NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC
: 08258 - 281039 – 281263, Fax: 08258 – 281265

Department of Computer Science and Engineering

B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

Mini Project on

Covid-19 Forecasting

Course code:18CS605
Course Title:Machine Learning Lab
Semester:6
Section:A

Submitted To:

Ms Divya Jennifer D'souza
Assistant Professor
Department of Computer Science
and Engineering

Submitted By:

Adarsh J Padmashali
4NM18CS003
Ajay Kumar
4NM18CS010

Date of Submission:

29-05-2021

NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC
: 08258 - 281039 – 281263, Fax: 08258 – 281265

Department of Computer Science and Engineering

B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

CERTIFICATE

It is certified that the mini project entitled **Covid-19 Forecasting** is a bonafide work carried out by

Adarsh J Padmashali 4NM18CS003
Ajay Kumar 4NM18CS010

In partial fulfillment of requirements for the award of Bachelor of Engineering Degree in computer science and engineering prescribed by VTU,Belgaum.

It is certified that all corrections indicated for Internal Assessment have been incorporated in the report.

The mini project report have been approved as it satisfies the academic requirements in respect of the project work prescribed for Bachelor of Engineering Degree

Signature of Guide	**Signature of HOD**
Ms Divya Jennifer D'souza	Dr Jyothi Shetty
Assistant Professor	Head of Department
Department of CSE	Department of CSE

Abstract

Humanity has faced pandemics since the beginning of time. The twentieth century saw multiple influenza pandemics, and now we are facing a COVID-19 pandemic caused by a coronavirus.

Coronaviruses are not new to humans or even to you. Coronaviruses are a family of viruses best identified by the crown-like spikes that cover their surface (corona is Latin for 'crown'). Coronaviruses cause upper-respiratory tract illnesses like the common cold and the 2003 SARS and 2012 MERS outbreaks. In the winter of 2019, a new coronavirus, now officially called SARS-CoV-2, emerged in Wuhan, China. The virus made the jump from animals to humans and causes a disease called COVID-19. For some people, often children and young adults, SARS-CoV-2 causes few or no symptoms. For others it can lead to severe lung damage and even death. The virus can be spread fairly easily, including by people who are infected but display no symptoms, and as a result, we are in the middle of a global pandemic, with nearly all countries in the world reporting an increasing number of infected individuals. Scientists and health professionals around the globe are working hard to rapidly learn more about this new coronavirus and the disease it causes and to prevent the spread of COVID-19.

So we as engineering students wish to help the Govt and Private entities by forecasting covid-19 cases and deaths in India.We will be using machine learning approach to forecast cases and deaths.Our model will be capable to forecast the cases and deaths upto 14 days.

Table of Contents

Introduction

The current pandemic has forced a sudden and unimaginable turnaround on everyone , at all levels.Our lives,primarily in the way we behaved and related in different aspects of our professional and personal lives have been strongly changed.Now, we are still in the middle of this pandemic , not knowing when and how it will end.

No one in the world can predict when this pandemic will end.However, we have developed a Machine Learning model which can forecast covid-19 cases and deaths in India.This forecast can be used by the Government to take several decisions like lockdowns,improving health infrastructure in advance.

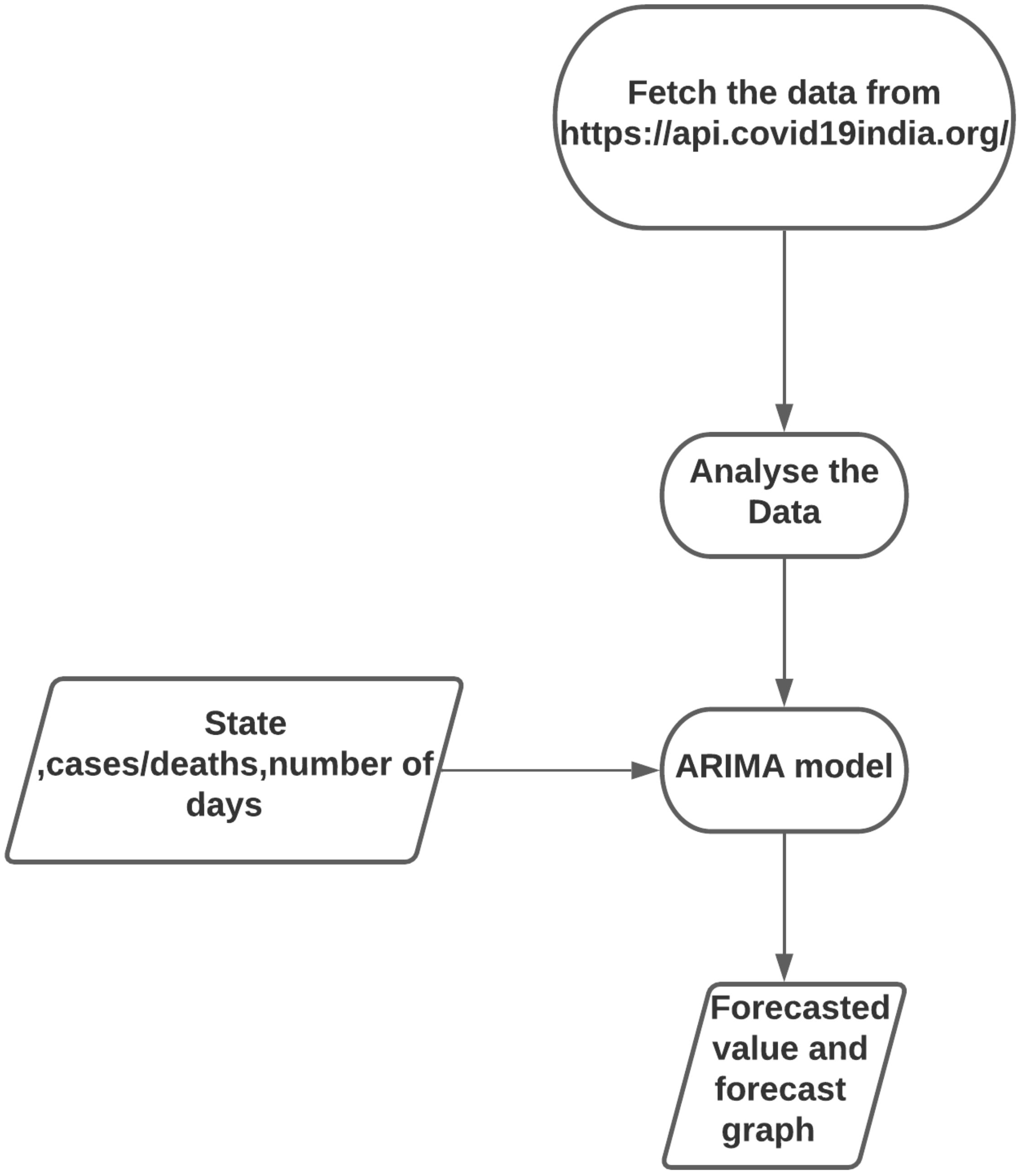
Time series forecasting algorithm **Auto Regressive Integrated Moving Average(ARIMA)** has been used to forecast the cases and deaths in India.User will have the option to select state and number of days of forecast(7 or 14 day forecast)

Requirements

- **Software Requirements**
  - Operating System:Any
  - Python 3.0 or Above
    - Numpy
    - Pandas
    - Matplotlib
    - pmdarima
    - Datetime
    - Flask
  - Browser : Any

- **Hardware Requirements**
  - RAM : Minimum 4GB
  - Processor : Intel i3 or Above

Design


Fetch the data from https://api.covid19india.org/
Analyse the Data
State ,cases/deaths,number of days
ARIMA model
Forecasted value and forecast graph

Fetching data :

- Dataset which we are using for this project is real time data.

- We are using two datasets from https://api.covid19india.org/
  - https://api.covid19india.org/csv/latest/state_wise_daily.csv
    - Consists of each state's data on daily confirmed cases ,daily recovered cases and daily deceased cases from 14/03/2020.
  - https://api.covid19india.org/csv/latest/state_wise.csv
    - Consists of each state’s data on total cases,recoveries,deaths and state codes

Analyzing Data:

- In this section , we get to know some details of the dataset.
- Datatype of each column
- We find the relationship between columns by data visualization

ARIMA Model:

- Using the ARIMA model, you can forecast a time series using the series past values.
- A time series is a sequence where a metric is recorded over regular time intervals.
- Depending on the frequency, a time series can be of yearly (ex: annual budget), quarterly (ex: expenses), monthly (ex: air traffic), weekly (ex: sales qty), daily (ex: weather), hourly (ex: stocks price), minutes (ex: inbound calls in a call center) and even seconds wise (ex: web traffic).
- Forecasting is the next step where you want to predict the future values the series is going to take.
- Now forecasting a time series can be broadly divided into two types.If you use only the previous values of the time series to predict its future values, it is called **Univariate Time Series Forecasting**.
- And if you use predictors other than the series (a.k.a exogenous variables) to forecast it is called **MultiVariate Time Series Forecasting**.

- ARIMA, short for ‘AutoRegressive Integrated Moving Average’, is a forecasting algorithm based on the idea that the information in the past values of the time series can alone be used to predict the future values.
- ARIMA is actually a class of models that ‘explains’ a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values.
- Any ‘non-seasonal’ time series that exhibits patterns and is not a random white noise can be modeled with ARIMA models.
- ***Auto Regressive:***
    - Autoregression is a time series model that uses observations from previous time steps as input to a regression equation to predict the value at the next time step.
    - A regression model, such as linear regression, models an output value based on a linear combination of input values.
    - Eg: $y=a_0+a_1x$ Where y is the prediction, a_0 and a_1 are coefficients found by optimizing the model on training data, and x is an input value.
    - This technique can be used on time series where input variables are taken as observations at previous time steps, called lag variables.
    - For example, we can predict the value for the next time step (t+1) given the observations at the last two time steps (t-1 and t-2). As a regression model, this would look as follows: $x(t+1)=a_0+a_1x(t-1)+a_2x(t-2)$
    - Because the regression model uses data from the same input variable at previous time steps, it is referred to as an autoregression (regression of self).
    - We denote it as AR(p), where “p” is called the order of the model and represents the number of lagged values we want to include.
    - For instance, if we take X as time-series variable, then an AR(1), also known as a simple autoregressive model, would look something like this: $X_t = C + \phi_1 X_{t-1} + \epsilon_t$

- X_{t-1} represents the value of X during the previous period.
- The coefficient ϕ_1 is a numeric constant by which we multiply the lagged variable (X_{t-1}). You can interpret it as the part of the previous value which remains in the future
- ϵ_t- It's called the residual and represents the difference between our prediction for period t and the correct value ($\epsilon_t = y_t - \hat{y}_t$). These residuals are usually unpredictable differences because if there's a pattern, it will be captured in the other incumbents of the model.
- AR(2)=$X_t = C + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \epsilon_t$

- ***Moving Average:***
    - In time-series, we sometimes observe similarities between past errors and present values. That's because certain unpredictable events happen, and they need to be accounted for.
    - In other words, by knowing how far off our estimation yesterday was, compared to the actual value, we can tweak our model, so that it responds accordingly.
    - A simple Moving Average (MA) model looks like this:$r_t = c + \theta_1 \epsilon_{t-1} + \epsilon_t$
    - r is some time series variable
    - r_t represents the values of "r" in the current period - t
    - θ1 is a numeric coefficient for the value associated with the 1st lag
    - ϵ_t and ϵ_{t-1} which represent the errors for the current and the previous period, respectively.
- Now, in both models, we have the present-day value r_t equal a sum of a constant c; some error term ϵ_t; and a lagged value (r_{t-1}, ϵ_{t-1}) multiplied by an assigned coefficient (ϕ_1, θ_1). The only major difference is that the autoregressive model uses the value of the variable (r_{t-1}), while the moving average model relies on the residual (ϵ_{t-1}).
- **Auto Regressive Integrated Moving Average:**
    - Let's suppose that "Y" is some random time-series variable. Then, a simple Autoregressive Moving Average model would look something like this: $y_t = c + \phi_1 y_{t-1} + \theta_1 \epsilon_{t-1} + \epsilon_t$

  - y_t and y_{t-1} represent the values in the current period and 1 period ago respectively
  - ϵ_t and ϵ_{t-1} are the error terms for the same two periods. The error term from the last period is used to help us correct our predictions. By knowing how far off we were in our last estimate, we can make a more accurate estimation this time.
  - "c" is just a baseline constant factor. It simply means we can plug in any constant factor here. If the equation doesn't have such a baseline, we just assume c=0.
  - ϕ_1, expresses on average what part of the value last period (y_{t-1}) is relevant in explaining the current one. Similarly, the latter, θ_1, represents the same for the past error term (ϵ_{t-1}).
  - an ARMA (P, Q) model, takes the previous values up to P periods ago, but also takes the residuals of up to Q lags.
  - the equation for an ARMA (2,3) would look like this:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \theta_1 \epsilon_{t-1} + + \theta_2 \epsilon_{t-2} + \theta_3 \epsilon_{t-3} + \epsilon_t$$

  - However,there is a additional parameter d,called degree of differencing ie number of times that the raw observations are differenced
- **Auto ARIMA**
  - Usually, in the basic ARIMA model, we need to provide the p,d, and q values which are essential. We use statistical techniques to generate these values by performing the difference to eliminate the non-stationarity
  - In Auto ARIMA, the model itself will generate the optimal p, d, and q values which would be suitable for the data set to provide better forecasting.
  - We will be using Auto ARIMA for this project.
- Input for our Auto ARIMA model will be
  - State
  - Confirmed cases/deaths
  - Days(7 or 14)
- Output will be forecast value with lower and upper limit.And also a forecast plot.

Implementation

Importing necessary libraries

```
from flask import Flask,render_template,request
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import matplotlib.dates as mdates
import datetime
import sklearn
from statsmodels.tsa.arima_model import ARIMA
import pmdarima as pm
import datetime as dt
from io import BytesIO
import base64
```

Reading Dataset 1 which contains all states daily confirmed cases,deaths,recoveries

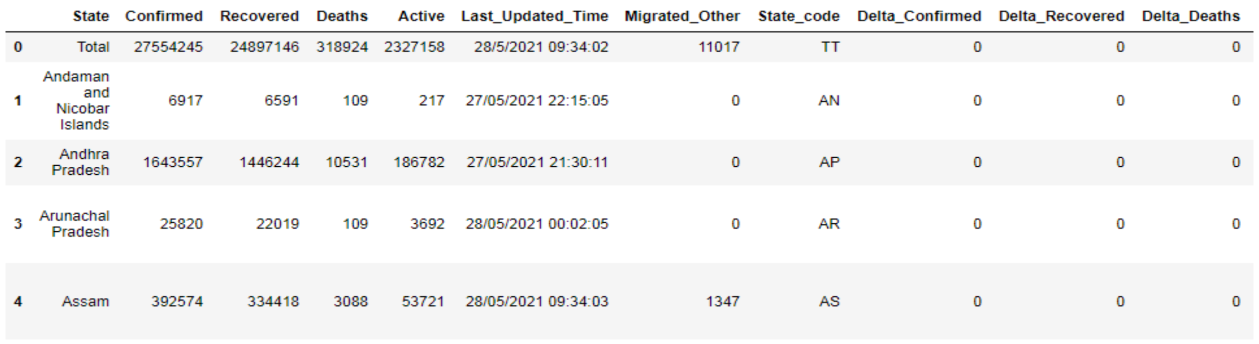
```
data1=pd.read_csv('https://api.covid19india.org/csv/lat
est/state_wise_daily.csv')
data1.head()
```

	Date	Date_YMD	Status	TT	AN	AP	AR	AS	BR	CH	...	PB	RJ	SK	TN	TG	TR	UP	UT	WB	UN
0	14-Mar-20	2020-03-14	Confirmed	81	0	1	0	0	0	0	...	1	3	0	1	1	0	12	0	0	0
1	14-Mar-20	2020-03-14	Recovered	9	0	0	0	0	0	0	...	0	1	0	0	0	0	4	0	0	0
2	14-Mar-20	2020-03-14	Deceased	2	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	15-Mar-20	2020-03-15	Confirmed	27	0	0	0	0	0	0	...	0	1	0	0	2	0	1	0	0	0
4	15-Mar-20	2020-03-15	Recovered	4	0	0	0	0	0	0	...	0	2	0	0	1	0	0	0	0	0

Reading data2 which contains state code

```
data2=pd.read_csv('https://api.covid19india.org/csv/lat
est/state_wise.csv')
indexNames = data2[data2['State_code'] == 'UN'].index
#Removing the row "state unassigned"(UN)
data2.drop(indexNames,inplace=True)
```

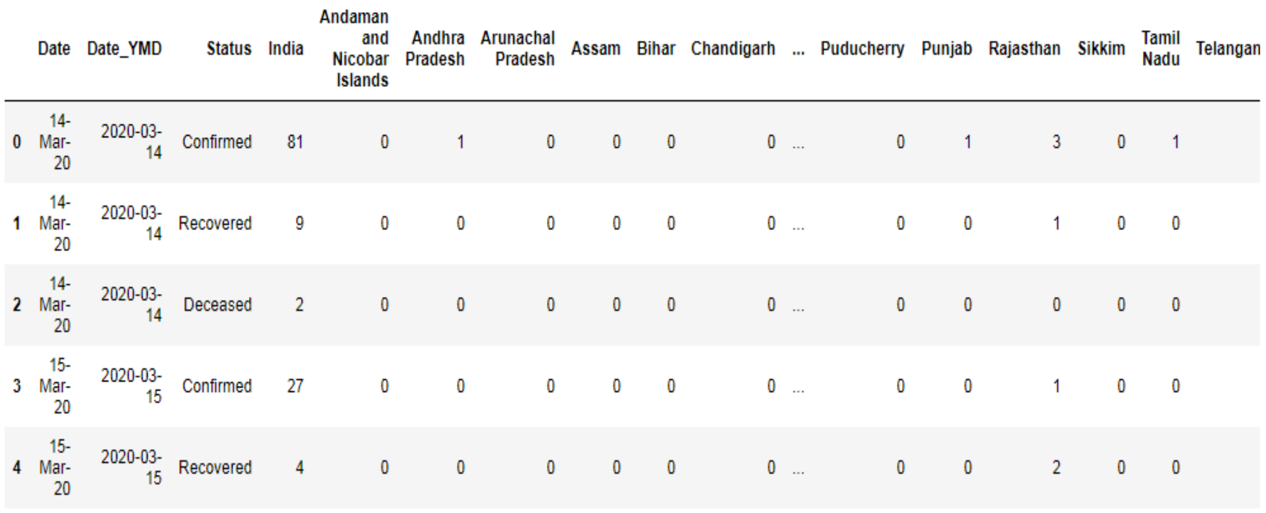
```
data2.head()
```



	State	Confirmed	Recovered	Deaths	Active	Last_Updated_Time	Migrated_Other	State_code	Delta_Confirmed	Delta_Recovered	Delta_Deaths
0	Total	27554245	24897146	318924	2327158	28/5/2021 09:34:02	11017	TT	0	0	0
1	Andaman and Nicobar Islands	6917	6591	109	217	27/05/2021 22:15:05	0	AN	0	0	0
2	Andhra Pradesh	1643557	1446244	10531	186782	27/05/2021 21:30:11	0	AP	0	0	0
3	Arunachal Pradesh	25820	22019	109	3692	28/05/2021 00:02:05	0	AR	0	0	0
4	Assam	392574	334418	3088	53721	28/05/2021 09:34:03	1347	AS	0	0	0

Renaming the columns of data1 ie state code to name

```
df=pd.DataFrame()
df=data2['State']
df=df.rename_axis('Index').reset_index()
df.pop('Index')
df.iloc[0,0]='India'
l=(data1.columns) #Getting the columns
l=l[3:] # First 3 columns not required
for i in range(len(l)):
    data1=data1.rename(columns = {l[i]:df.iloc[i,0]})
data1.head()
```



	Date	Date_YMD	Status	India	Andaman and Nicobar Islands	Andhra Pradesh	Arunachal Pradesh	Assam	Bihar	Chandigarh	...	Puducherry	Punjab	Rajasthan	Sikkim	Tamil Nadu	Telangan
0	14-Mar-20	2020-03-14	Confirmed	81	0	1	0	0	0	0	...	0	1	3	0	1	
1	14-Mar-20	2020-03-14	Recovered	9	0	0	0	0	0	0	...	0	0	1	0	0	
2	14-Mar-20	2020-03-14	Deceased	2	0	0	0	0	0	0	...	0	0	0	0	0	
3	15-Mar-20	2020-03-15	Confirmed	27	0	0	0	0	0	0	...	0	0	1	0	0	
4	15-Mar-20	2020-03-15	Recovered	4	0	0	0	0	0	0	...	0	0	2	0	0	

Get the data of India

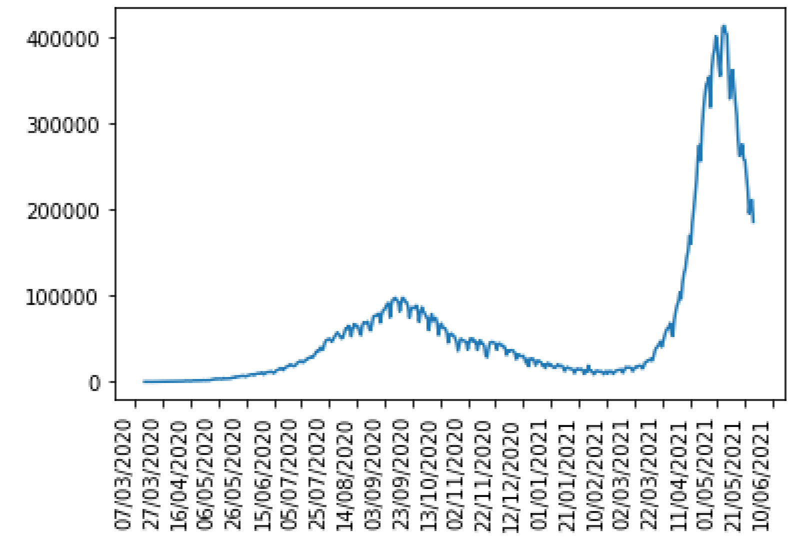
```
li1=[] # date
```

```
li2=[] # Confirmed cases
li3=[] # deaths
for i in range(len(data1)):
    if data1.loc[i,'Status']=='Confirmed':     #Status is
Confirmed
        li2.append(data1.loc[i,'India'])
        li1.append(data1.loc[i,'Date_YMD'])
    if data1.loc[i,'Status']=='Deceased':
        li3.append(data1.loc[i,'India'])

data_state=pd.DataFrame(list(zip(li1,li2,li3)),columns=['Dat
e','Confirmed Cases','Death'])
data_state.head()
```

	Date	Confirmed Cases	Death
0	2020-03-14	81	2
1	2020-03-15	27	0
2	2020-03-16	15	0
3	2020-03-17	11	1
4	2020-03-18	37	0

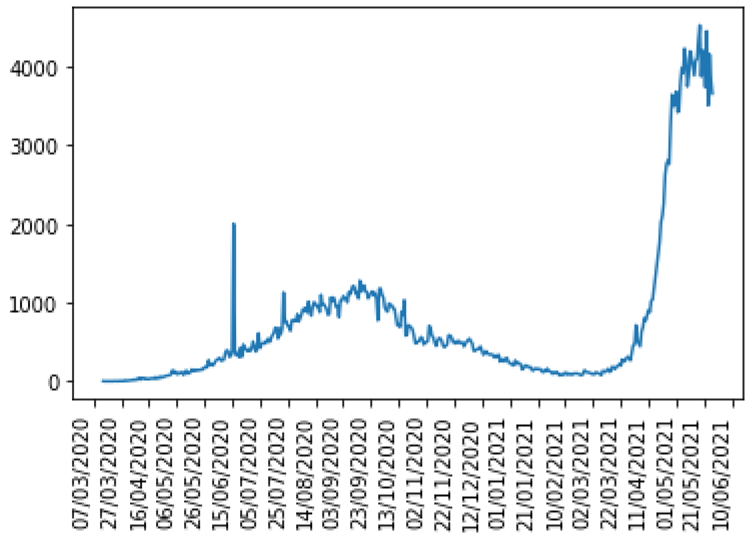
Plot of daily cases vs Date for India

```
    x = [dt.datetime.strptime(d,'%Y-%m-%d').date() for d in
    data_state['Date']]
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatte
    r('%d/%m/%Y'))
    plt.gca().xaxis.set_major_locator(mdates.DayLocator(int
    erval=20))
    plt.plot(x,data_state['Confirmed Cases'])
    plt.gcf().autofmt_xdate(rotation=90)
    plt.show()
```



Plot daily deaths vs Death for India

```
plt.gca().xaxis.set_major_formatter(mdates.DateFormatte
r('%d/%m/%Y'))
plt.gca().xaxis.set_major_locator(mdates.DayLocator(int
erval=20))
plt.plot(x,data_state['Death'])
plt.gcf().autofmt_xdate(rotation=90)
plt.show()
```



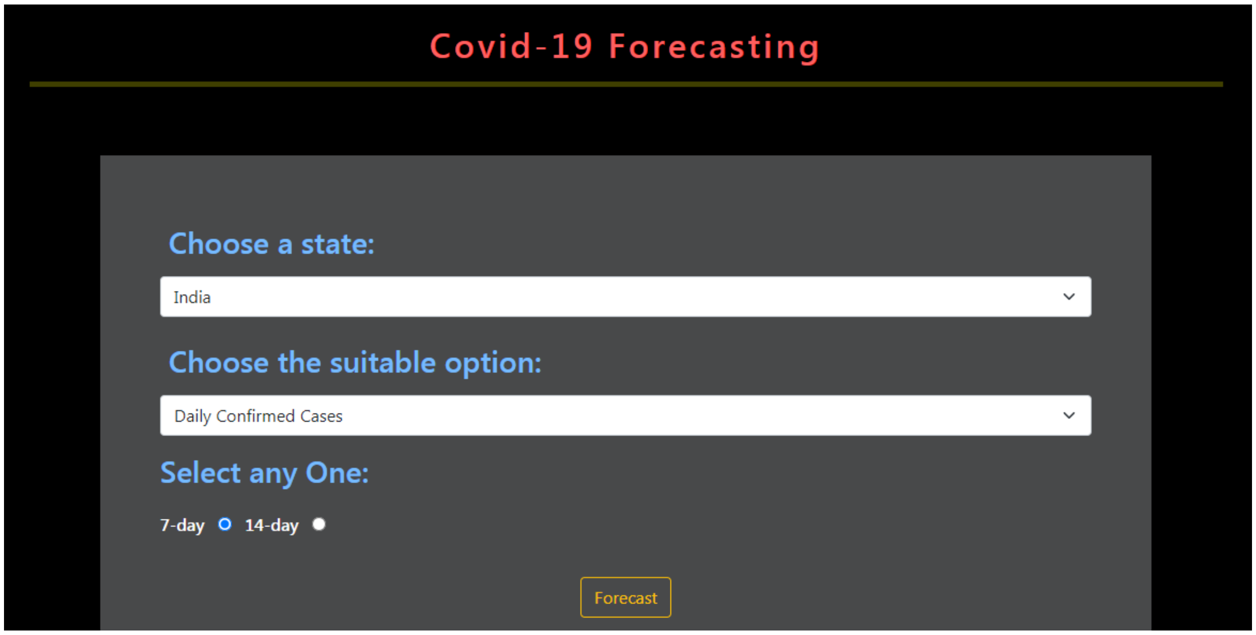
Defining the forecast function which takes state name,number of days,confirmed cases/death as input and it will return a dataframe having date,lower value,upper value,predicted value

```
def forecast(state,attr,days):
    print("For state:",state)
    li1=[]     #Date
    li2=[]     #Confirmed
    li3=[]    #Deaths
    for i in range(len(data1)):
        if data1.loc[i,'Status']=='Confirmed':     #Status
is Confirmed
            li2.append(data1.loc[i,state])
            li1.append(data1.loc[i,'Date_YMD'])
        if data1.loc[i,'Status']=='Deceased':
            li3.append(data1.loc[i,state])


data_state=pd.DataFrame(list(zip(li1,li2,li3)),columns=['Dat
e','Confirmed Cases','Death'])
n_periods = days

x=[dt.datetime.strptime(data_state.iloc[-1]['Date'],'%Y-%m-%
d').date()]

for i in range(n_periods):
    x.append(x[-1]+dt.timedelta(days=1))
x.pop(0)

data_state.set_index('Date',inplace=True)
model = pm.auto_arima(data_state[attr], start_p=1,
start_q=0,
test='adf',       # use adf test to find optimal 'd'
max_p=50, max_q=50, # maximum p and q
m=1,              # frequency of series
d=None,           # let model determine 'd'
seasonal=False,   # No Seasonality
start_P=0,
D=0,
trace=True,
error_action='ignore',
suppress_warnings=True,
stepwise=True)

fc, confint
=model.predict(n_periods=n_periods,return_conf_int=True)
fc=fc.astype(int)
confint=confint.astype(int)
index_of_fc = np.arange(len(data_state[attr]),
len(data_state[attr])+n_periods)


 # make series for plotting purpose
 fc_series = pd.Series(fc, index=index_of_fc)
 lower_series = pd.Series(confint[:, 0], index=index_of_fc)
 upper_series = pd.Series(confint[:, 1], index=index_of_fc)

img = BytesIO()
with plt.style.context(('dark_background')):
     fig=plt.figure(figsize=(12,10))
     plt.gca().xaxis.set_major_locator(mdates.DayLocator(int
     erval=3))
        plt.plot(data_state[attr])
        plt.plot(fc_series, color='red')
        plt.gcf().autofmt_xdate(rotation=90)
        plt.fill_between(lower_series.index,
                    lower_series,
                    upper_series,
                    color='white', alpha=.35)
        plt.grid()
        plt.savefig(img, format='png')
        plt.close()
        img.seek(0)
        plot_url =
base64.b64encode(img.getvalue()).decode('utf8')

    df=pd.DataFrame()
    df['Date']=x
    df['Lower_value']=list(lower_series)
    df['Upper_value']=list(upper_series)
    df['Predicted value']=list(fc_series)
    print(df)
    return df,plot_url
```

Ingerating ML with Flask

```
app = Flask(__name__)
@app.route('/')
def home():
    return
render_template('index.html',states=df['State'])

@app.route('/predict',methods=['POST'])
def hello():
    print('Submitted')
    form_data=request.form
    print(form_data)
    print(form_data['States'])
    if(form_data['days']=='14-day'):
        days=14
    else:
        days=7
    df,plot_url=forecast(form_data['States'],
form_data['cases-deaths'],days)
    return
render_template('result.html',df=df,state=form_data['St
ates'],attr=form_data['cases-deaths'],days=days,plot_ur
l=plot_url)
```

Results



Covid-19 Forecasting

Choose a state:

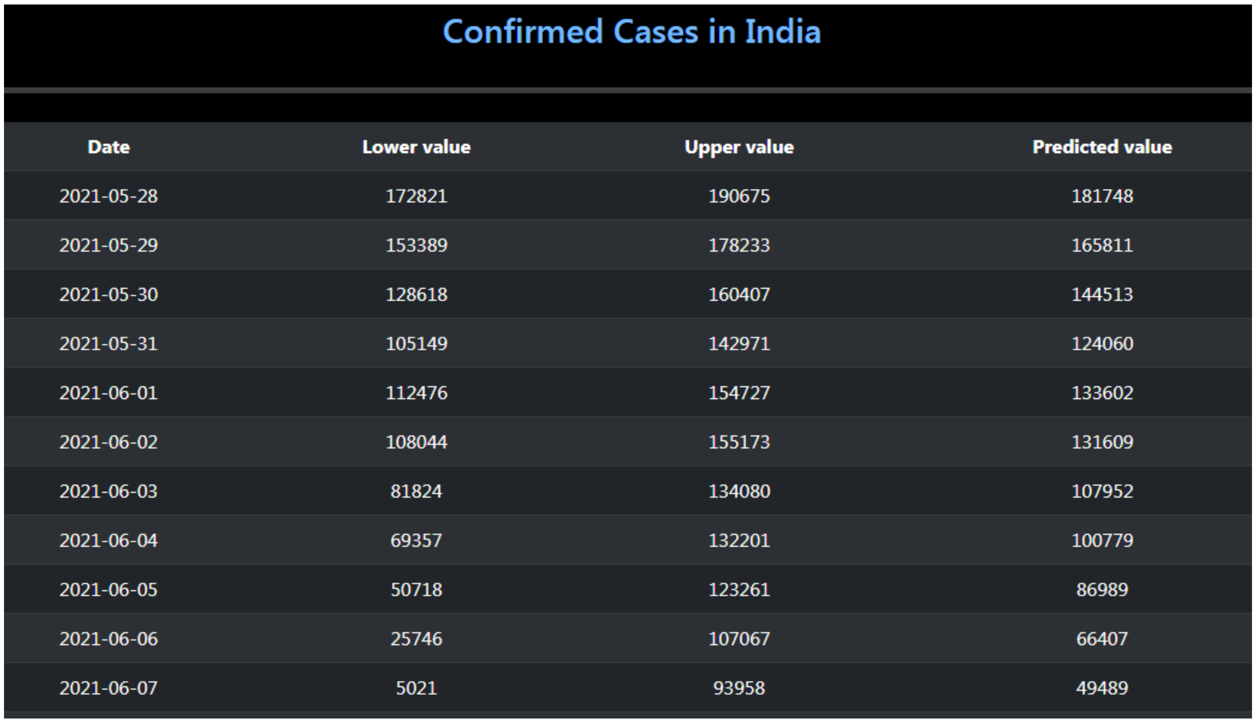
India

Choose the suitable option:

Daily Confirmed Cases

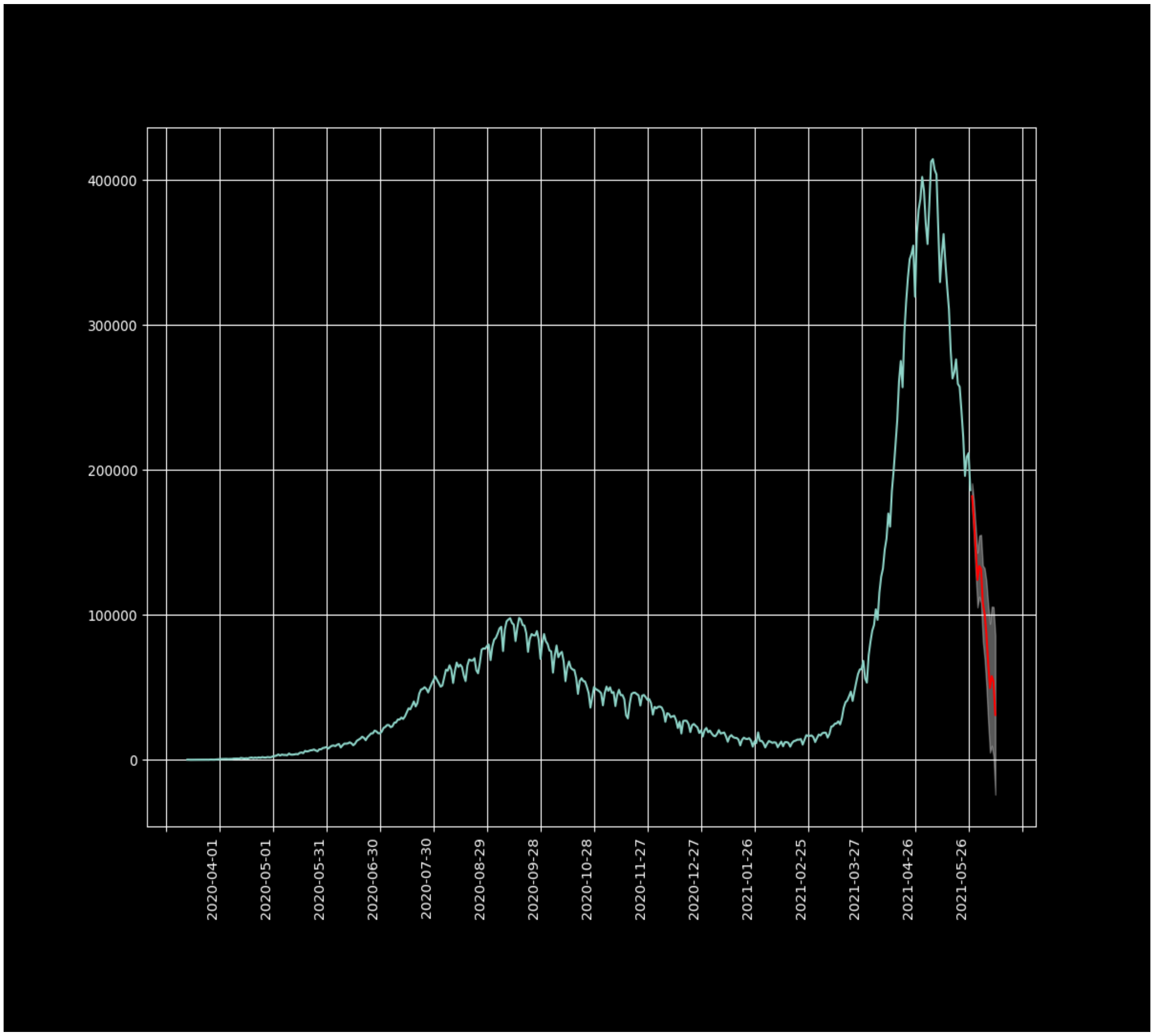
Select any One:

[✓ 7-day] 14-day



Confirmed Cases in India

Date	Lower value	Upper value	Predicted value
2021-05-28	172821	190675	181748
2021-05-29	153389	178233	165811
2021-05-30	128618	160407	144513
2021-05-31	105149	142971	124060
2021-06-01	112476	154727	133602
2021-06-02	108044	155173	131609
2021-06-03	81824	134080	107952
2021-06-04	69357	132201	100779
2021-06-05	50718	123261	86989
2021-06-06	25746	107067	66407
2021-06-07	5021	93958	49489


400000
300000
200000
100000
0
2020-04-01
2020-05-01
2020-05-31
2020-06-30
2020-07-30
2020-08-29
2020-09-28
2020-10-28
2020-11-27
2020-12-27
2021-01-26
2021-02-25
2021-03-27
2021-04-26
2021-05-26

Conclusion

In this country people are having difficulties due to covid-19.Many patients are not able to get medicines,hospital beds.

In our project, we are able to forecast covid-19 cases.This will help the government to take decisions on increasing health infrastructure,lockdowns etc.

References

- https://www.analyticsvidhya.com/blog/2020/10/how-to-create-an-arima-model-for-time-series-forecasting-in-python/
- https://flask.palletsprojects.com/en/2.0.x/tutorial/index.html