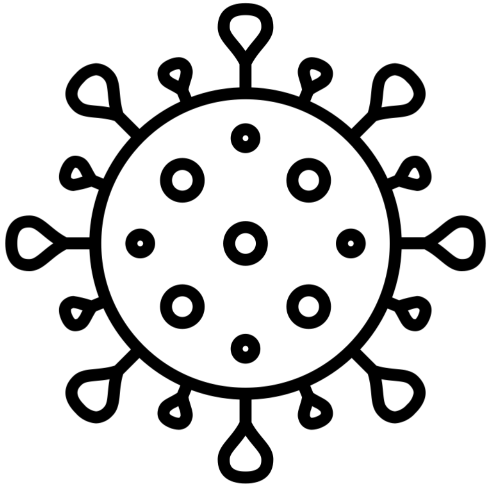
DATA SCIENCE MINI PROJECT

TEAM 11

AGGARWAL ANUSHA

SARAF ISHITA



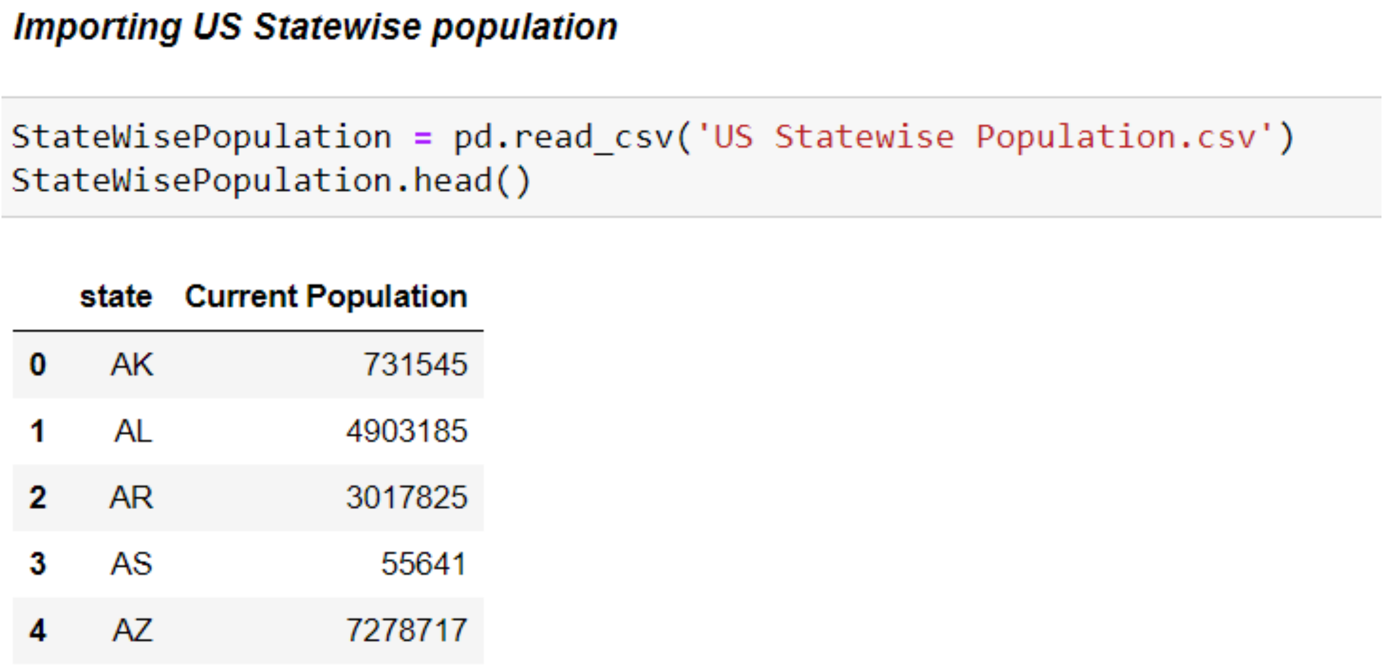
Dataset used: Covid Tracking Project

Aim:

- To help analyze the current COVID situation in different states by means by relevant visualizations

- To predict useful variables such as new cases, active cases, deaths, hospitalizations, probability of an individual getting COVID, etc. to help the state governments, testing qgencies, hospitals, and individuals during the pandemic

Data Collection

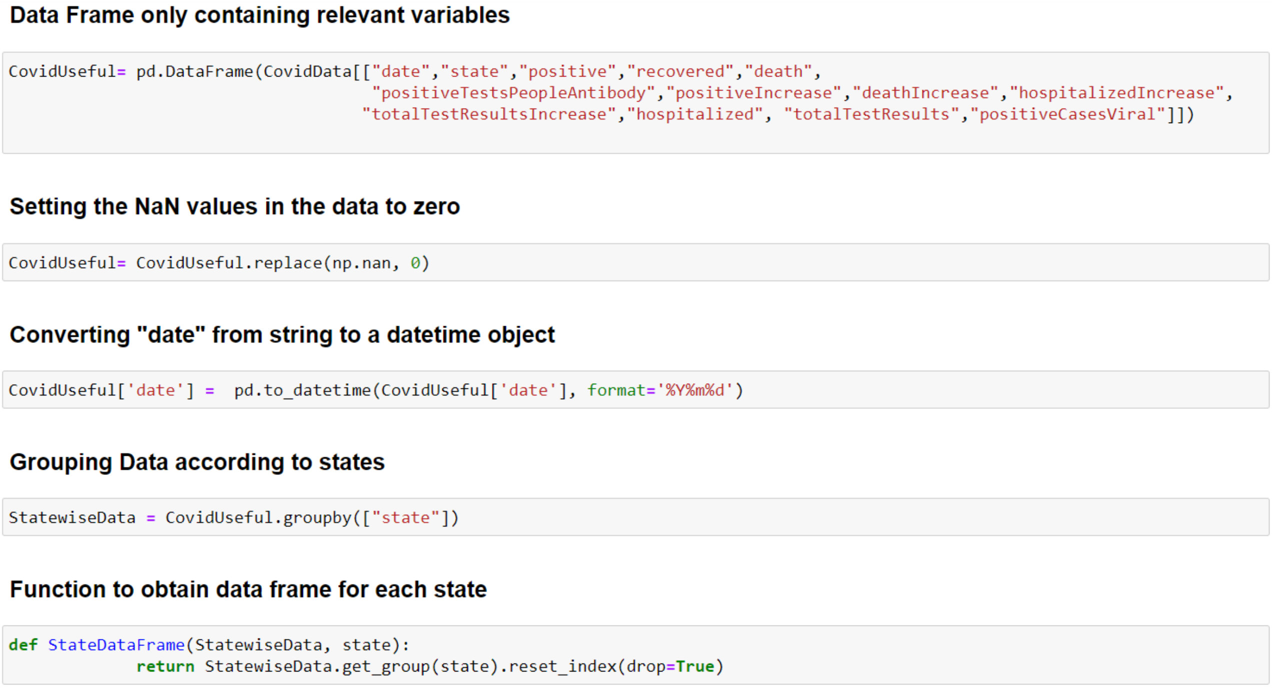


Importing US Statewise population

```
StateWisePopulation = pd.read_csv('US Statewise Population.csv')
StateWisePopulation.head()
```

	state	Current Population
0	AK	731545
1	AL	4903185
2	AR	3017825
3	AS	55641
4	AZ	7278717

Data Cleaning and Data Merging



Data Frame only containing relevant variables

```
CovidUseful= pd.DataFrame(CovidData[["date","state","positive","recovered","death",
                                     "positiveTestsPeopleAntibody","positiveIncrease","deathIncrease","hospitalizedIncrease",
                                    "totalTestResultsIncrease","hospitalized", "totalTestResults","positiveCasesViral"]])
```

Setting the NaN values in the data to zero

```
CovidUseful= CovidUseful.replace(np.nan, 0)
```

Converting "date" from string to a datetime object

```
CovidUseful['date'] =  pd.to_datetime(CovidUseful['date'], format='%Y%m%d')
```

Grouping Data according to states

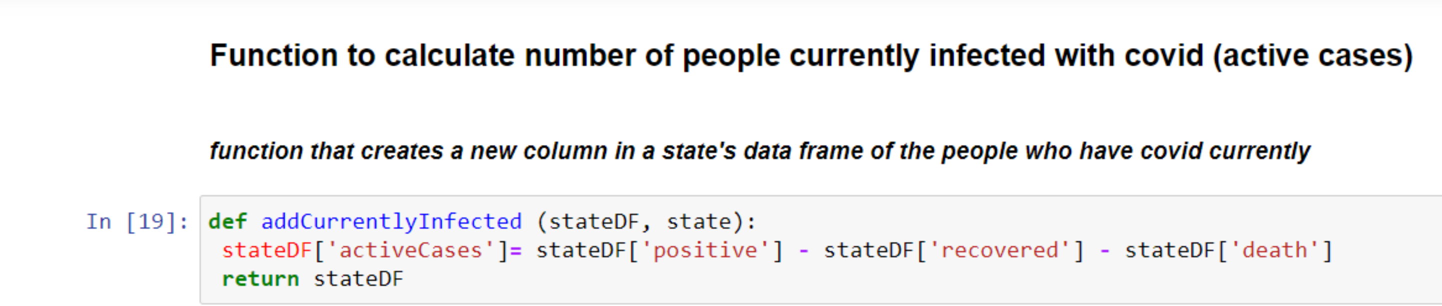
```
StatewiseData = CovidUseful.groupby(["state"])
```

Function to obtain data frame for each state

```
def StateDataFrame(StatewiseData, state):
             return StatewiseData.get_group(state).reset_index(drop=True)
```

Data Curation

CALCULATE NUMBER OF PEOPLE CURRENTLY INFECTED WITH COVID (ACTIVE CASES)



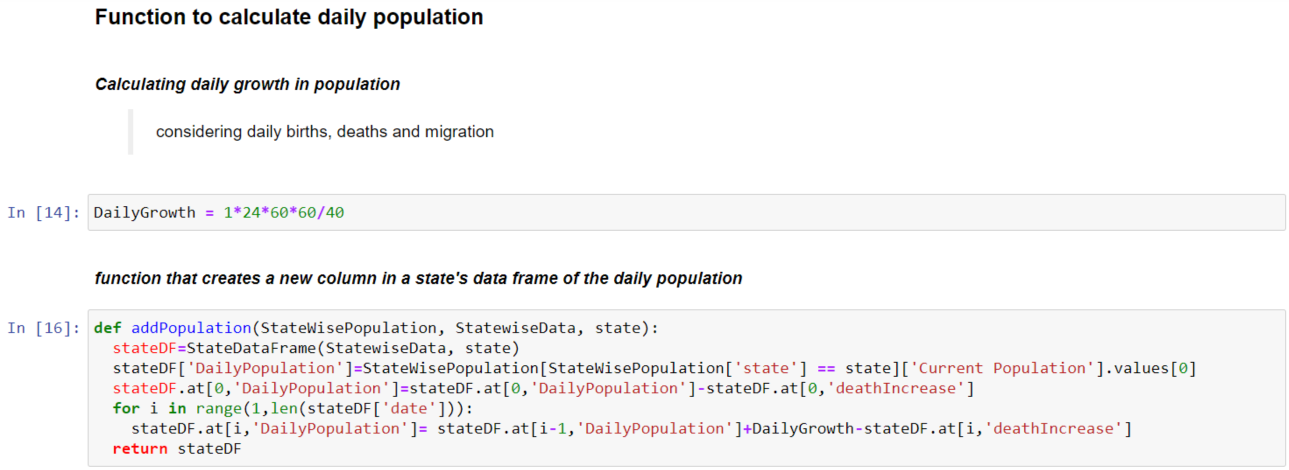
Function to calculate number of people currently infected with covid (active cases)

function that creates a new column in a state's data frame of the people who have covid currently

```
In [19]: def addCurrentlyInfected (stateDF, state):
          stateDF['activeCases']= stateDF['positive'] - stateDF['recovered'] - stateDF['death']
          return stateDF
```

Data Curation

CALCULATE DAILY POPULATION



Function to calculate daily population

Calculating daily growth in population

> considering daily births, deaths and migration

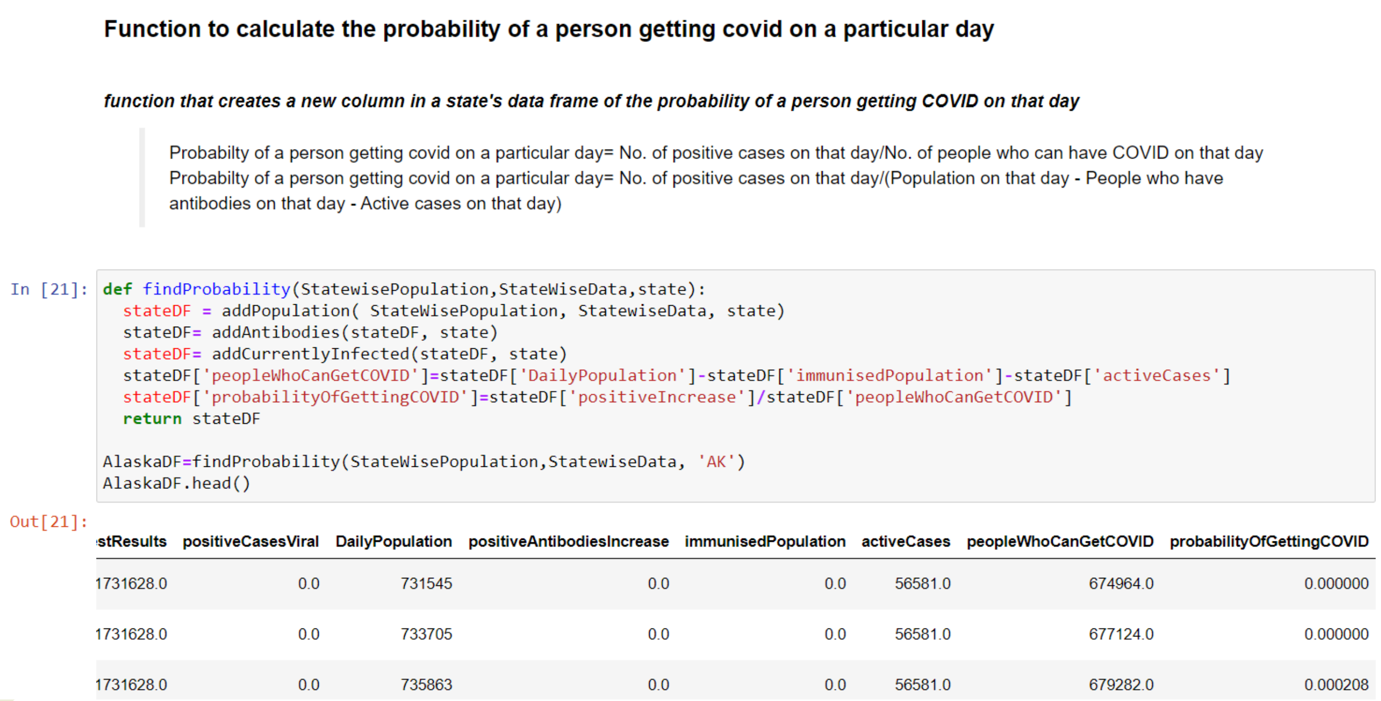
```
In [14]: DailyGrowth = 1*24*60*60/40
```

function that creates a new column in a state's data frame of the daily population

```
In [16]: def addPopulation(StateWisePopulation, StatewiseData, state):
           stateDF=StateDataFrame(StatewiseData, state)
           stateDF['DailyPopulation']=StateWisePopulation[StateWisePopulation['state'] == state]['Current Population'].values[0]
           stateDF.at[0,'DailyPopulation']=stateDF.at[0,'DailyPopulation']-stateDF.at[0,'deathIncrease']
           for i in range(1,len(stateDF['date'])):
             stateDF.at[i,'DailyPopulation']= stateDF.at[i-1,'DailyPopulation']+DailyGrowth-stateDF.at[i,'deathIncrease']
           return stateDF
```

Data Curation

CALCULATE THE PROBABILITY OF A PERSON GETTING COVID ON A PARTICULAR DAY



Function to calculate the probability of a person getting covid on a particular day

function that creates a new column in a state's data frame of the probability of a person getting COVID on that day

> Probabilty of a person getting covid on a particular day= No. of positive cases on that day/No. of people who can have COVID on that day
> Probabilty of a person getting covid on a particular day= No. of positive cases on that day/(Population on that day - People who have antibodies on that day - Active cases on that day)

In [21]:

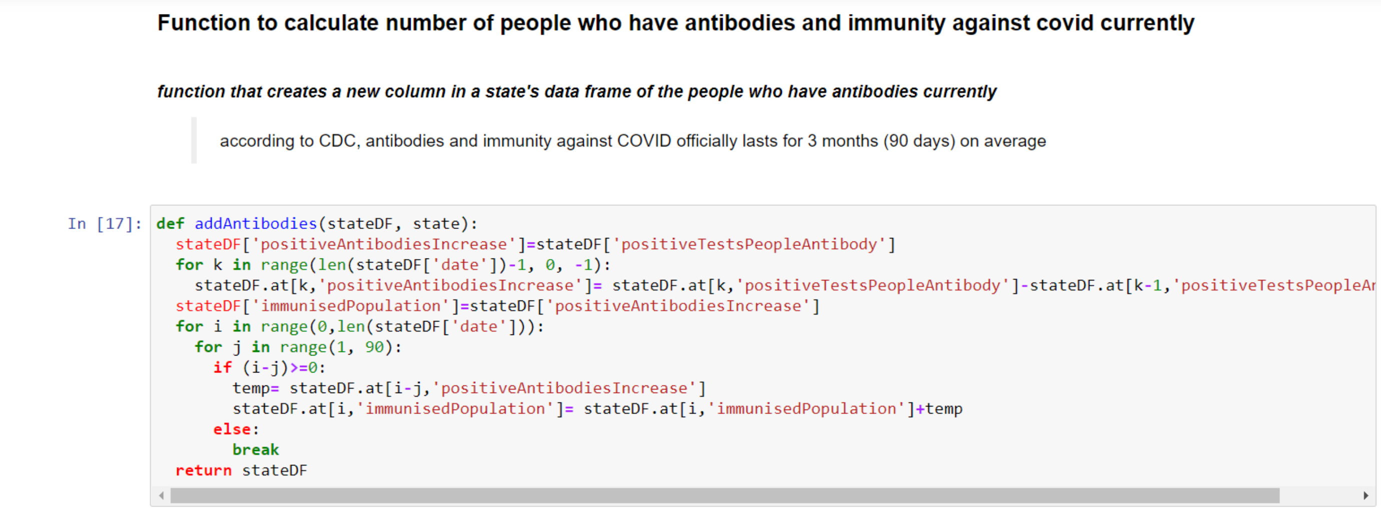
```
def findProbability(StatewisePopulation,StateWiseData,state):
  stateDF = addPopulation( StateWisePopulation, StatewiseData, state)
  stateDF= addAntibodies(stateDF, state)
  stateDF= addCurrentlyInfected(stateDF, state)
  stateDF['peopleWhoCanGetCOVID']=stateDF['DailyPopulation']-stateDF['immunisedPopulation']-stateDF['activeCases']
  stateDF['probabilityOfGettingCOVID']=stateDF['positiveIncrease']/stateDF['peopleWhoCanGetCOVID']
  return stateDF

AlaskaDF=findProbability(StateWisePopulation,StatewiseData, 'AK')
AlaskaDF.head()
```

Out[21]:

stResults	positiveCasesViral	DailyPopulation	positiveAntibodiesIncrease	immunisedPopulation	activeCases	peopleWhoCanGetCOVID	probabilityOfGettingCOVID
1731628.0	0.0	731545	0.0	0.0	56581.0	674964.0	0.000000
1731628.0	0.0	733705	0.0	0.0	56581.0	677124.0	0.000000
1731628.0	0.0	735863	0.0	0.0	56581.0	679282.0	0.000208

Data Curation

CALCULATE NUMBER OF PEOPLE WHO HAVE ANTIBODIES AND IMMUNITY AGAINST COVID



Function to calculate number of people who have antibodies and immunity against covid currently

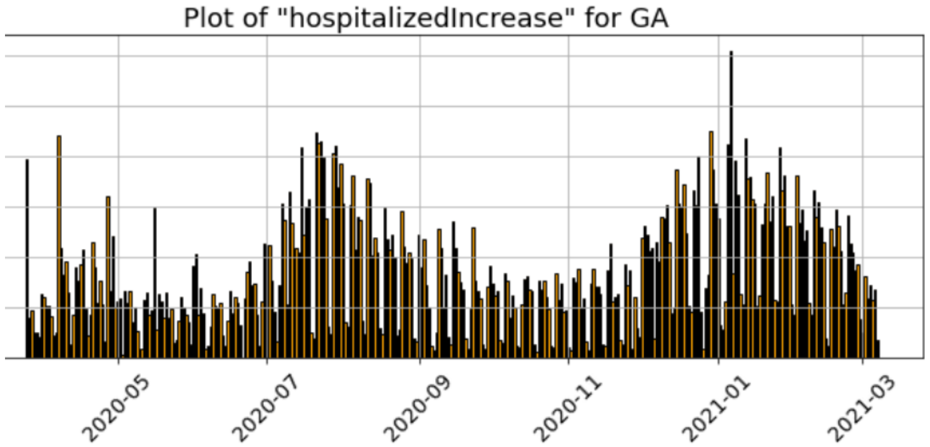
function that creates a new column in a state's data frame of the people who have antibodies currently

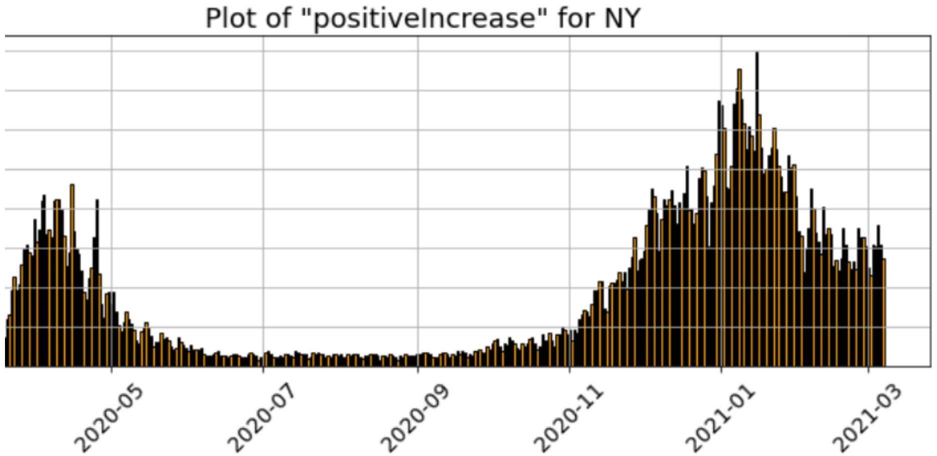
> according to CDC, antibodies and immunity against COVID officially lasts for 3 months (90 days) on average

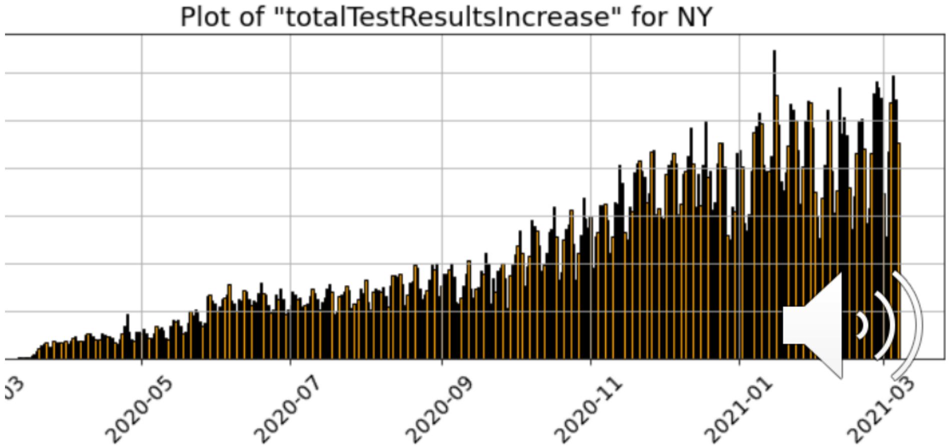
In [17]:
```
def addAntibodies(stateDF, state):
  stateDF['positiveAntibodiesIncrease']=stateDF['positiveTestsPeopleAntibody']
  for k in range(len(stateDF['date'])-1, 0, -1):
    stateDF.at[k,'positiveAntibodiesIncrease']= stateDF.at[k,'positiveTestsPeopleAntibody']-stateDF.at[k-1,'positiveTestsPeopleAr
  stateDF['immunisedPopulation']=stateDF['positiveAntibodiesIncrease']
  for i in range(0,len(stateDF['date'])):
    for j in range(1, 90):
      if (i-j)>=0:
        temp= stateDF.at[i-j,'positiveAntibodiesIncrease']
        stateDF.at[i,'immunisedPopulation']= stateDF.at[i,'immunisedPopulation']+temp
      else:
        break
  return stateDF
```

Exploratory Data Analysis (EDA)

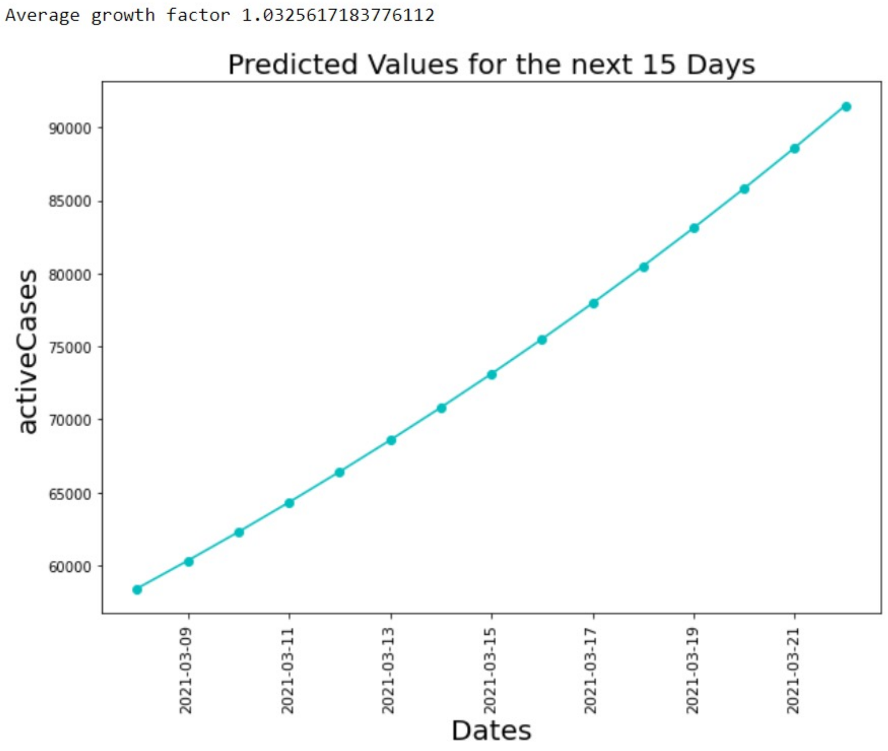
BAR CHARTS







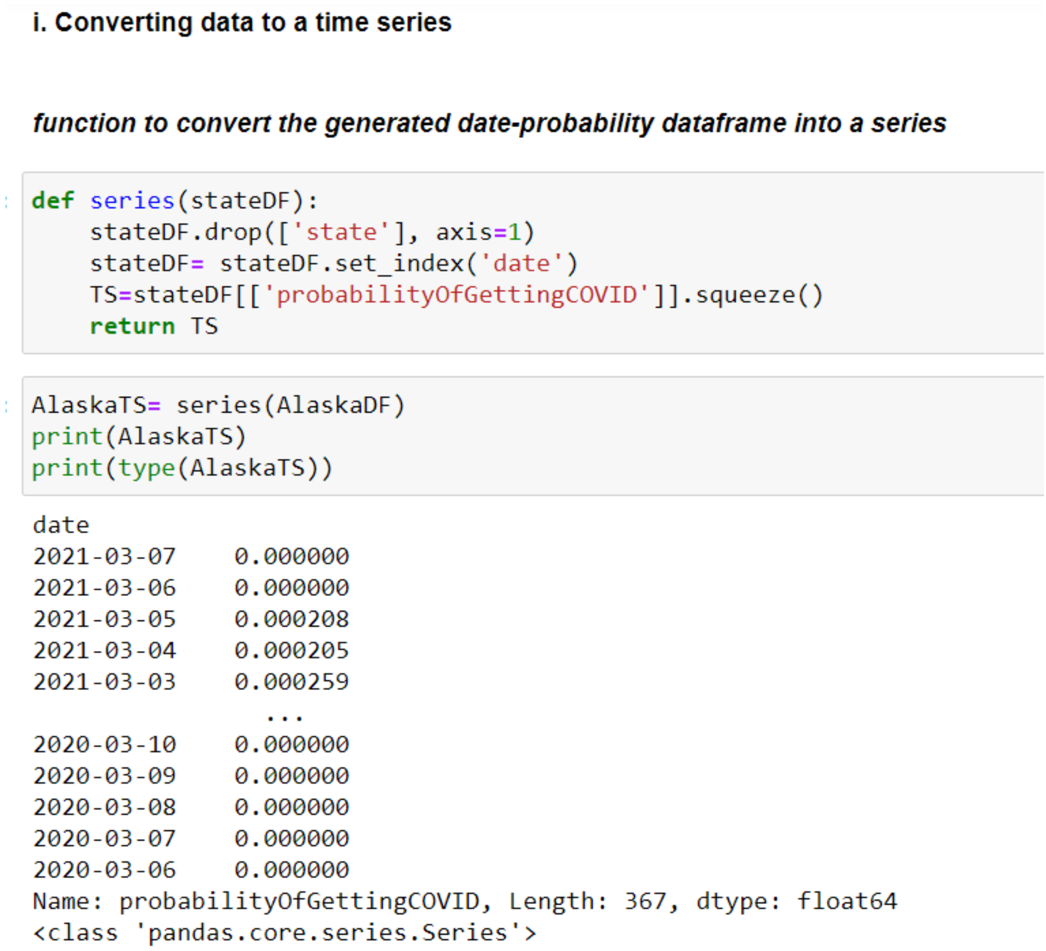
Method 1: Using average growth factor



```
Average growth factor 1.0325617183776112
```

Method 2: ARIMA Model

STEP 1: CONVERTING DATA TO TIME SERIES



i. Converting data to a time series

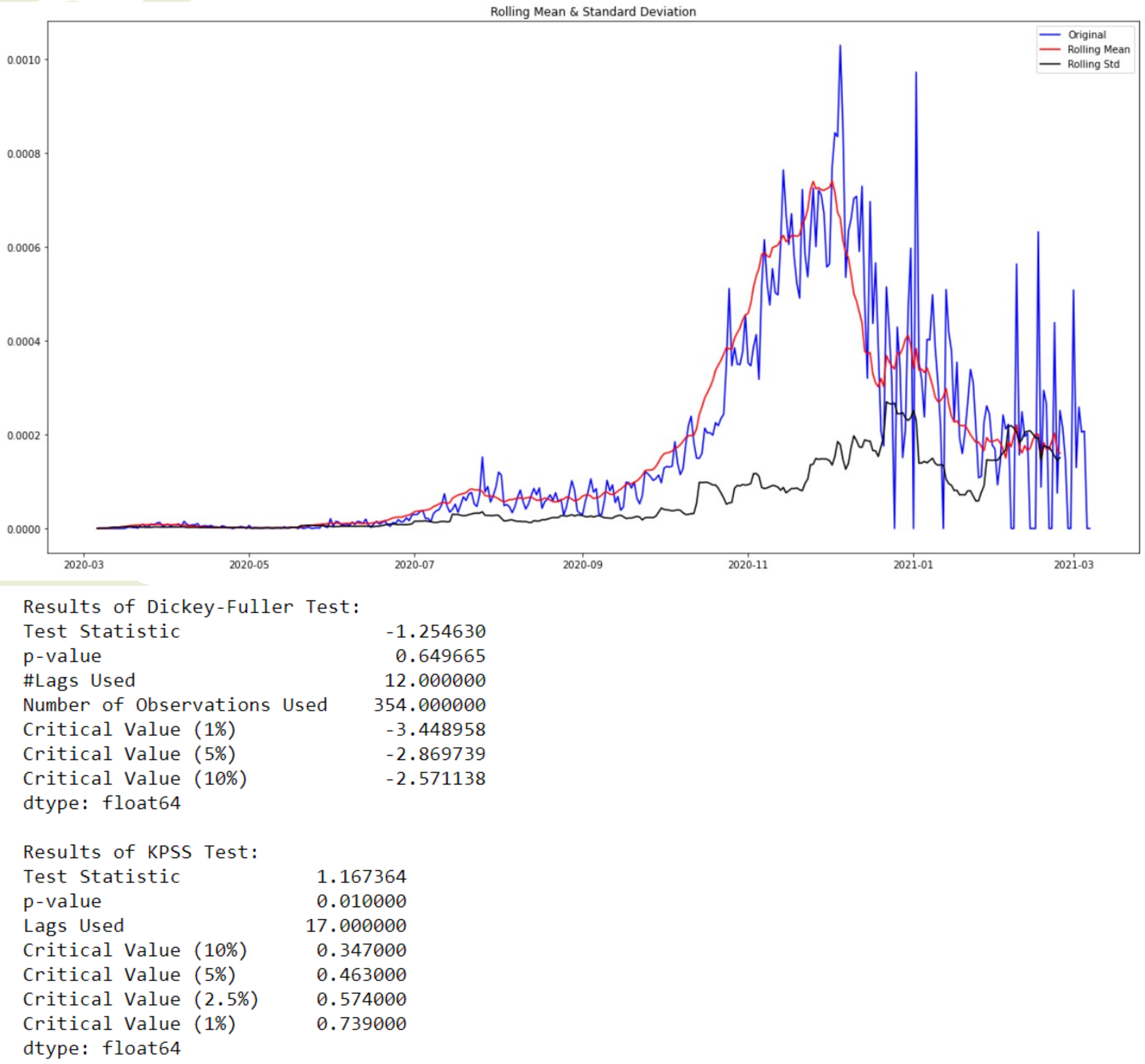
function to convert the generated date-probability dataframe into a series

```
def series(stateDF):
    stateDF.drop(['state'], axis=1)
    stateDF= stateDF.set_index('date')
    TS=stateDF[['probabilityOfGettingCOVID']].squeeze()
    return TS
```

```
AlaskaTS= series(AlaskaDF)
print(AlaskaTS)
print(type(AlaskaTS))
```

```
date
2021-03-07    0.000000
2021-03-06    0.000000
2021-03-05    0.000208
2021-03-04    0.000205
2021-03-03    0.000259
                ...
2020-03-10    0.000000
2020-03-09    0.000000
2020-03-08    0.000000
2020-03-07    0.000000
2020-03-06    0.000000
Name: probabilityOfGettingCOVID, Length: 367, dtype: float64
<class 'pandas.core.series.Series'>
```

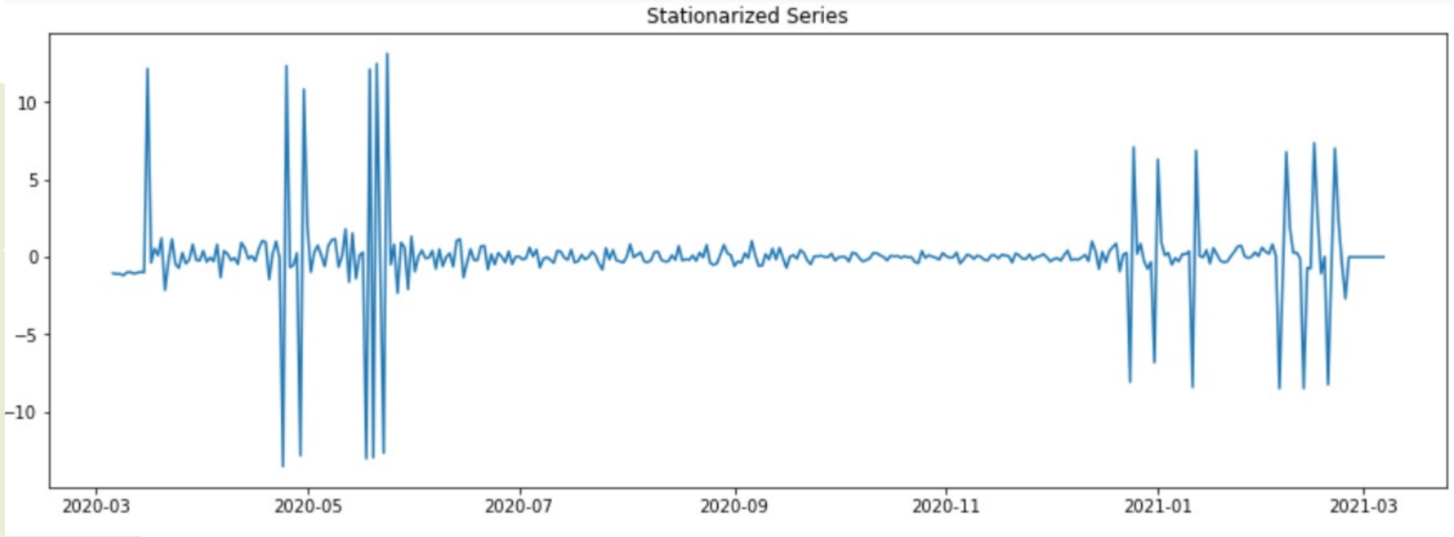
- STEP 2: CHECKING STATIONARITY OF TIME SERIES



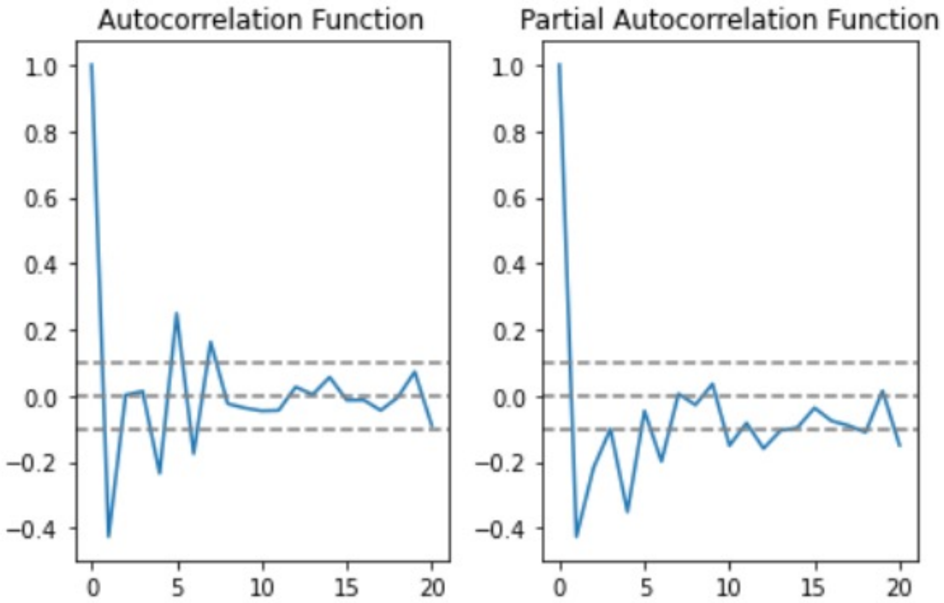
```
Results of Dickey-Fuller Test:
Test Statistic                  -1.254630
p-value                          0.649665
#Lags Used                      12.000000
Number of Observations Used    354.000000
Critical Value (1%)             -3.448958
Critical Value (5%)             -2.869739
Critical Value (10%)            -2.571138
dtype: float64

Results of KPSS Test:
Test Statistic            1.167364
p-value                   0.010000
Lags Used                17.000000
Critical Value (10%)      0.347000
Critical Value (5%)       0.463000
Critical Value (2.5%)     0.574000
Critical Value (1%)       0.739000
dtype: float64
```

Since the Test Statistic of Dickey-Fuller Test is bigger than all the critical values we cannot say that the time series is stationary

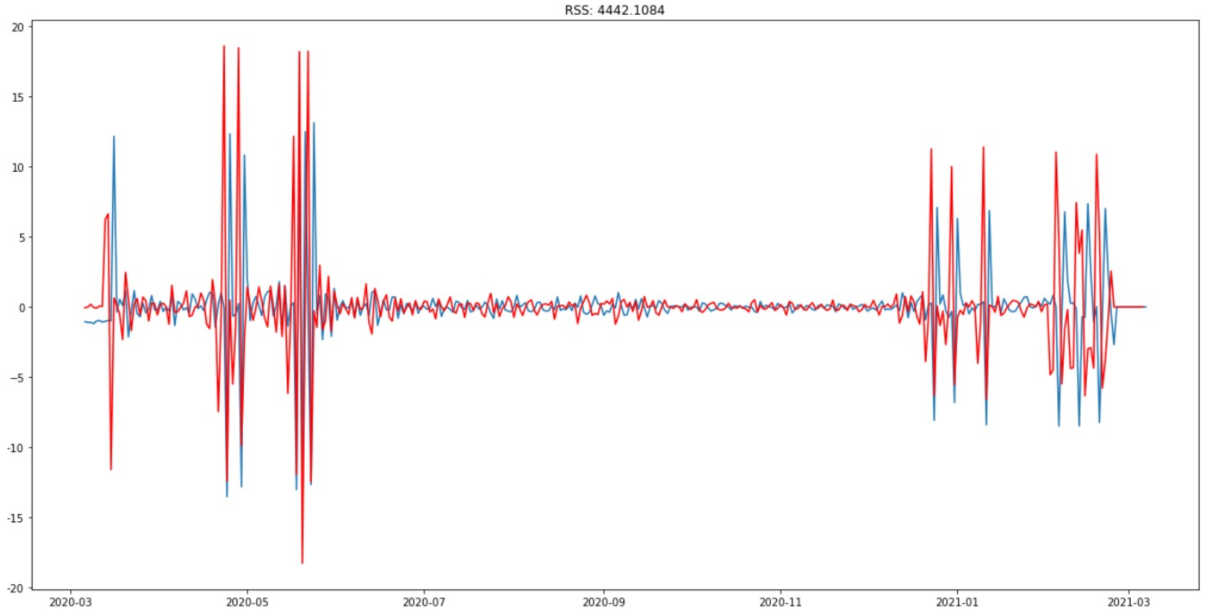
- STEP 3: MAKING THE TIME SERIES STATIONARY


Stationarized Series

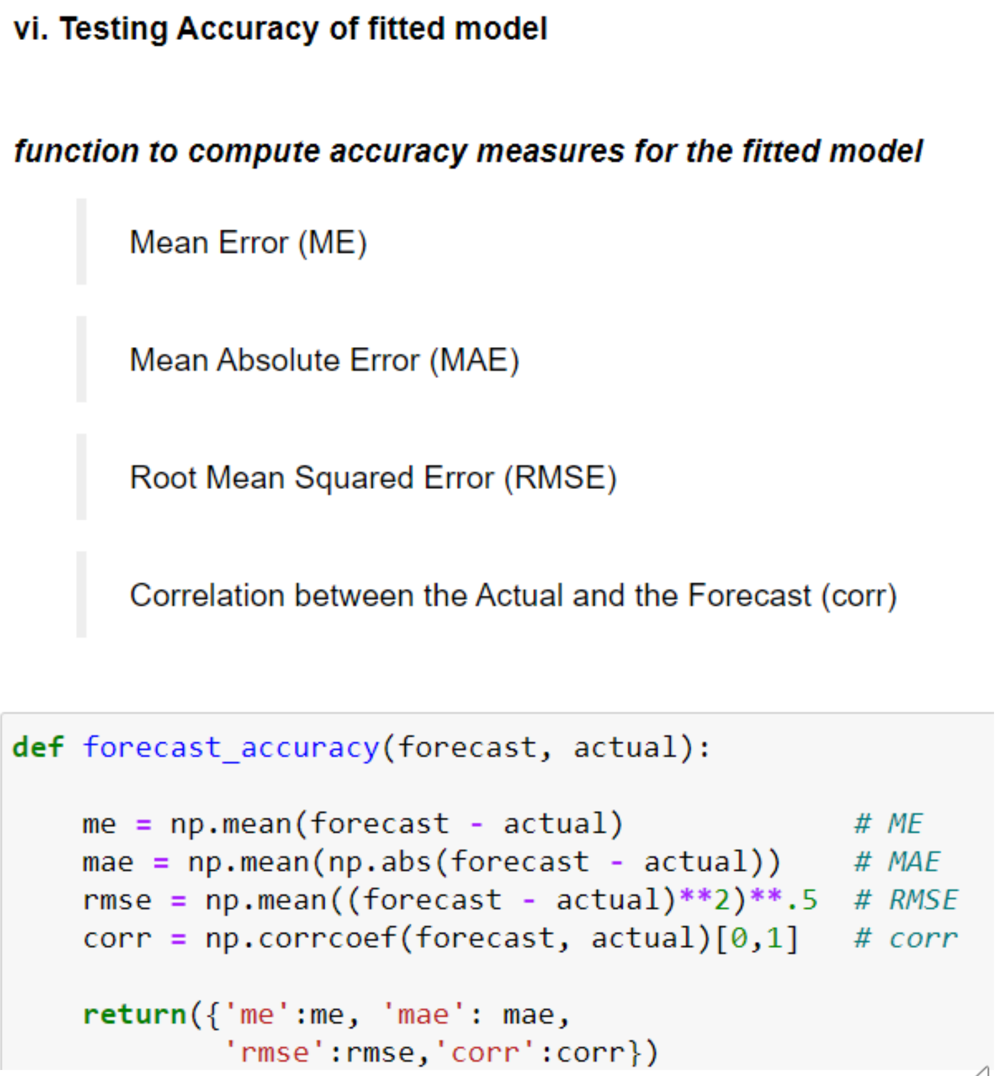
- STEP 4: FINDING THE BEST FIT PARAMETERS FOR THE ARIMA MODEL



- STEP 5: FITTING THE ARIMA MODEL



- STEP 6: ACCURACY



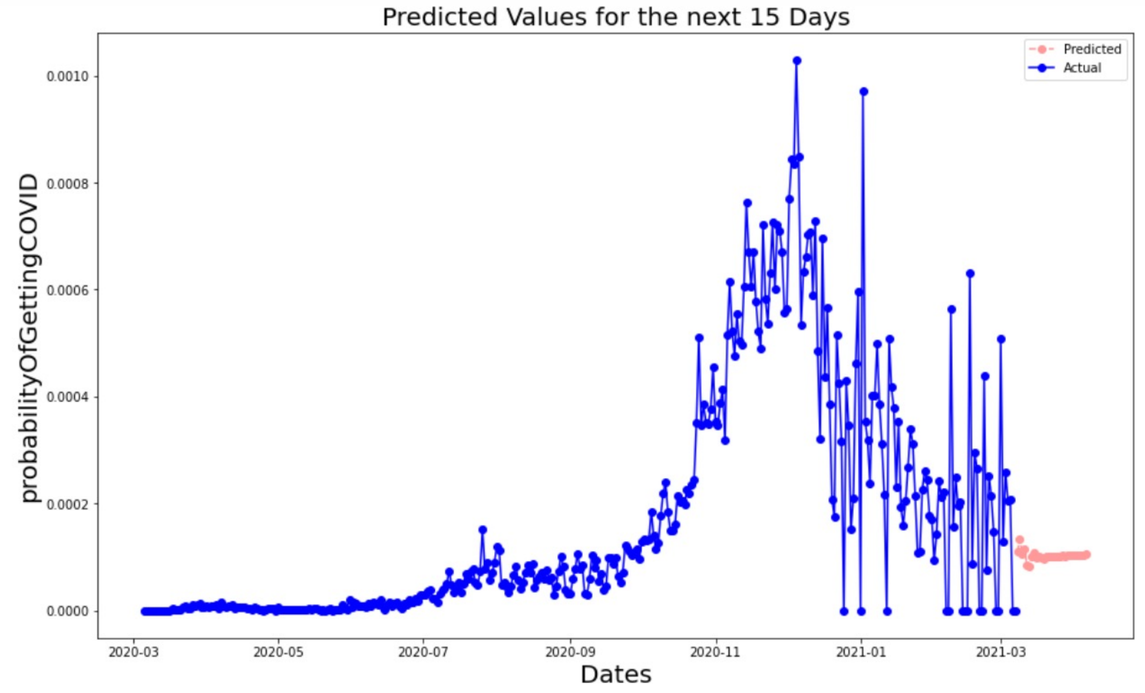
vi. Testing Accuracy of fitted model

function to compute accuracy measures for the fitted model

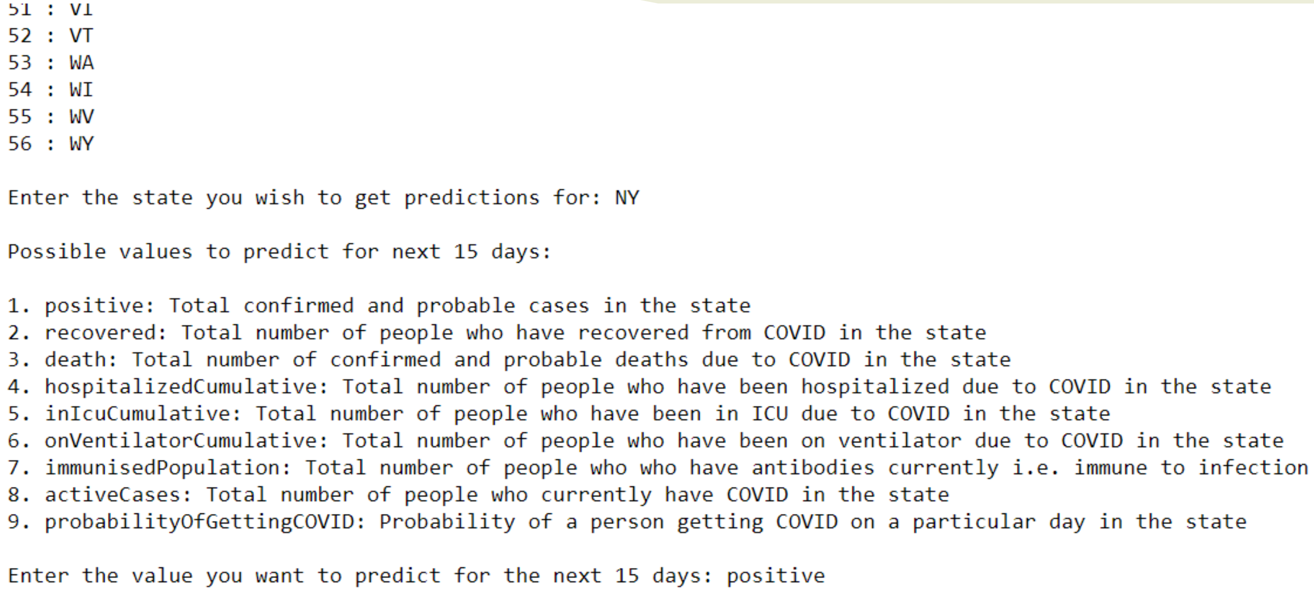
> Mean Error (ME)
>
> Mean Absolute Error (MAE)
>
> Root Mean Squared Error (RMSE)
>
> Correlation between the Actual and the Forecast (corr)

```
def forecast_accuracy(forecast, actual):

    me = np.mean(forecast - actual)             # ME
    mae = np.mean(np.abs(forecast - actual))    # MAE
    rmse = np.mean((forecast - actual)**2)**.5  # RMSE
    corr = np.corrcoef(forecast, actual)[0,1]   # corr

    return({'me':me, 'mae': mae,
            'rmse':rmse,'corr':corr})
```

- STEP 7: PREDICTING VALUES FOR NEXT 15 DAYS S

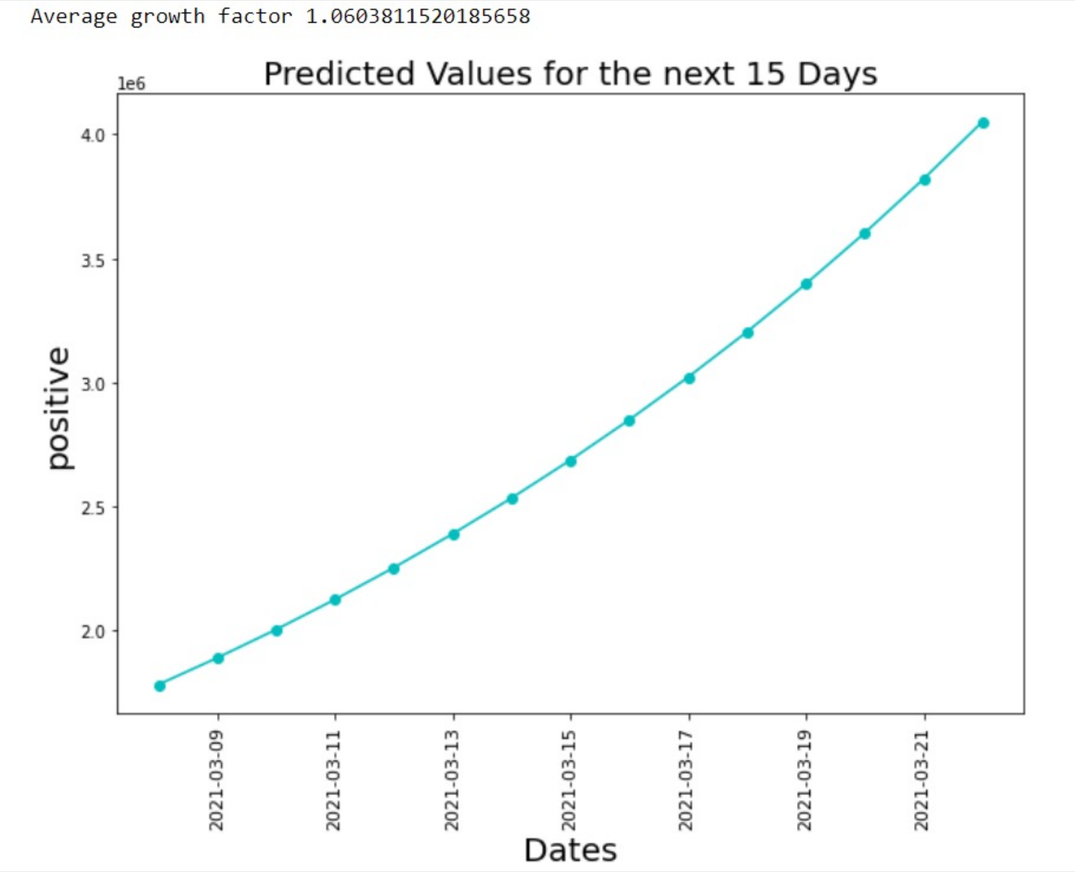


User Interface



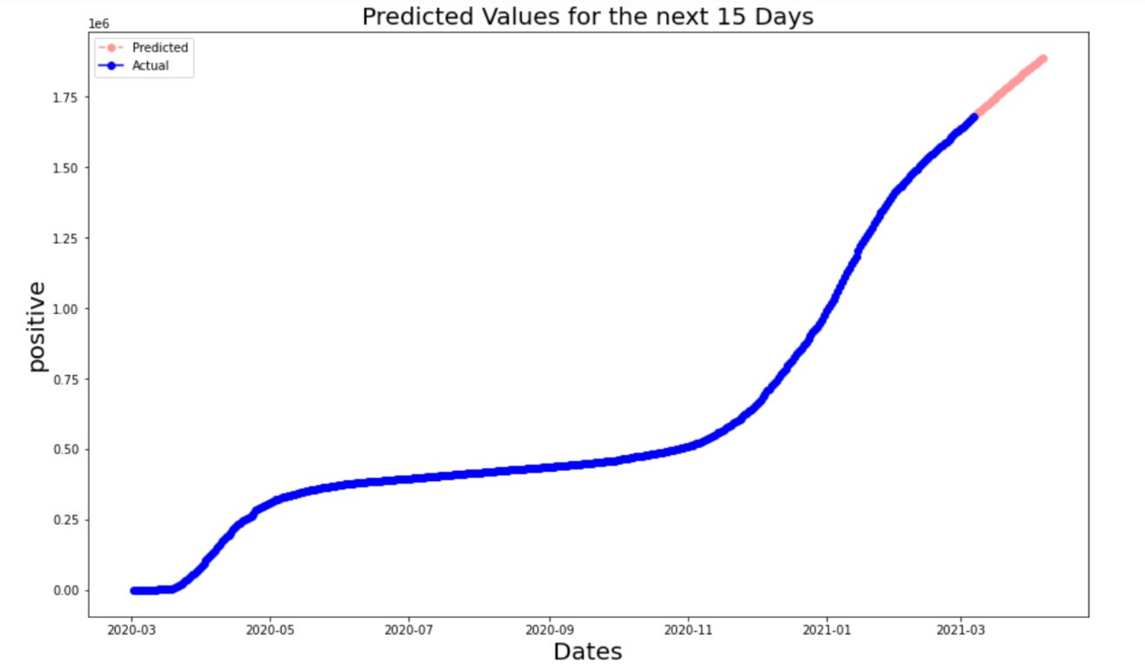
```
51 : VI
52 : VT
53 : WA
54 : WI
55 : WV
56 : WY

Enter the state you wish to get predictions for: NY

Possible values to predict for next 15 days:

1. positive: Total confirmed and probable cases in the state
2. recovered: Total number of people who have recovered from COVID in the state
3. death: Total number of confirmed and probable deaths due to COVID in the state
4. hospitalizedCumulative: Total number of people who have been hospitalized due to COVID in the state
5. inIcuCumulative: Total number of people who have been in ICU due to COVID in the state
6. onVentilatorCumulative: Total number of people who have been on ventilator due to COVID in the state
7. immunisedPopulation: Total number of people who who have antibodies currently i.e. immune to infection
8. activeCases: Total number of people who currently have COVID in the state
9. probabilityOfGettingCOVID: Probability of a person getting COVID on a particular day in the state

Enter the value you want to predict for the next 15 days: positive
```

Result 1:

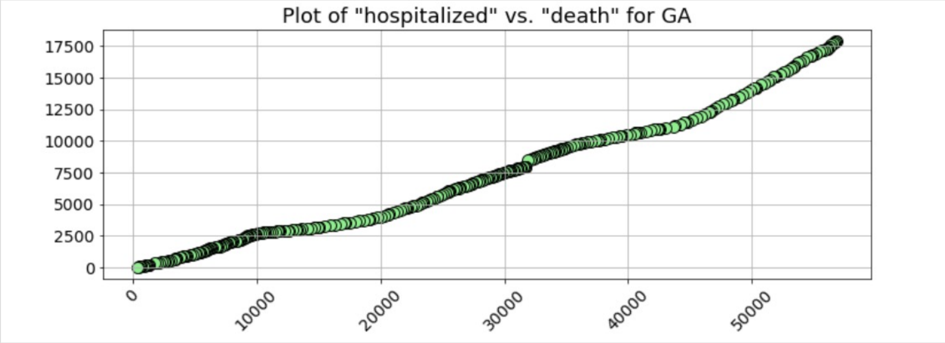


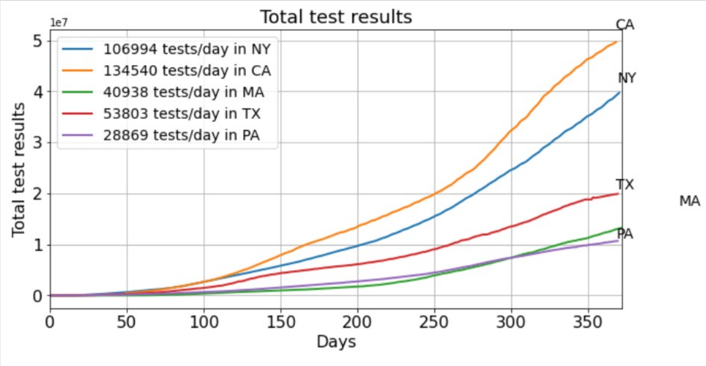
```
Average growth factor 1.0603811520185658
```

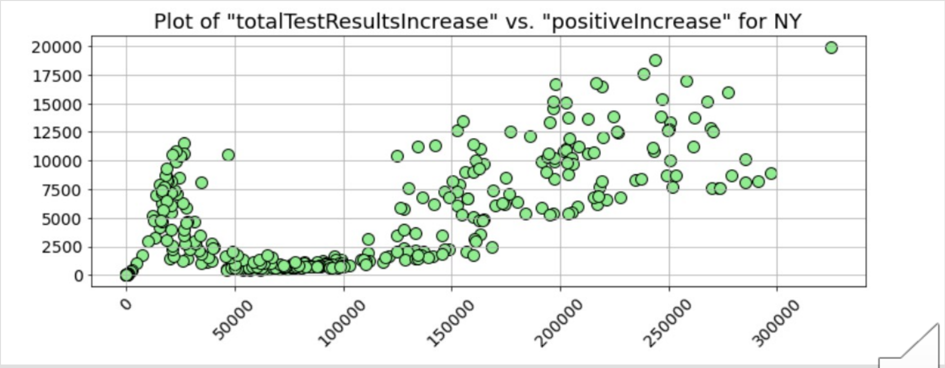
Result 2:


Predicted Values for the next 15 Days
1e6
Predicted
Actual
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
positive
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01
2021-03
Dates

Data driven insights







Conclusion

Citation

https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/

https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/