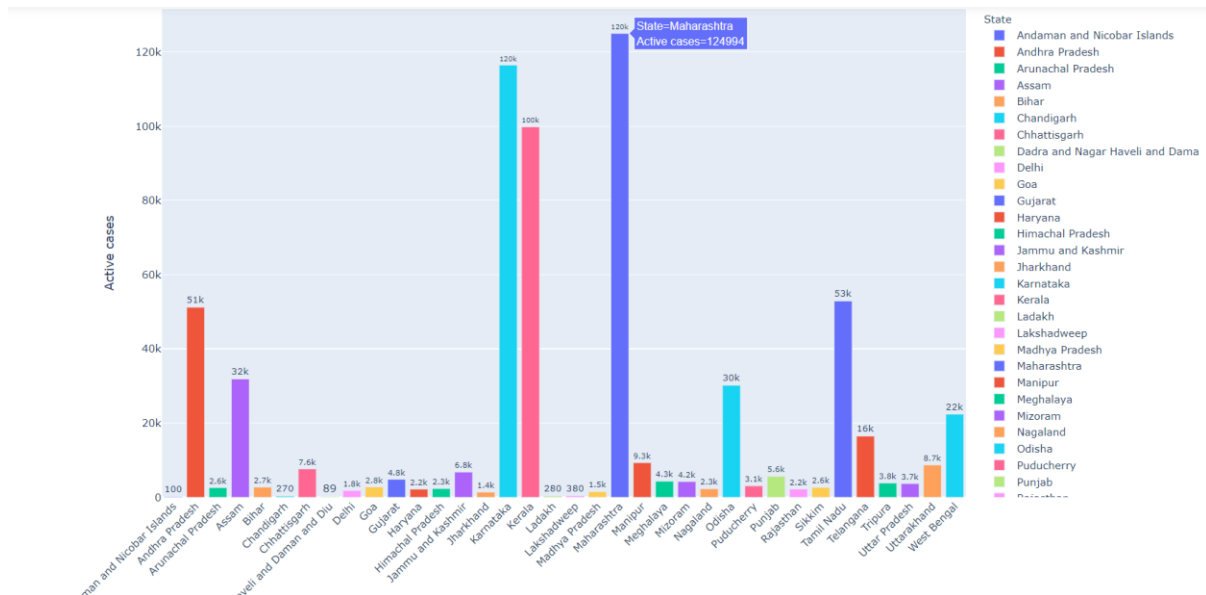


Bar View:

In this view the data of selected date will be represented in bar graph for all states .

We need to select the type of cases to represent .

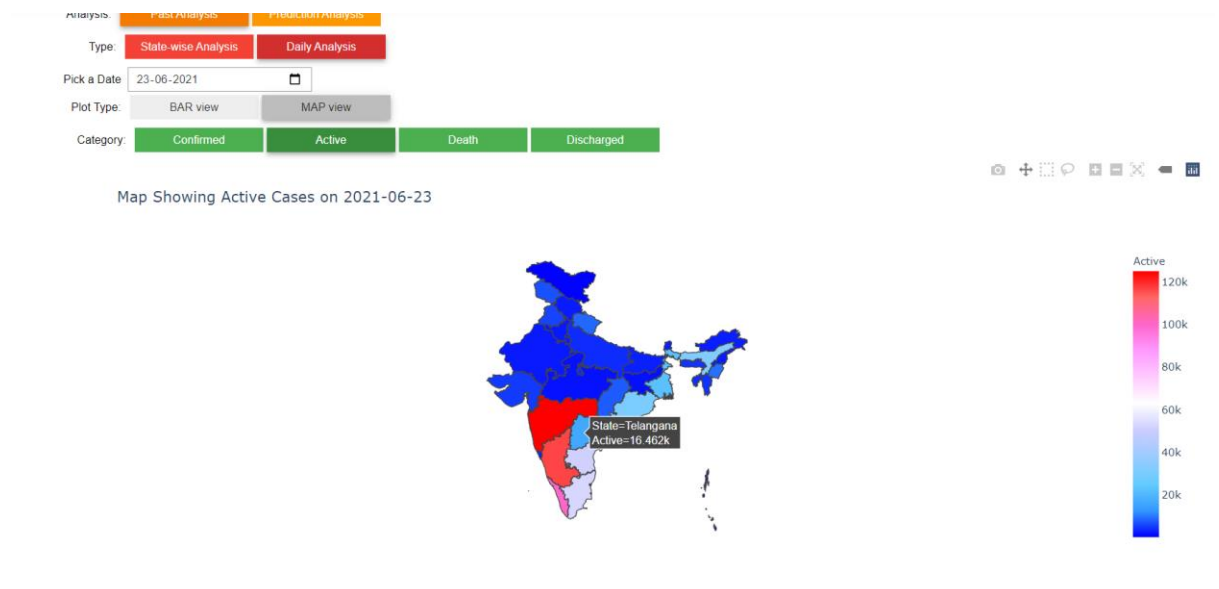
Below diagram shows active cases in india on 23 rd june of 2021 on bar graph.



Map View

All types of cases like confirmed cases, Active cases, Deaths, Recovered cases of selected date in calendar are shown in India map (Choropleth plot) .

○ Active cases



Confirmed Cases

Type: **State-wise Analysis** **Daily Analysis**

Pick a Date: 23-06-2021 ☐

Plot Type: **BAR view** **MAP view**

Category: **Confirmed** **Active** **Death** **Discharged**

Map Showing Confirmed Cases on 2021-06-23



○ Death cases

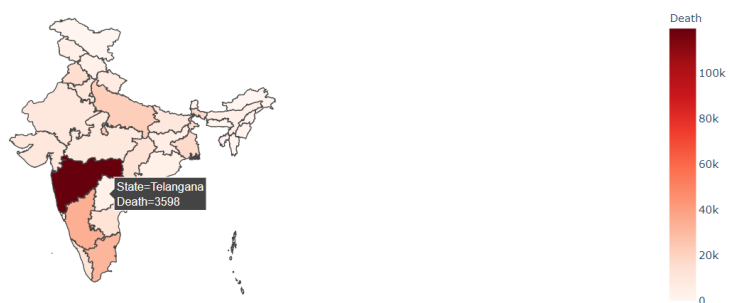
Type: **State-wise Analysis** **Daily Analysis**

Pick a Date: 23-06-2021 ☐

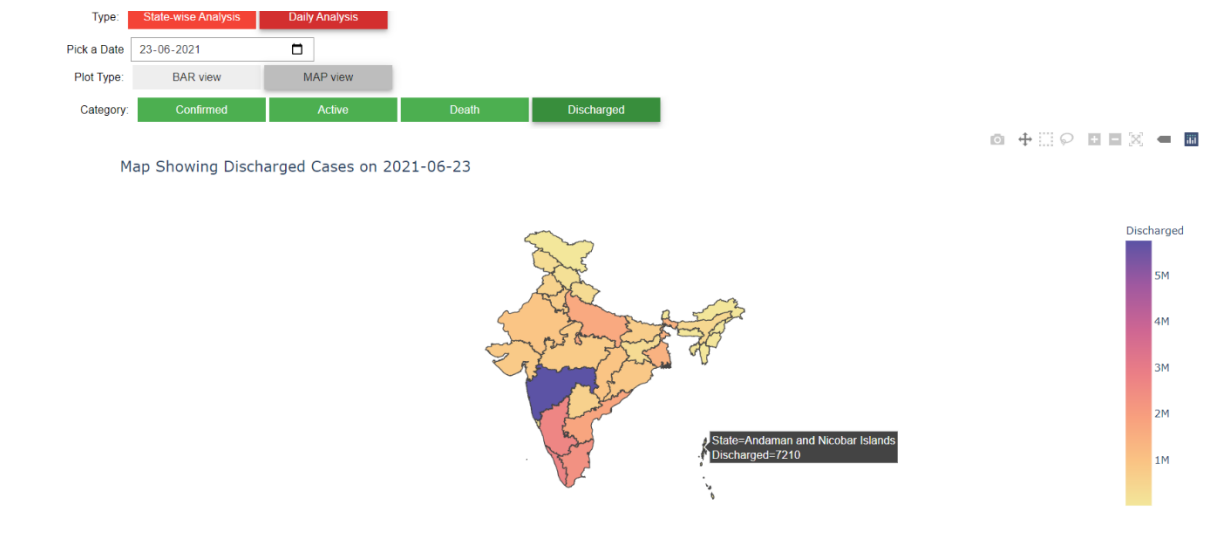
Plot Type: **BAR view** **MAP view**

Category: **Confirmed** **Active** **Death** **Discharged**

Map Showing Death Cases on 2021-06-23



○ Recovered Cases



Prediction Analysis



Above diagram shows the Predicted cases on JULY 15 2021 of Telangana State.

Algorithms:

For implementation, we should import some libraries and they are:

- **Pandas :**

It is a software library written in python for data manipulation and Analysis. Pandas are suited for different kinds of data. Some of them are ordered and unordered time series data.

- **Plotly:**

Library to plot interactive plots. Some libraries should install from plotly. They are:

- Plotly.express : for plotting interactive graphs.
- Plotly.graph_objects
- Chart_studio : to enable offline plotting of plotly graphs.

- **Json :**

- It is used to read json files and converting them into dictionary.

- **Numpy**

- **Cufflinks**

- **Urlli.request**

- **Urlopen**

It is method used to extract data by passing URL addresses.

- **Matplotlib:**

It is a python package used for 2D graphics.

- **Ipywidgets:**

To create html widgets with in Jupyter. There are some libraries should from Ipywidget. They are

- Interact
- IntSlider
- IntRangeSlider
- widgets
- interact_manual
- HBox
- fixed.

For prediction , we have used fbprophet library, which is best in handling Time-series data .

Advantages of Prophet:

- It is available in Python.
- Accurate
- Fast
- Tunable Forecasting
- Fully Automated

Prophet is a open source software and it is released by Facebook data science team. It is used for forecasting time series where we are update the data daily or weekly or yearly.

It is robust to outlier, missing data and dramatic changes in time series analysis.

Dropdown

Select State: Telangana

Plot Type:

Period:

Showing Sc

Te

600k

500k

^

s

Andaman and Nicobar Islands

Andhra Pradesh

Arunachal Pradesh

Assam

Bihar

Chandigarh

Chhattisgarh

Dadra and Nagar Haveli and Daman and Diu

Delhi

Goa

Gujarat

Haryana

Himachal Pradesh

Jammu and Kashmir

Jharkhand

Karnataka

Kerala

Ladakh

Lakshadweep

Madhya Pradesh

ToggleButton

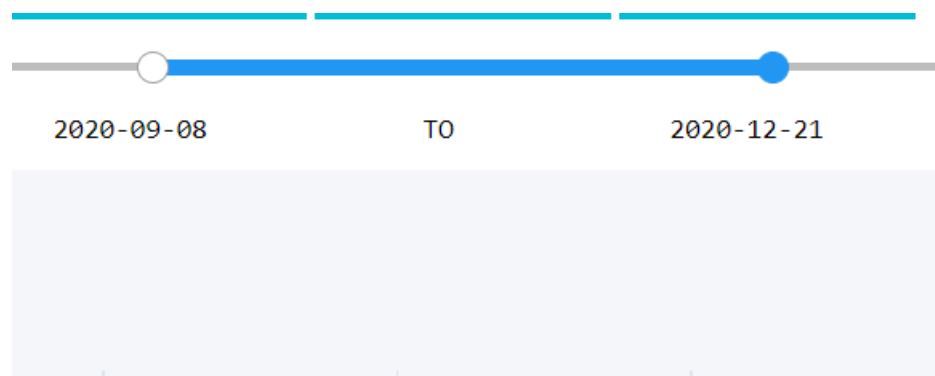
Analysis: Past Analysis Prediction Analysis

Type: State-wise Analysis Daily Analysis

IntRangeSlider:



Hbox:



Some features present in plots are:

- Zoom out
- Zoom in
- Autoscale
- Reset axes
- Download as png
- Compare data on hover

Roles and Responsibilities:

Role	Name	Responsibilities
Data Gathering & Analysis	D. Mallesh	<ul style="list-style-type: none">• Data entry• Data cleaning• UML diagrams• Documentation
Core implementation & UI Design	Dadu Sharief	<ul style="list-style-type: none">• Data preprocessing• Data analysis• White Box Testing• Dashboard design• Data Forecasting• Documentation• ML Concepts
Data Forecasting	G.Sree Sai Raghavendar	<ul style="list-style-type: none">• Data prediction• Testing• ML concepts• Documentation• Black Box Testing

Contribution of project

Market Potential:

Day -by-day covid cases are increasing unimaginably, it is necessary to understand the condition of covid in every state. In this project we are creating a dashboard which used to analyze the data. There are several options in it. We can select a required state in dashboard and the situation of that state can be seen in different time periods with different kind of visualizations. We can also select type of plots and compare attributes using the legend of the plotted graph.

Innovativeness:

In this project, we have created a Dashboard as an Interface to access every plot defined and every feature created in the project with interactive design. This makes comfortable for any novice to make use of this project without any technical background.

Usefulness:

This pandemic is forcing governments to introduce some extreme measures to fight against its spread. This project is used to understand how the cases rate will change in future and to bring awareness about next wave of covid-19 ahead to the public. This project helps to alert public about next wave and also helps in visualizing number of confirmed, recovered and deceased cases of states among India by simple user-friendly Interface(Dashboard).

This project is designed for people to understand the situation of covid-19 in India across all states including Union territories.

Report Organization

The remaining section of the report is structured as follows:

- **Chapter 2** provides detailed technical and non-technical requirements.
- **Chapter 3** provides analysis and design of this project
- **Chapter 4** provides Construction, implementation details of this project.
- **Chapter 5** provides Conclusion and references.

CHAPTER 2

REQUIREMENT ENGINEERING

Functional Requirement:

It describes the core functionality of the application.

Interface Requirement

- Field 1 – Datasets in CSV format(available online, URL is required,need not to download).
- Field 2- Download Python latest version.
- Field 3- Install all libraries specified in chapter 1.

- Filed5- Dataframes containing predicted data..
- Webpage1- visualize the data and plotting its instance.

Non Functional Requirement

Non functional requirements are those requirements of the system which are not directly concerned with specific functionality delivered by the system.

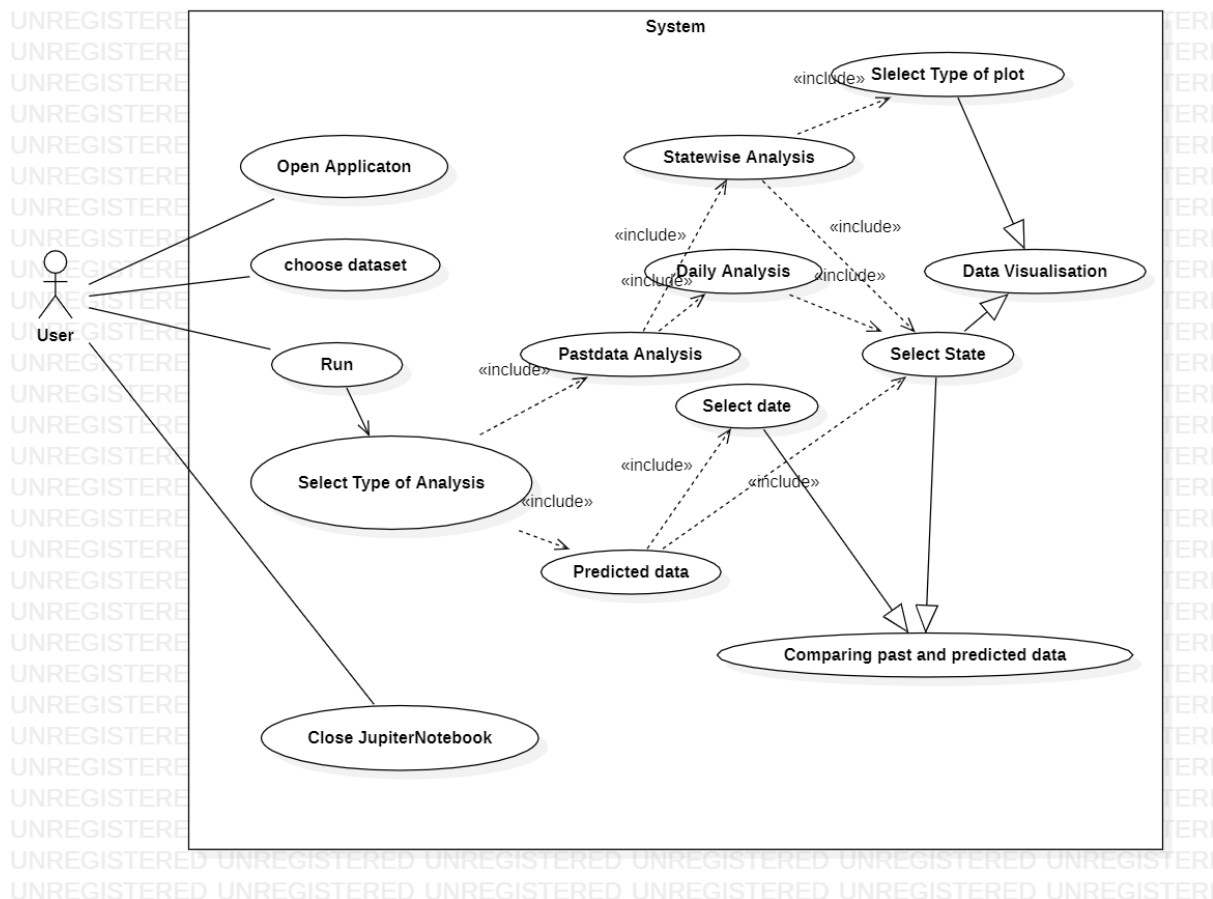
- To provide max accuracy
- Ease of use
- Availability
- Reliability
- Provide Visualized analysis

CHAPTER -3

ANALYSIS AND DESIGN

USE CASE DIAGRAM:

A Use Case Diagram is a graphical representation of user interaction with system. It shows about different types of users interact with the system. ACTORS represents the users in system.



Class Diagram:

It is a static view of application. It describes about the attributes and operations performed by attributes in the class.

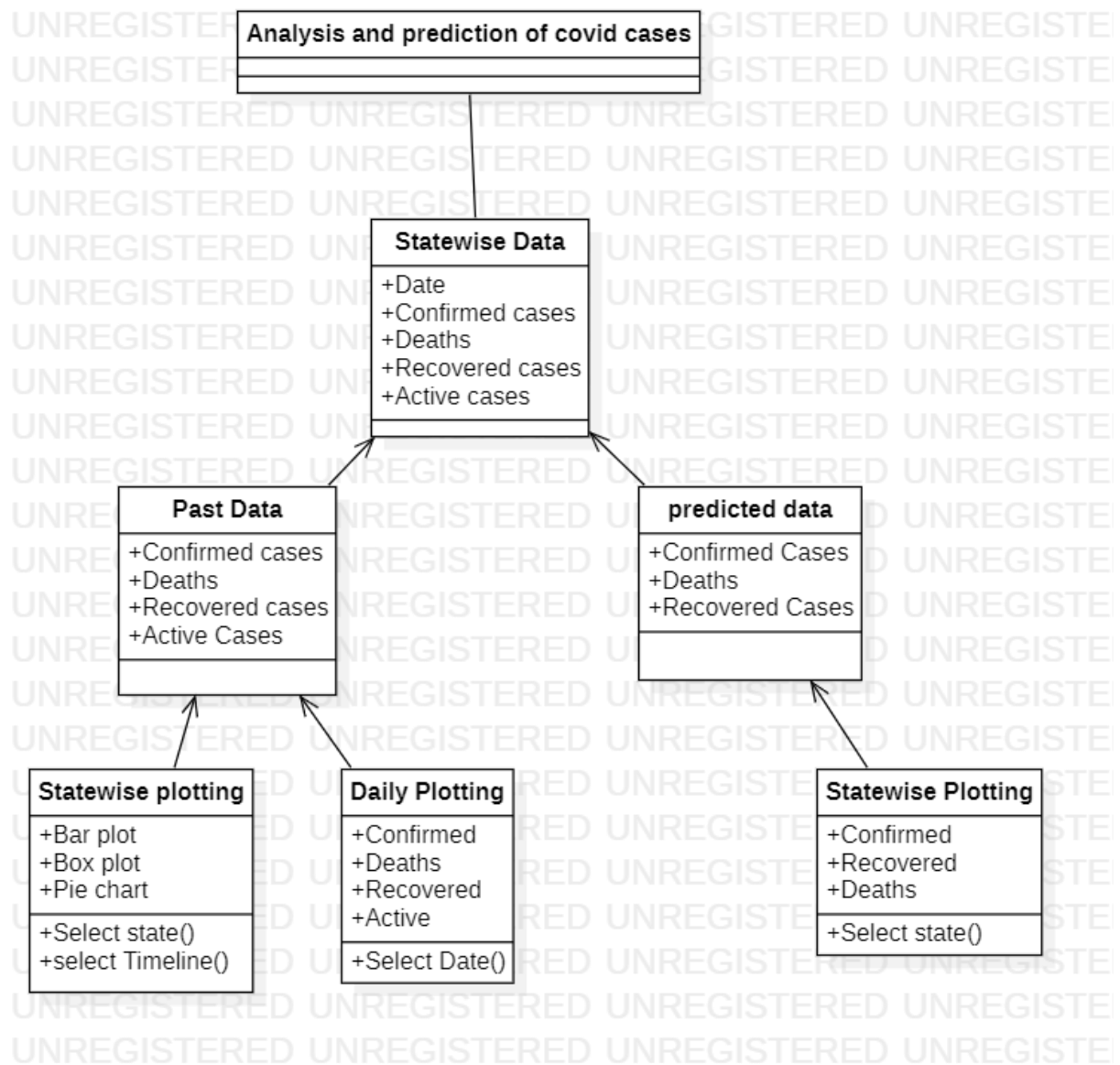
In this class diagram, there are class like past analysis, prediction analysis, statewise plots, Daily plots. The attributes from classes past analysis and predicted analysis are taken from dataset which are confirmed cases, Deaths, Recovered cases.

Operations performed in past analysis are:

- Select Timeperiod(): select the range of period in timeline to analysis the data for selected period.
- Select Type of cases: Select one type from confirmed, Deaths, Active and recovered cases.
- Select State()

Operations performed in prediction analysis are:

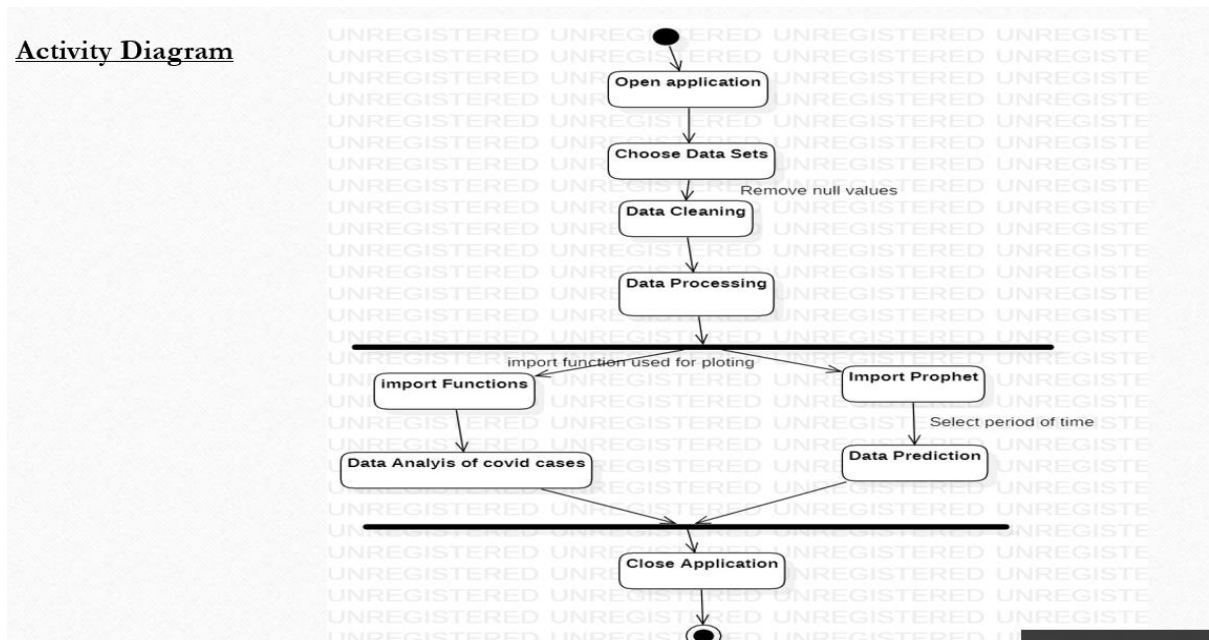
- Select Date()
- Select State()
- SelectView()



Acitivity Diagram:

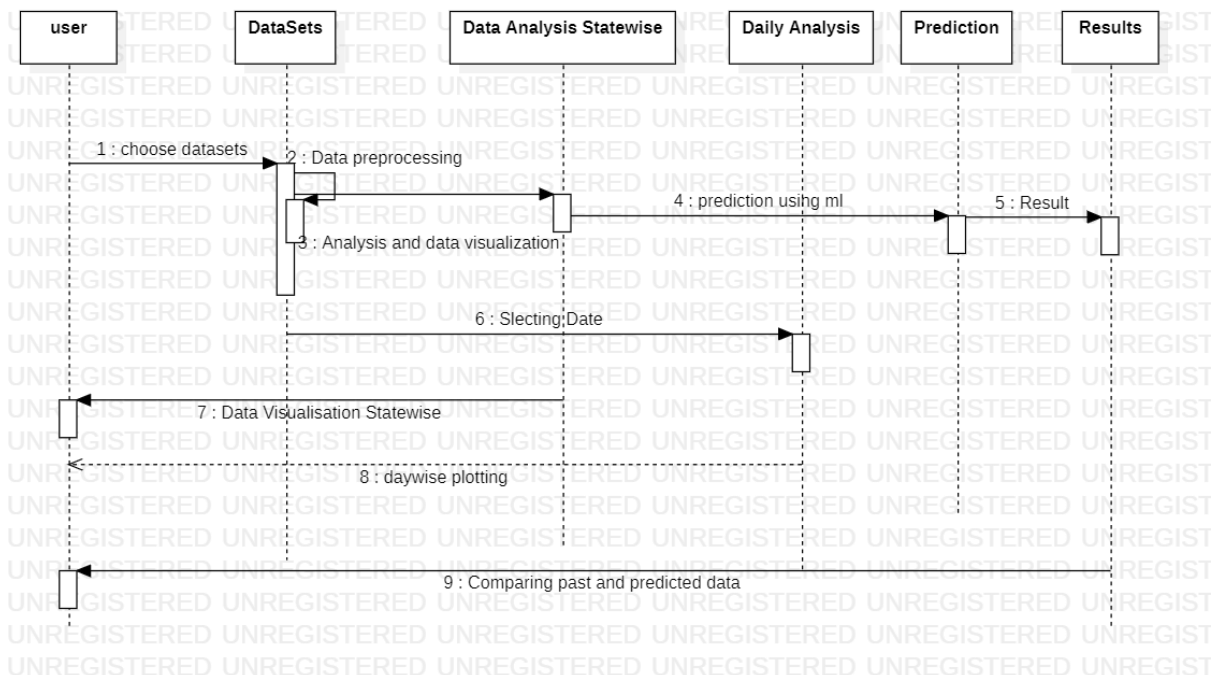
It describes about the dynamic view of the system. It represents active flow of system and describes sequences of one activity to another activity.

Activity Diagram

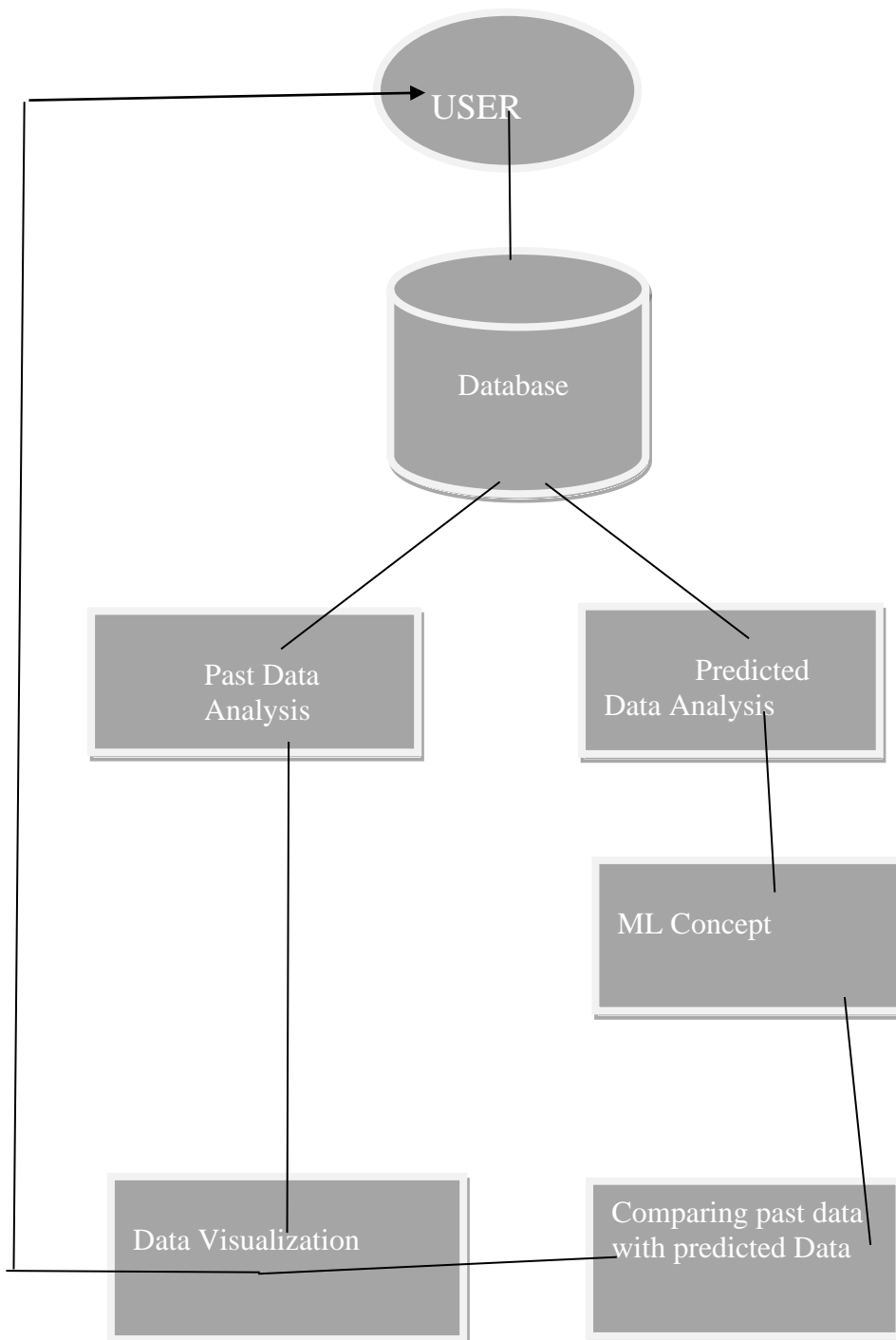


Sequence Diagram:

It is an interactive diagram ,it shows step by step order of how objects works together



System architecture



CHAPTER-4

Construction

System Requirements:

Software Requirements:

- Anaconda3-64 bit
 - Anaconda Navigator
 - Jupyter Notebook
- Python version- above 3.7.x

Hardware Requirements:

- Operating Systems : Windows 7 or above
- Ram :4GB or Greater
- System architecture: 64-bit x86, 32-bit x86 with Windows / Linux

Implementation:

The implementation of project is done with python. Anaconda is one of several Python distributions. It is a new distribution of Python. It has more than 100 new packages. Anaconda is used for scientific computing, statistical analysis, machine learning etc.

Anaconda helps to getting up and running the specific packages and it also used for separating from different environments.

MODULE1:


In this module ,we are analyzing data exported from Github Repository using URL and appropriate python libraries. Visualization of the data is done in this module.We should install all required libraries from anaconda navigator in this model.Some of them are:

Pandas, Numpy, matplotlib.Pyplot, Plotly, ipywidgets, fbprophet etc.

Steps to do in this module:

- Import all required libraries:

- Panadas: Commands to install pandas: **pip install pandas**



```
Microsoft Windows [Version 10.0.19042.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sai Raghavendar>pip install pandas
Collecting pandas
  Downloading pandas-1.2.5-cp38-cp38-win32.whl (8.2 MB)
    |#####| 8.2 MB 1.6 MB/s
Requirement already satisfied: numpy>=1.16.5 in c:\users\sai raghavendar\appdata\local\programs\python\python38-32\lib\site-packages (from pandas) (1.20.3)
Collecting python-dateutil>=2.7.3
  Using cached python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
Collecting pytz>=2017.3
  Using cached pytz-2021.1-py2.py3-none-any.whl (510 kB)
Collecting six>=1.5
  Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, pytz, python-dateutil, pandas
Successfully installed pandas-1.2.5 python-dateutil-2.8.1 pytz-2021.1 six-1.16.0

C:\Users\Sai Raghavendar>
```

- Plotly: **pip install plotly**
- Cufflinks :**pip install cufflinks**
- Json : **pip install jsonlib**
- Numpy : **pip install numpy**
- Matplotlib : **pip install matplotlib**
- Urlopen : **pip install urlopen**
- Ipywidgets : **pip install ipywidgets** or **conda install -c conda-forge ipywidgets**
- Fbprophet:
 - **conda install -c conda-forge fbprophet**
 - **conda install -c conda-forge/label/cf201901 fbprophet**
 - **conda install -c conda-forge/label/cf202003 fbprophet**

- Extracting of Data
- Data Preprocessing
- Past-data Analysis
 - State-wise Analysis

1) Bar Plot

- 2) Box Plot
- 3) Scatter plot
- 4) Pie plot
- Daily Analysis
 - 1) Bar view
 - 2) Map View

MODULE 2:

In this module, We are forecasting data by applying Prophet model using past data with the help of fbprophet library. In Prophet we should create a dataframe with 2 columns which are “ds” and “y”. DS stands for date and Y gives the predicted values in form of range (yhat_lower ,y_upper) and also with attributes monthly daily and seasonal trends. Create a base model with prediction interval and passing daily_seasonality and yearly_seasonality parameters. Invoke make_future_dataframe method to predict the data for next “period”(in our case next 30 days) of time. After prediction, store the dataframe according to the states and use that dataframes for further visualization of predicted data in Dashboard utility functions.

- Import Prophet
- Prediction Analysis
 - Compare predicted dataset with past data

Testing:

Software Testing is used to detect errors in software, it identifies bugs and resolves before completion of the project. Testing is crucial to cross validate the working of all modules as defined.

There are different types of Testing. Some of them are:

- WhiteBox Testing
- BlackBox Testing

White Box Testing:

It mainly focuses on inner structure and operation of the application. The design and software are tested to control any errors in input-outputs .

Black Box Testing:

Black Box testing checks for errors from end-user perspective . This testing focuses on information domain of the project.

TEST CASES

Action	Expected Output	Output
1.Open the application	Application will open.	Application is opened.
2.Select Dataset	Dataset will be Selected.	Dataset Selected.
3.Data Cleaning	Remove null values from dataset.	Null values are removed.
4.Data Analysis	Visualize data from dataset.	Visualized the data.
5.Data Prediction	Build prophet model and	Future data is predicted.
	Predict the future data.	
6.Plot Predicted data	Visualize data from predicted dataset.	Visualized the predicted data.

Implementation

Module 1

```
import ipywidgets

from ipywidgets import interact, IntSlider, interactive, widgets, interact_manual, HBox, fixed
from IPython.display import HTML

import warnings

warnings.filterwarnings('ignore')

layout = widgets.Layout(width='400px', height='30px')

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

%matplotlib inline

from plotly.offline import iplot

import chart_studio.plotly as py

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)

import cufflinks as cf
cf.go_offline()

import plotly.express as px

import plotly.graph_objects as go

import json

from urllib.request import urlopen

from fbprophet import Prophet
```

#Extracting and Preprocessing Dataset obtained from Covid-19 API

```
df = pd.read_csv('https://api.covid19india.org/csv/latest/states.csv')
df = df[df.State != 'India']
df = df[df.State != 'State Unassigned']
df.drop(['Other', 'Tested'], axis=1, inplace=True)
```

```

df.rename(columns={'Recovered': 'Discharged', 'Deceased': 'Death'}, inplace = True)
df['Active'] = df.apply(lambda row: row.Confirmed - (row.Discharged + row.Death), axis = 1)
df_o = df.copy()
df.Date = pd.to_datetime(df.Date)
df.set_index("Date", inplace = True)
df = df[['State', 'Confirmed', 'Death', 'Active', 'Discharged']]

```

#Classifying The DataFrame to distribute the data among states of India

```

states = df.groupby('State')
state={}
l=[]
for i,j in states:
    state[i] = j[-365:]
for i,j in state.items():
    l.append(i)
del states
states_list = l

```

Setting daily frequency State-wise and filling the Data for the dates which are not recorded(missed), with previous date data.

```

for i in state.keys():
    tdf = state[i]
    tdf = tdf[['Confirmed', 'Death', 'Active', 'Discharged']].astype(int)
    tdf = tdf.asfreq('d')
    state[i] = tdf.fillna(method = 'ffill')
min_date = state['Goa'].index.min()
max_date = state['Goa'].index.max()
pdf = df_o.copy()
pdf = pdf[['Date', 'State', 'Confirmed', 'Death', 'Active', 'Discharged']]

```

#Classifying The DataFrame to distribute the data among states of India

```

pstates = pdf.groupby('State')

```



```
pstate={}
for i,j in pstates:
    pstate[i] = j[-365:]
```

#Daily Analysis

#Creating widgets and Dashboard Functions

#Extracting Geojson file for states of India

with urlopen('https://raw.githubusercontent.com/sharif9911/Interactive-Covid-19-India--Dashboard-with-in-Jupyter-Notebook/main/states_india.geojson') as response:

```
    counties = json.load(response)
select_date = df.index.max()
select_date = widgets.DatePicker(
    description='Pick a Date',
    disabled=False,
)
kinda_cat=widgets.ToggleButtons(
    options=['Confirmed', 'Active', 'Death', 'Discharged'],
    description='Category:',
    disabled=False,
    button_style='success', # 'success', 'info', 'warning', 'danger' or ''
    # tooltips=['Sc', 'Bar', 'Box'],
    #icons=['check'] * 3
)
```

```
view=widgets.ToggleButtons(
    options=['BAR view', 'MAP view'],
    description='Plot Type:',
    disabled=False,
    button_style="", # 'success', 'info', 'warning', 'danger' or ''
    # tooltips=['Sc', 'Bar', 'Box'],
    #icons=['check'] * 3
)
```

```

def day_bar(select_date,view):
    if(select_date == None):
        select_date= max_date
    if(select_date < df.index.min() or select_date > df.index.max()):
        print('PLEASE PICK A VALID DATE IN THE CALENDAR i.e From ',str(min_date)[:10],' to '
, str(max_date)[:10],'\n ----Refer to the Database for more Information!')
    return
    t_df = pd.DataFrame(columns=['Confirmed', 'Active', 'Death', 'Discharged'])
    for i,j in state.items():
        t_df.loc[i]= j.loc[str(select_date)]
        t_df = t_df[['Confirmed','Death','Active','Discharged']].astype(int)
    theme= {'Discharged': 'sunset', 'Active': 'picnic', 'Death': 'reds', 'Confirmed': 'turbid'}
    def plot_bar(kinda_cat):
        if view == 'BAR view':
            fig = px.bar(t_df,x=t_df.index, y=kinda_cat, text=kinda_cat, color=t_df.index,title=
kinda_cat + ' Cases on '+ str(select_date)[:10],
                labels={'index': 'State', 'Active': str(kinda_cat + ' cases')},width=1400,height=900)

            fig.update_traces(texttemplate='%{text:.2s}', textposition='outside')
            fig.update_layout(uniformtext_minsize=8, xaxis_tickangle=-45)
            fig.show()
        else:
            fig = px.choropleth(t_df, geojson=counties, locations=t_df.index, color=kinda_cat,
                color_continuous_scale=theme[kinda_cat],
                featureidkey='properties.ST_NM',
                labels={'index': 'State'}, title='Map Showing '+kinda_cat+' Cases on
'+str(select_date)[:10]
                )
            fig.update_geos(fitbounds="locations", visible=False, showsubunits=True,
subunitcolor="Green")
            fig.show()

```

```

interact(plot_bar, kinda_cat=kinda_cat)
# interact(day_bar, select_date=select_date, view = view)

```

#State-wise Analysis

```

def stategraph():
    select_state = widgets.Dropdown(
        options=states_list,
        value='Telangana',
        description='Select State:',
        disabled=False,
    )

    kinda_plot=widgets.ToggleButtons(
        options=['Scatter Plot', 'Box Plot', 'Bar Plot','Pie Plot'],
        description='Plot Type:',
        disabled=False,
        button_style='info', # 'success', 'info', 'warning', 'danger' or ''
        # tooltips=['Sc', 'Bar', 'Box'],
        #icons=['check'] * 3
    )

    def printgraph(select_state,kinda_plot):
        date_range = widgets.IntRangeSlider(
            value=[0, state[select_state].count().max()],
            min=0,
            max=state[select_state].count().max()-1,
            step=1,
            disabled=False,
            description='Period:',
            continuous_update=False,
            layout=widgets.Layout(width='90%'),

```

```

orientation='horizontal',
readout=False,
readout_format='d'
)
def plot(date_range):
    if kinda_plot != 'Pie Plot':

print('Showing',kinda_plot,'from','\t',str(state[select_state].index[date_range[0]][:10],
      '\t\t','TO','\t\t',str(state[select_state].index[date_range[1]][:10])
      state[select_state].iloc[date_range[0]:date_range[1]].iplot(kind =
kinda_plot.split()[0].lower(),
      xTitle='<---- Date ---->', yTitle='<---- Number of Cases ---->',
      title = select_state, width=5,dash=['solid','dash','dot','dashdot'],
      )
    else:
        print('!! Please move the Right Extreme pointer to change period !!\n')
        print('Showing',kinda_plot,'on',str(state[select_state].index[
max(date_range[0],date_range[1]) ][:10], ' for '+select_state.upper())
        colors = ['blue', 'black', 'red', 'green']
        fig = go.Figure(data=[go.Pie(labels=['Confirmed','Death','Active','Discharged'],
            values=state[select_state].iloc[ max(date_range[0],date_range[1]) ]))]
        # Define hover info, text size, pull amount for each pie slice, and stroke
        fig.update_traces(hoverinfo='label+value', textfont_size=20,
            textinfo='label+percent', pull=[0, 0, 0.1, 0.1],
            marker=dict(colors=colors, line=dict(color='#FFFFFF', width=2)))
        fig.show()

    interact(plot,date_range=date_range)
    interact(printgraph,select_state=select_state,kinda_plot=kinda_plot)

#interact(stategraph)

```

Module 2

#Preparing Dataframes for forecasting data of next 30 days

```
fut_state={}

for i,j in pstate.items():
    fd = pd.DataFrame(columns=['Date','Confirmed', 'Death', 'Discharged'])
    for k in ['Confirmed','Death','Discharged']:
        ts_d = j[['Date',k]][-50:].copy()
        ts_d.rename(columns={'Date':'ds',k:'y'}, inplace=True)
        m = Prophet(interval_width = 0.95,daily_seasonality = True, yearly_seasonality = False)
        m.fit(ts_d)
        future = m.make_future_dataframe(periods=30)
        forecast = m.predict(future)
        fd['Date']=forecast.ds[-30:]
        fd[k]=forecast.yhat[-30:]
    del m
    j = j.append(fd,ignore_index = True)
    j.Date = pd.to_datetime(j.Date)
    j.set_index("Date", inplace = True)
    j = j[['Confirmed','Death','Discharged']].astype(int)
    j = j[['Confirmed','Death','Discharged']]
    fut_state[i] = j
```

#Prediction Analysis

```
select_state = widgets.Dropdown(
    options=states_list,
    value='Telangana',
    description='Select State:',
    disabled=False,
)
```

```

def plotg(select_state):
    print('Showing Scatter PLOT for the coming 30 Days of ',select_state.upper(),'State','
(predicted using FBPROPHET)')
    fut_state[select_state].iplot(xTitle='<---- Date ---->', yTitle='<---- Number of Cases ---->',
        title = select_state, width=5,dash=['dash','dot','dashdot'])

#interact(plotg,select_state=select_state)

```

#Main Interface of Dashboard

```

select_analysis = widgets.ToggleButtons(
    options=['Past Analysis', 'Prediction Analysis'],
    description='Analysis:',
    disabled=False,
    button_style='warning', # 'success', 'info', 'warning', 'danger' or ''
    #   tooltips=['Sc', 'Bar', 'Box'],
    #icons=['check'] * 3
)

def analy(select_analysis):
    if(select_analysis[1] == 'a'):
        past = widgets.ToggleButtons(
            options=['State-wise Analysis', 'Daily Analysis'],
            description='Type:',
            disabled=False,
            button_style='danger', # 'success', 'info', 'warning', 'danger' or ''
            #   tooltips=['Sc', 'Bar', 'Box'],
            #icons=['check'] * 3
        )

        def p_analy(past):
            if past[0]=='S':
                interact(stategraph)
            else:

```

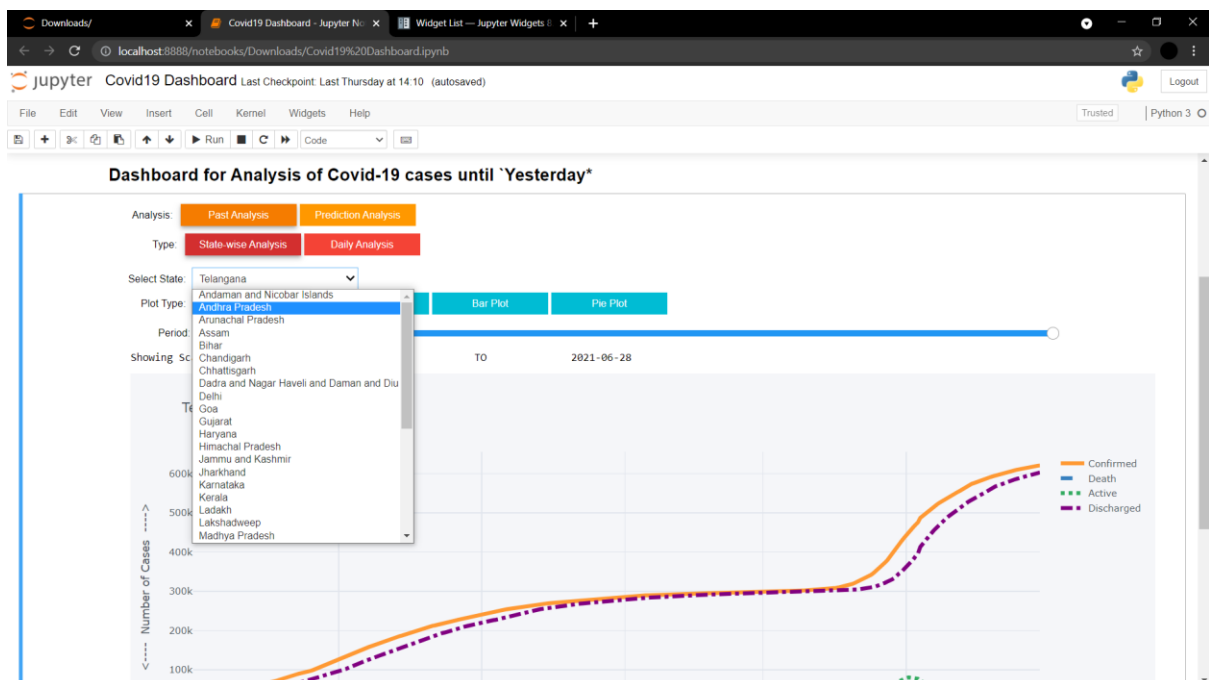
```
interact(day_bar, select_date=select_date, view = view)
```

```
interact(p_analy, past=past)
```

else:

```
interact(plotg, select_state=select_state)
```

```
interact(analy, select_analysis=select_analysis)
```



CHAPTER 5

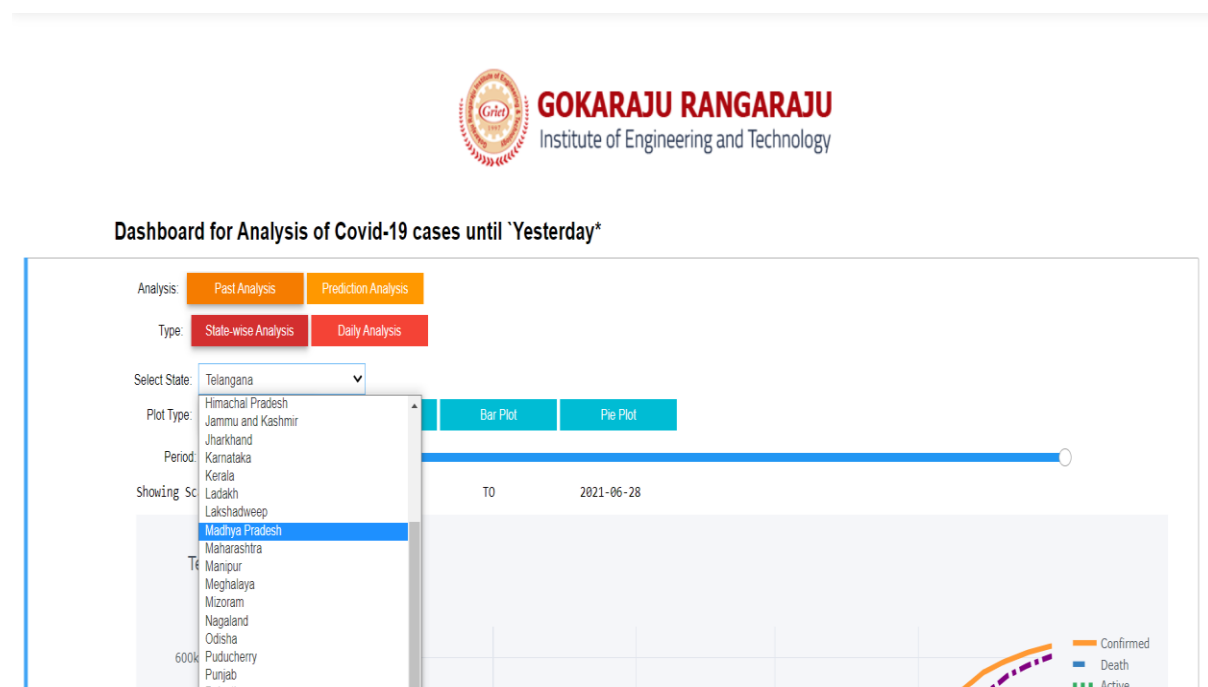
Conclusion:

This study results analyzing the present situation of covid-19 in India, by visualizing through plots and forecasting cases of next 30 days using fbprophet library.

Our Dataset containing the recorded Covid-19 data of states and union territories of India, updates on daily basis provided by 'covid19.org'. Our dashboard functions are defined in such a way that they are adapted to change in time series and gives desired output nevertheless, when ever you run the notebook, provides the legitimate information until today.

Following plot depicts the situation of selected state until now among different categories such as number of confirmed, deceased, recovered and also the Active cases.

Dashboard



Past Analysis:



Prediction Analysis:



```
[7]: <function main .analy(select analysis)>
```

References :

1. CSV dataset from covid19.org maintained by [Ministry of Health & Family Welfare](#).
2. Testing data from [covid19india](#).
3. Geojson file extracted and modified from github repository to plot choropleths (India Map).
4. Documentation of fbprophet, plotly and ipywidgets.
5. Youtube videos of 365Datascience.