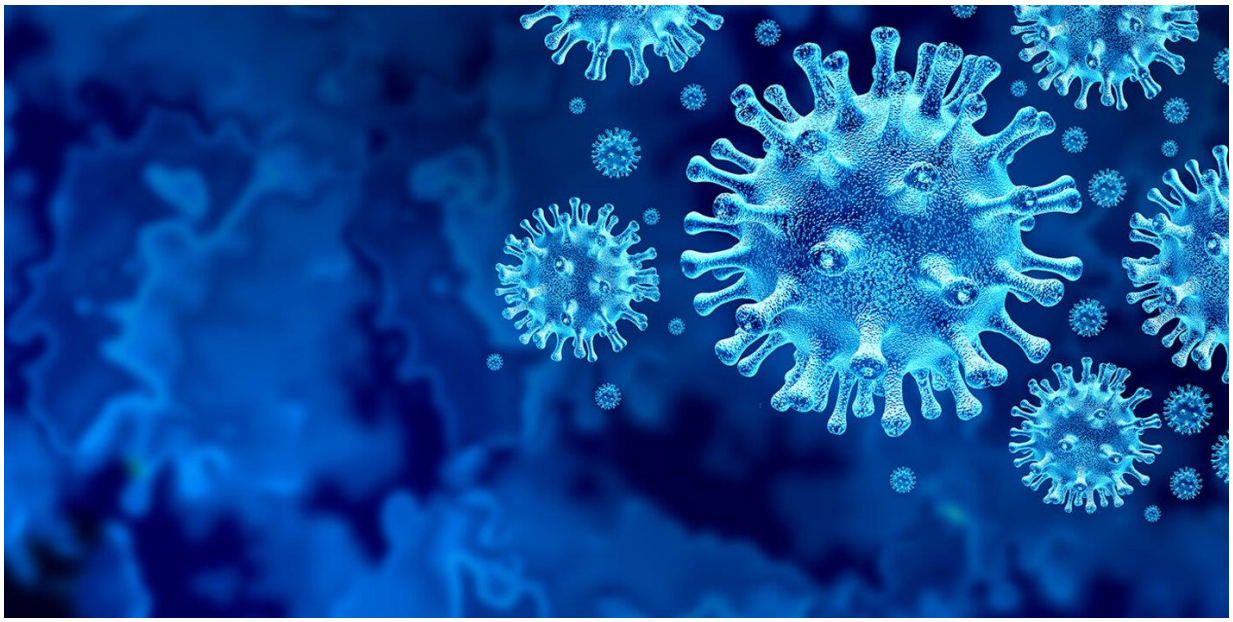
Project Report

Capstone Project

Impact of COVID – 19 in U.S.A



Project By

Rohan Desai – RUID 195008774

Vidhi Kamlesh Shah – RUID 196001942

Under Guidance of Dr. Sergei Schreider

Rutgers University, Newark.

Masters in Information Technology and Analytics

RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

Index

Contents

Introduction

Coronavirus disease (COVID-19) is an infectious disease caused by a newly discovered coronavirus. On February 11, 2020 the World Health Organization announced an official name for the disease that is causing the 2019 novel coronavirus outbreak, first identified in Wuhan China. The new name of this disease is coronavirus disease 2019, abbreviated as COVID-19.

Most people infected with the COVID-19 virus will experience mild to moderate respiratory illness and recover without requiring special treatment. Older people, and those with underlying medical problems like cardiovascular disease, diabetes, chronic respiratory disease, and cancer are more likely to develop serious illness. The best way to prevent and slow down transmission is to be well informed about the COVID-19 virus, the disease it causes and how it spreads. Protect yourself and others from infection by washing your hands or using an alcohol based rub frequently and not touching your face.

The COVID-19 virus spreads primarily through droplets of saliva or discharge from the nose when an infected person coughs or sneezes, so it's important that you also practice respiratory etiquette (for example, by coughing into a flexed elbow).

There have been total 64.1 Million cases globally out of which 41.2 Million have been recovered and 1.48 Million Deaths. There have been severe and serious lockdowns in all nations of the world to avoid COVID - 19 for more than 9 months, However, the disease has been spreading everywhere along with Unites states of America.

In this project, we are conducting an analysis of COVID 19 cases in USA to study the impact on population and perform a prediction of Death cases to avoid further losses of lives using advanced Data Science models.

Objective of Study

Objective of this study is to provide a visualization of impact of COVID 19 disease in United States of America. By this we can have a live view of the situation in America (Death Cases, Recovered Cases and Confirmed cases).

This Analysis is important and can help to identify the counties and States where the COVID disease is spreading widely and those states and counties where it is recovering vastly.

The main objective of the Project is to identify the key areas where cases are exponentially high so that population could take appropriate measured to avoid the spread of disease.

Research Methodologies

In this Project we will be using Python Programming language to Extract Data and Store it in MongoDB Database on Cloud. Further we will use this data to Provide Data Analysis and Visualization along with applying various prediction models to predict the Death cases in Python.

We will also harness the power of MongoDB Atlas which is a cloud service to extract visualizations directly from the DB.

This is a live Project where we have Extract Data from John Hopkin's University's Dataset which is updated daily. We have Web scrapped their GitHub Data Repository and stored the data into MongoDB.

Further we have pulled the Data back in Python to perform prediction.

Data Source

Johns Hopkins University is tracking cases of COVID – 19 in USA. They update their website daily with new and updated number's or infected cases and recovered cases to various researchers and Scientist so that this data can be used for various complex resources.

We Pull Data from their Daily updated repository on GitHub. This Data can be found in RAW format here:

https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data

This Data has Confirmed Cases, Recovered Cases and Death Cases by State – Counties and Latitude and Longitude. A complete Metadata and Descriptions of fields is described in the link above.

Data Cleansing and Data Analysis

Following is the Data in Raw format in Repository of JHU.

raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_US.csv


```
UID,iso2,iso3,code3,FIPS,Admin2,Province_State,Country_Region,Lat,Long_,Combined_Key,Population,1/22/20,1/23/20,1/24/20,1/25/20,1/26/20,1/27/20,1/28/20,1/29/20,1/30/20,1/31/20,2/1/20,2/2
/20,2/3/20,2/4/20,2/5/20,2/6/20,2/7/20,2/8/20,2/9/20,2/10/20,2/11/20,2/12/20,2/13/20,2/14/20,2/15/20,2/16/20,2/17/20,2/18/20,2/19/20,2/20/20,2/21/20,2/22/20,2/23/20,2/24/20,2/25/20,2/26/
20,2/27/20,2/28/20,2/29/20,3/1/20,3/2/20,3/3/20,3/4/20,3/5/20,3/6/20,3/7/20,3/8/20,3/9/20,3/10/20,3/11/20,3/12/20,3/13/20,3/14/20,3/15/20,3/16/20,3/17/20,3/18/20,3/19/20,3/20/20,3/21/20,
3/22/20,3/23/20,3/24/20,3/25/20,3/26/20,3/27/20,3/28/20,3/29/20,3/30/20,3/31/20,4/1/20,4/2/20,4/3/20,4/4/20,4/5/20,4/6/20,4/7/20,4/8/20,4/9/20,4/10/20,4/11/20,4/12/20,4/13/20,4/14/20,4/1
5/20,4/16/20,4/17/20,4/18/20,4/19/20,4/20/20,4/21/20,4/22/20,4/23/20,4/24/20,4/25/20,4/26/20,4/27/20,4/28/20,4/29/20,4/30/20,5/1/20,5/2/20,5/3/20,5/4/20,5/5/20,5/6/20,5/7/20,5/8/20,5/9/2
0,5/10/20,5/11/20,5/12/20,5/13/20,5/14/20,5/15/20,5/16/20,5/17/20,5/18/20,5/19/20,5/20/20,5/21/20,5/22/20,5/23/20,5/24/20,5/25/20,5/26/20,5/27/20,5/28/20,5/29/20,5/30/20,5/31/20,6/1/20,6
/2/20,6/3/20,6/4/20,6/5/20,6/6/20,6/7/20,6/8/20,6/9/20,6/10/20,6/11/20,6/12/20,6/13/20,6/14/20,6/15/20,6/16/20,6/17/20,6/18/20,6/19/20,6/20/20,6/21/20,6/22/20,6/23/20,6/24/20,6/25/20,6/2
6/20,6/27/20,6/28/20,6/29/20,6/30/20,7/1/20,7/2/20,7/3/20,7/4/20,7/5/20,7/6/20,7/7/20,7/8/20,7/9/20,7/10/20,7/11/20,7/12/20,7/13/20,7/14/20,7/15/20,7/16/20,7/17/20,7/18/20,7/19/20,7/20/2
0,7/21/20,7/22/20,7/23/20,7/24/20,7/25/20,7/26/20,7/27/20,7/28/20,7/29/20,7/30/20,7/31/20,8/1/20,8/2/20,8/3/20,8/4/20,8/5/20,8/6/20,8/7/20,8/8/20,8/9/20,8/10/20,8/11/20,8/12/20,8/13/20,8
/14/20,8/15/20,8/16/20,8/17/20,8/18/20,8/19/20,8/20/20,8/21/20,8/22/20,8/23/20,8/24/20,8/25/20,8/26/20,8/27/20,8/28/20,8/29/20,8/30/20,8/31/20,9/1/20,9/2/20,9/3/20,9/4/20,9/5/20,9/6/20,9
/7/20,9/8/20,9/9/20,9/10/20,9/11/20,9/12/20,9/13/20,9/14/20,9/15/20,9/16/20,9/17/20,9/18/20,9/19/20,9/20/20,9/21/20,9/22/20,9/23/20,9/24/20,9/25/20,9/26/20,9/27/20,9/28/20,9/29/20,9/30/2
0,10/1/20,10/2/20,10/3/20,10/4/20,10/5/20,10/6/20,10/7/20,10/8/20,10/9/20,10/10/20,10/11/20,10/12/20,10/13/20,10/14/20,10/15/20,10/16/20,10/17/20,10/18/20,10/19/20,10/20/20,10/21/20,10/2
2/20,10/23/20,10/24/20,10/25/20,10/26/20,10/27/20,10/28/20,10/29/20,10/30/20,10/31/20,11/1/20,11/2/20,11/3/20,11/4/20,11/5/20,11/6/20,11/7/20,11/8/20,11/9/20,11/10/20,11/11/20,11/12/20,1
1/13/20,11/14/20,11/15/20,11/16/20,11/17/20,11/18/20,11/19/20,11/20/20,11/21/20,11/22/20,11/23/20,11/24/20,11/25/20,11/26/20,11/27/20,11/28/20,11/29/20,11/30/20,12/1/20
84001001,US,USA,840,1001.0,Autauga,Alabama,US,32.53952745,-86.64408227,"Autauga, Alabama,
US",55869,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,2,2,
2,1,1,2,2,2,2,2,3,4,4,4,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,3,3,3,3,3,3,3,3,3,3,4,4,5,5,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,6,7,7,8,8,8,8,10,10,10,11,11,11,11,11,12,12,12,12,12,12,12,13,14,14,
15,15,17,18,19,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,21,21,21,21,21,21,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22,23,23,23,23,23,23,23,23,23,23,23,
23,23,23,23,24,24,24,24,24,24,25,25,25,25,25,25,25,27,27,28,27,27,27,27,27,27,28,28,28,28,28,28,28,28,28,28,28,28,29,30,30,30,31,31,31,31,31,31,31,31,31,31,31,31,31,30,30,30,30,30,32,32,
36,36,36,36,37,37,39,39,39,39,39,39,41,42,42,42,42,42,42
84001003,US,USA,840,1003.0,Baldwin,Alabama,US,30.72774991,-87.72207058,"Baldwin, Alabama,
```


Here Every Day's data is added to the file in new column with date as column name. It was important and necessary to clean and process this Data in order to insert it into MongoDB. Also it was important to create a connection between mongo DB and Python to achieve this task.

<u>Code for connecting with MongoDB:</u>

```
mongo_url =
'mongodb+srv://my_data:test@cluster0.9q8b8.mongodb.net/mydata?retryWrites=true&w=majority'

from pymongo import MongoClient, collection

def get_mongo_collection(url, collection_name):

    client = MongoClient(url)

    db = client.mydata

    return db[collection_name]
```

Above Snippet has a link which is used for connecting python with Cluster of MongoDB.

Then we have defined a function to establish the connection.

Clean the Data:

Data Cleaning is an important step in data analysis and it takes about 50-60% of the time in any analysis to clean the data. The challenges we faced in cleaning data were to convert the date columns into rows to corresponding counties and States. This was achieved by a function we created which iterated objects to convert them into rows from columns.

Further we identified missing values and removed them. Since imputation was not meaningful in this analysis. Rows with null values were dropped.

Following code is for column to Row transposition of the Data records.

```
def get_data_in_url(url, pattern):
    data_frame = pd.read_csv(url)
    data_objects = []
    re.compile(pattern)
    pattern_matcher = re.compile(pattern)
    dates = []
    for col in data_frame.columns:
        if pattern_matcher.match(str(col)) is not None:
            dates.append(str(col))

    for index, row in data_frame.iterrows():
        county = row['Admin2']
        state = row['Province_State']
        region = row['Country_Region']
        latitude = row['Lat']
        longit = row['Long_']
        for date in dates:
            documentId = str(county) + "_" + str(state) + "_" + str(region) + "_" + str(date)
            value = row[date]
            confirmed = DataFormat(documentId, county, state, region, latitude, longit,
date, value)
            data_objects.append(vars(confirmed))
    return data_objects
```

Get Data function pulls Data from source using Link of repository and then it is cleaned in the function and inserted into Mongo dB Database.

Extract Live Data everyday:

We extract Data everyday on Day -1 basis. Code for this is as follows:

```
def pass_date_param():

    today_date = datetime.date.today()

    yesterday_date = today_date -
datetime.timedelta(days=1)

    formatted_date =
datetime.date.strftime(yesterday_date,
"X%m/X%d/%y").replace('X0', 'X').replace('X', '')

    return str(formatted_date)
```

Code for Pulling Data is Dynamic. We pull all the data, then we check for date which is Day -1 and then we iterate over only that date's data and then insert this into Database.

By using above function we determine yesterday's date and then we insert data specific to only that day. We perform this daily to keep the analysis Real time and Live.

Data Visualization

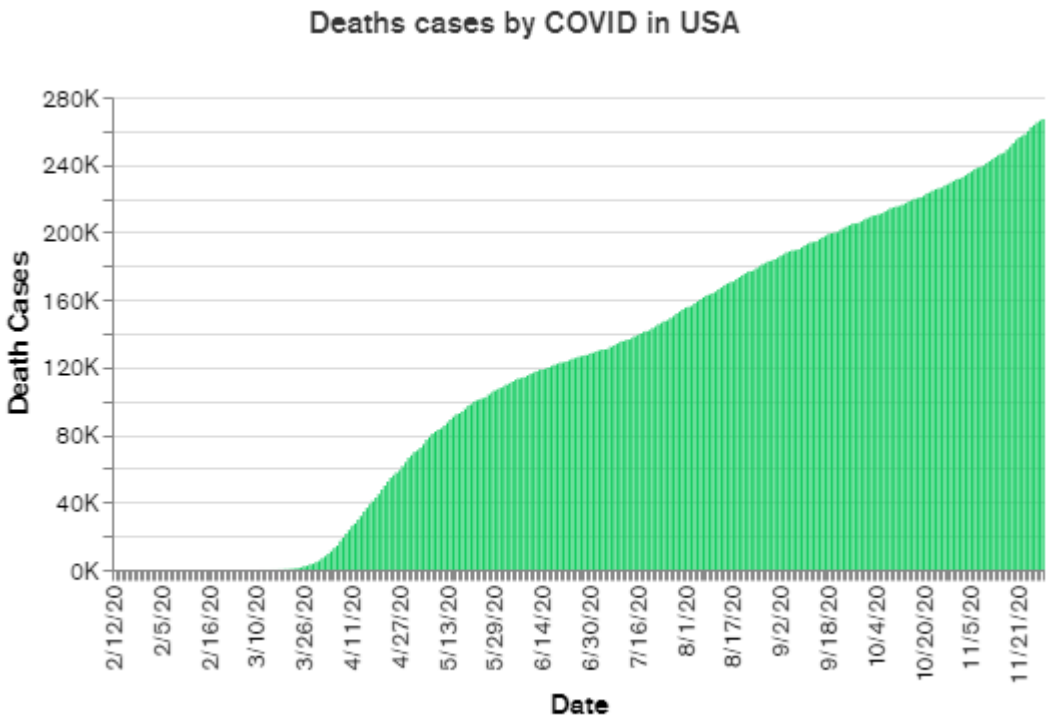
Data Visualization is graphical interpretation of the data which makes it easy to understand the Data and identify some key trends in Data.

There are various Data Visualization methods and Tools available which can be used by researchers and Data scientist to display information from data. Some of them are Tableau, IBM Cognos etc.

When it comes to Data science, generally Python is used for this purpose as Python has a huge repository of libraries for visualization which can be used to represent information in graphical format.

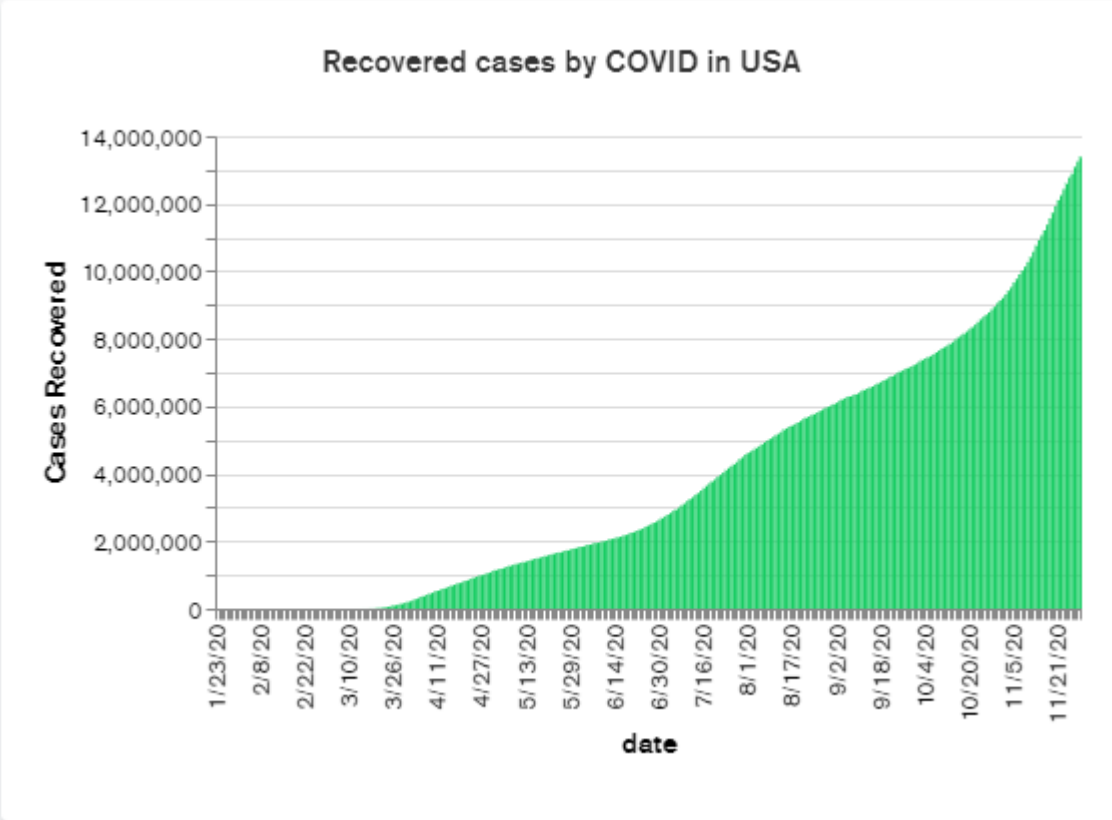
In this Project we have used Atlas service of MongoDB database. Atlas is a cloud based service which represents data in forms of charts and graph in Real time.

1) **Death Cases in U.S.A by COVID 19**



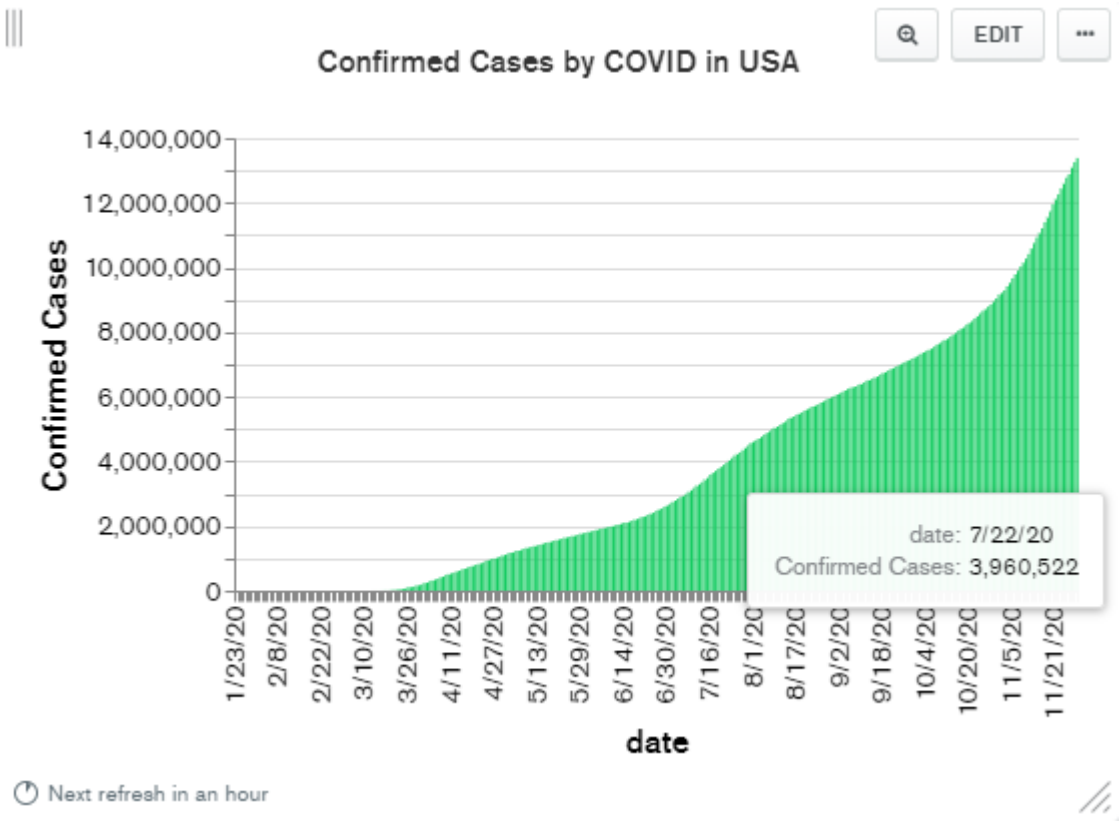
Above graph represents Deaths cases toll in USA. It takes live data from mongDB database and performs this task. Refresh rate for this is 4 hours daily. We can see that Death toll took a rise since March and it has been increasing since then.

2) Recovered Cases in U.S.A by COVID 19

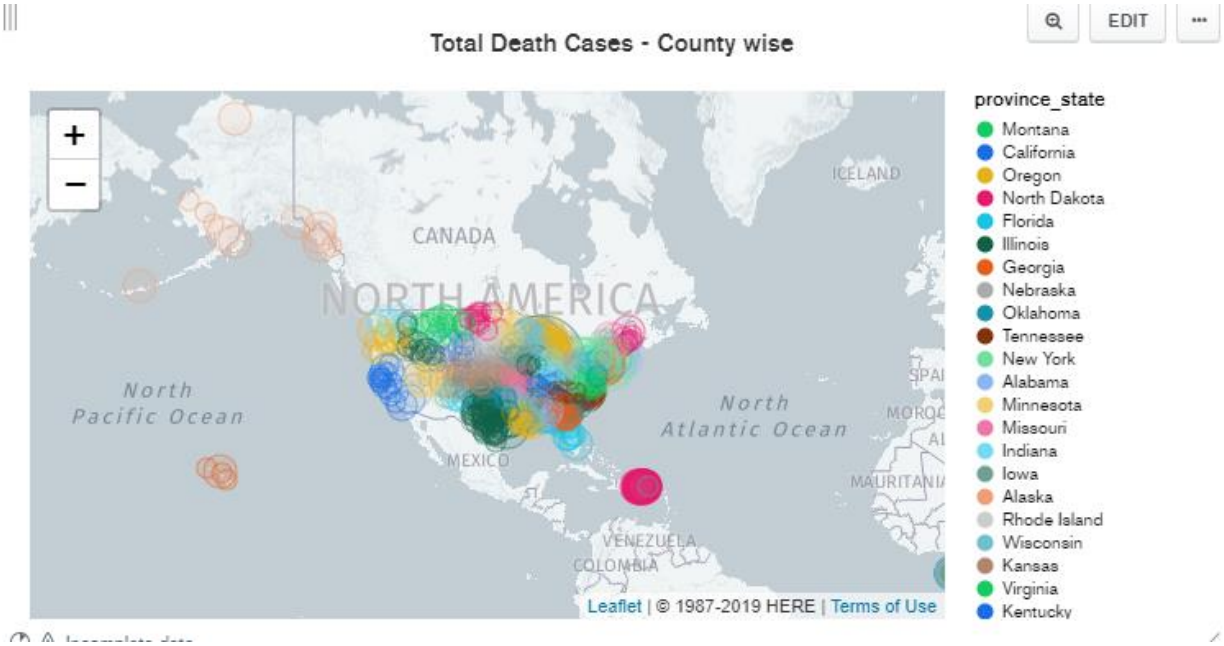


Above graph represents recovered cases toll in USA. Despite deaths we have significant number of recoveries in USA.

3) Confirmed Cases in U.S.A by COVID 19

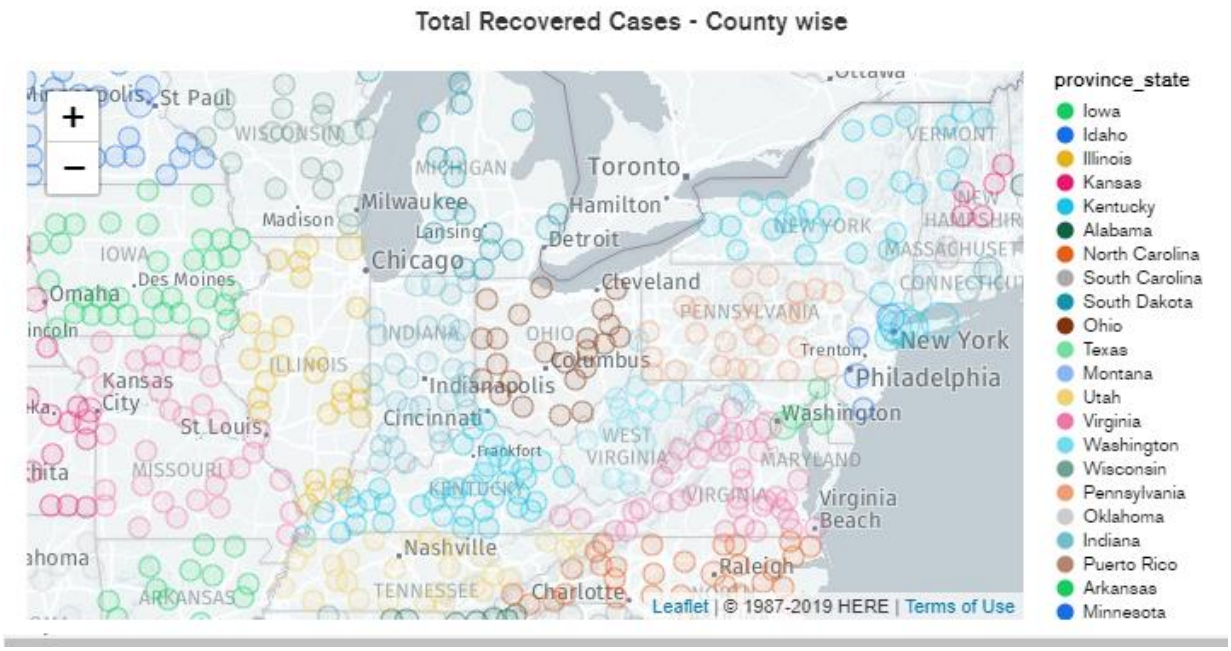


4) County wise Deaths in USA by COVID 19

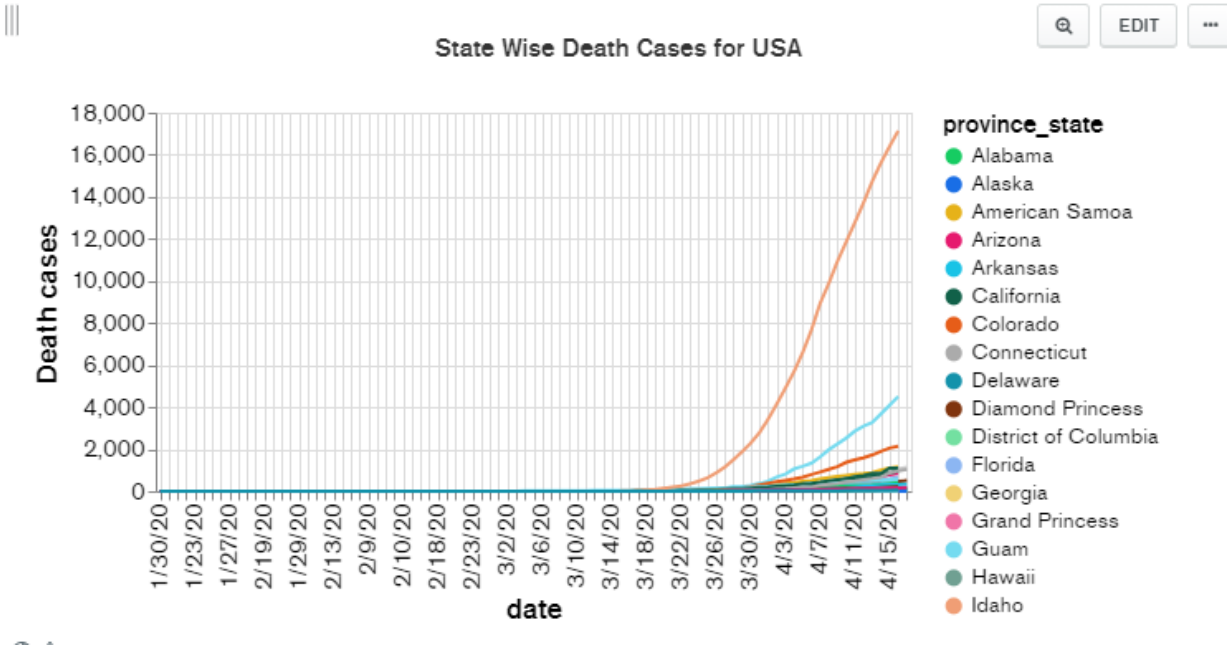


Our Data has latitude and longitude which helps us to plot geospatial coordinates on map of the cases. Here we have plotted county wise cases.

5) County wise Recovered in USA by COVID 19

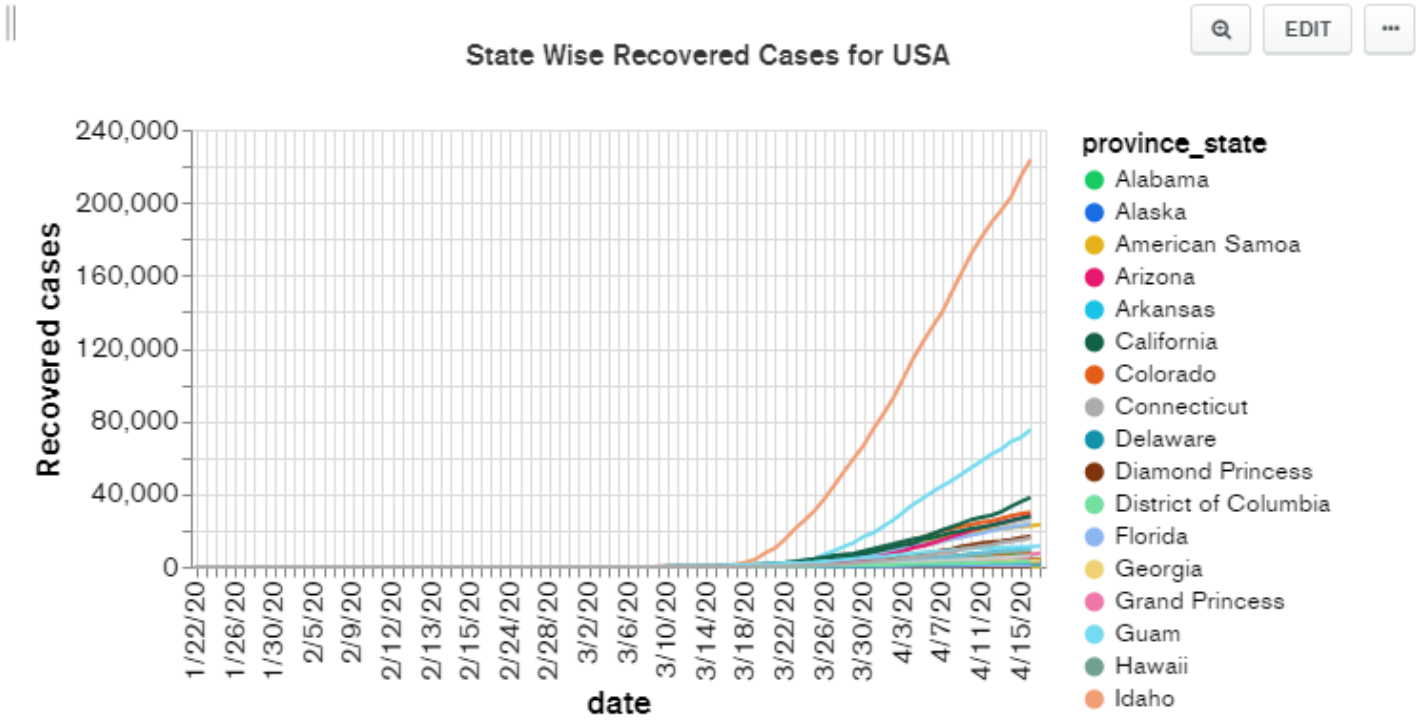


6) State wise Death in USA by COVID 19



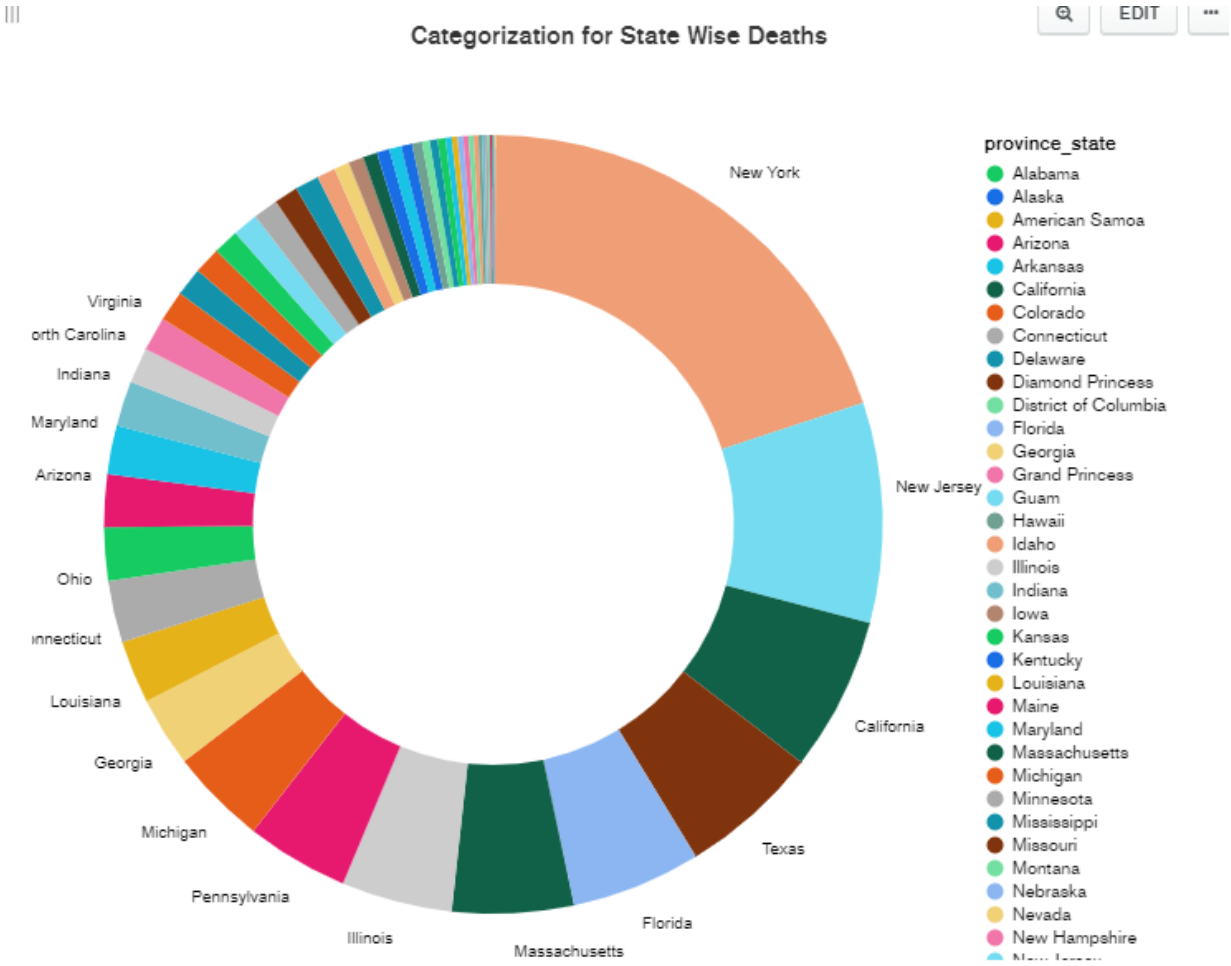
This figure gives us a state wise death toll which by which we can identify which state has what trend of deaths by COVID 19

7) State wise Recovery in USA by COVID 19



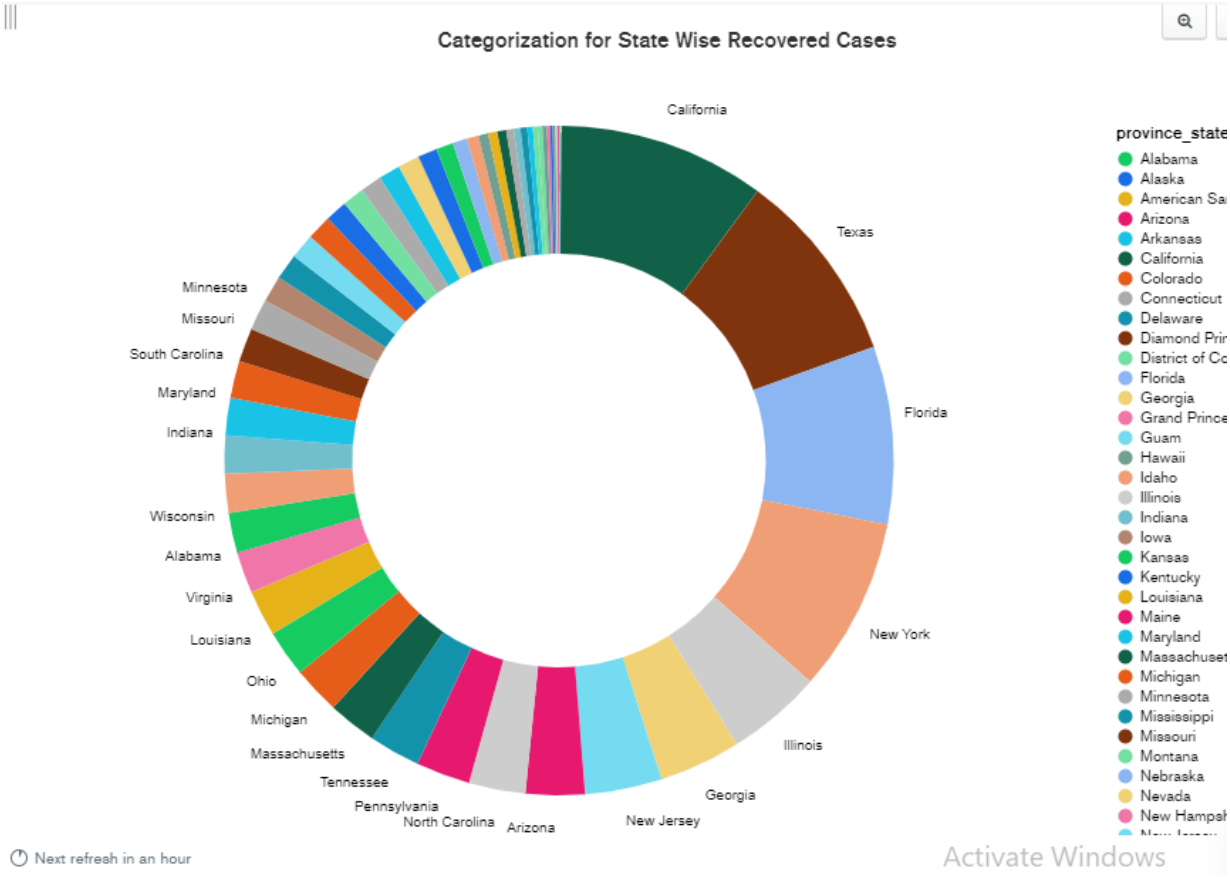
This figure gives us a state wise recovery toll which by which we can identify which state has what trend of deaths by COVID 19

8) State wise deaths using pie chart in USA by COVID 19



Above pie chart indicates New York State has maximum number of deaths state wise.

9) State wise deaths using pie chart in USA by COVID 19



Above Pie chart is for State wise recovered cases of covid 19. We can see that California has maximum number of recovered cases.

Prediction

Prediction is an important and complex process in Data Science. Here we will be predicting Death cases in USA by applying 2 models on our time series. Prediction or estimate, is an assertion about a future occasion. They are frequently, however not generally, founded on experience or information.

In our Project, the Data which we have cleaned and placed in DB, we will extract it in Python and use Forecasting to predict the model and its accuracy.

While moving to Prediction there are certain tests which are to be performed on Data in order to make data eligible for prediction.

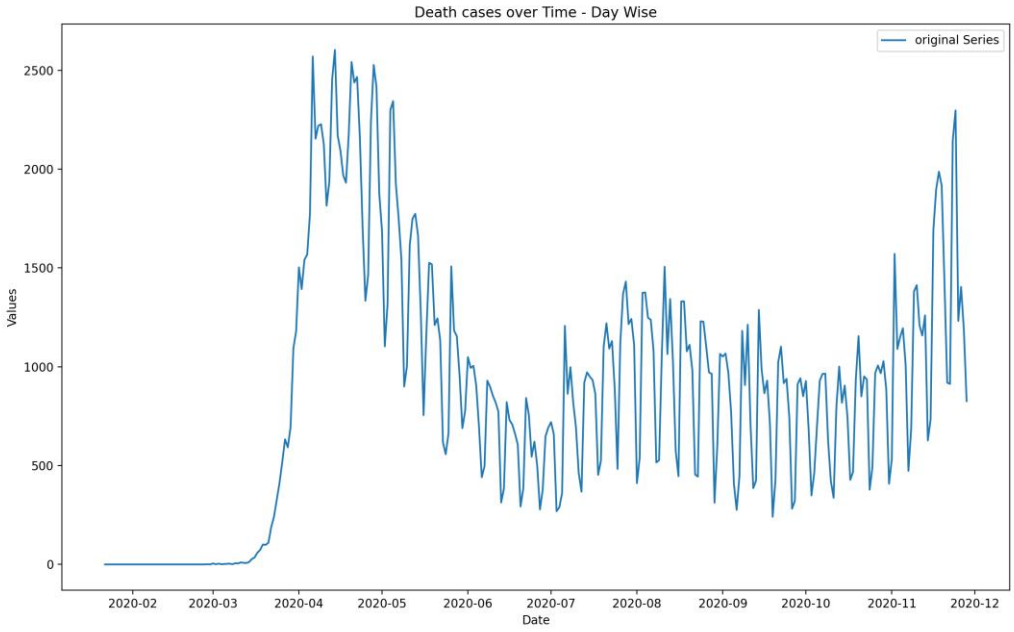
1) **Get data from MongoDb collection to Python dataframe**

   First we will have to pull the data from MongoDb in Python. We have used following code to perform this task.

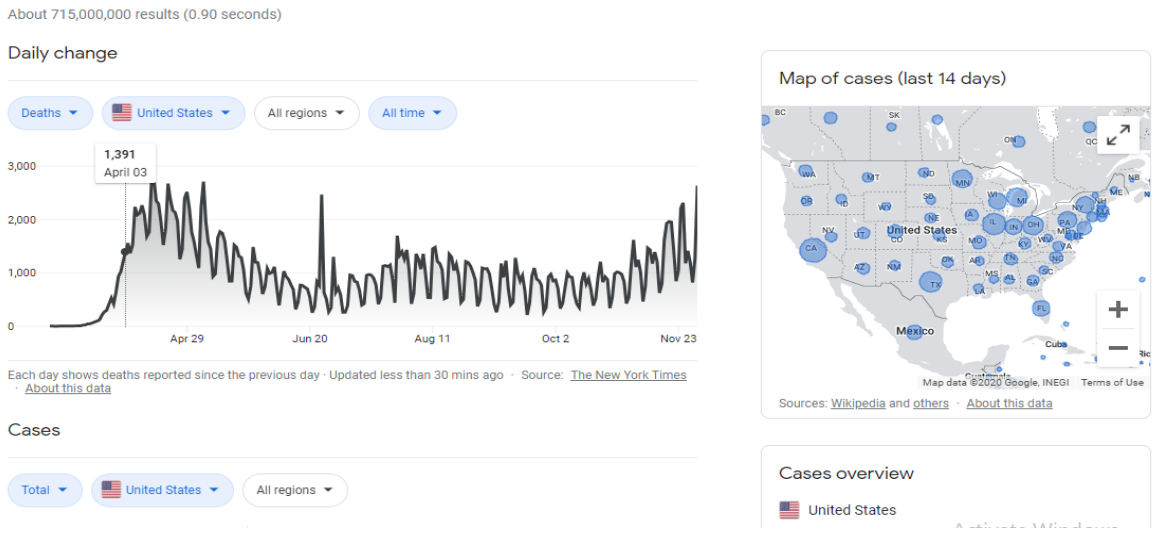
```
def data_prep_death_cases(mongo_url):
    client = MongoClient(mongo_url)
    db = client.mydata
    specified_collection = db["deaths_cases"]  # collection name for data prep
    mydata = specified_collection.find({}, {"date": 1, "value": 1})
    df_deathcases = pd.DataFrame.from_records(list(mydata))  # convert dict to list df
    df_deathcases['date'] = pd.to_datetime(df_deathcases['date'], format='%m/%d/%y')  #
format date
    groupby_data_df = df_deathcases.groupby('date',
as_index=True)['value'].sum().reset_index().sort_values(by=['date'],
                                                        ascending=True)
    groupby_data_df['value_diff'] = groupby_data_df['value'] - groupby_data_df['value'].shift(-1)
    groupby_data_df['value_diff'] = groupby_data_df['value_diff'].abs()  # fill null with 0 and
take absolute value
    groupby_data_df = groupby_data_df.drop(columns=['value'])
    groupby_data_df = groupby_data_df.dropna()
    print(groupby_data_df.tail(30))
    # test_mpl(groupby_data_df)
    return groupby_data_df
```

2) Plot the Data

We now Plot the Data after performing basic cleaning on the data. We pull Data from Death cases collection and we plot it over time. Following is the Series we get.

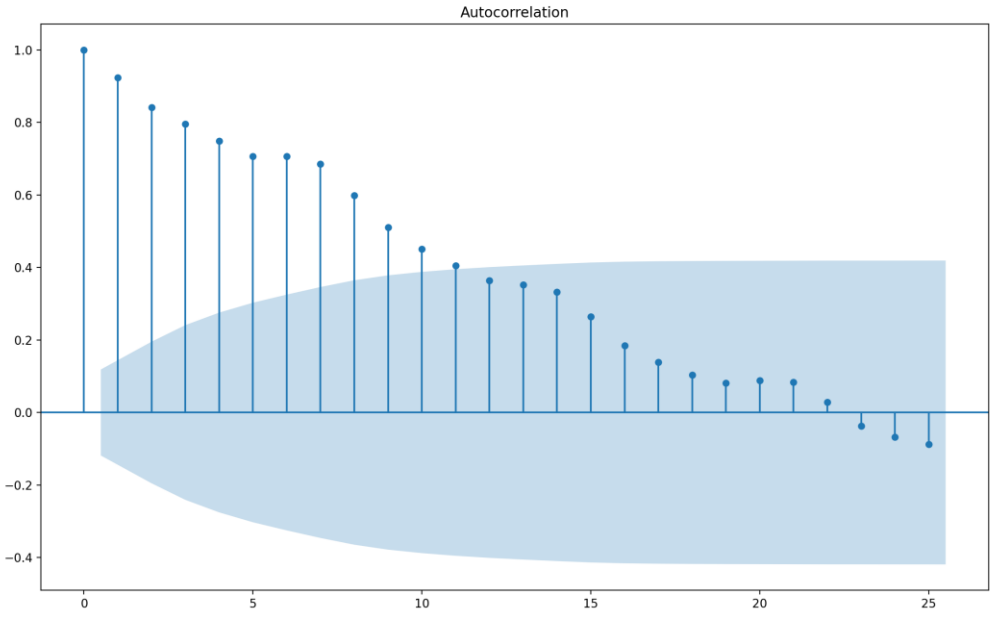


Following is the Plot from google, which resembles our Data Plot.



3) Check stationarity of Data

Above Plot clearly shows us that the data is not stationary. We perform Dicker Fuller test to prove this and plot an ACF to know more about stationarity of Data.

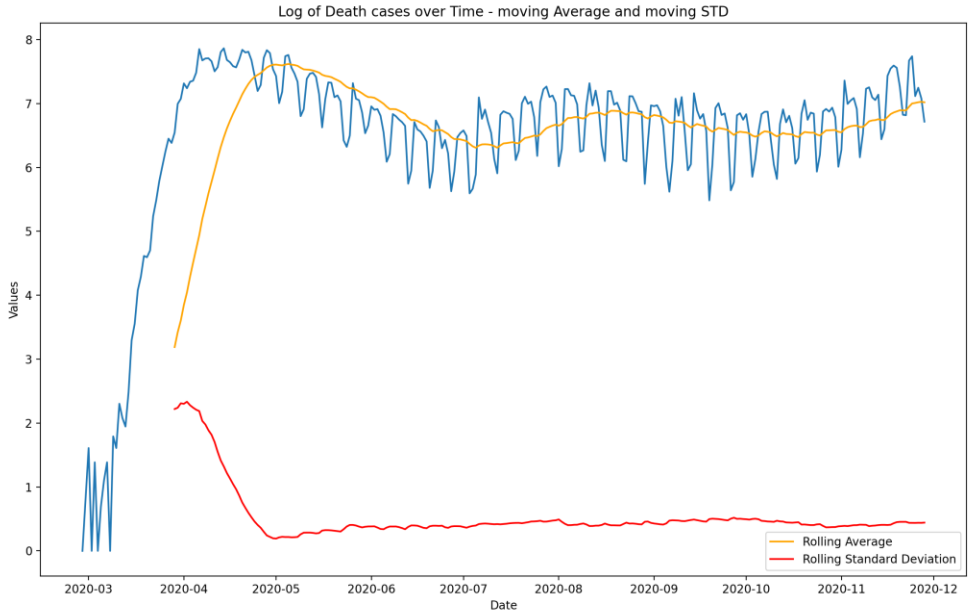


Above Plot represents that the data is not stationary. Hence to make it stationary we take Log of Data.

4) Make time series Stationary

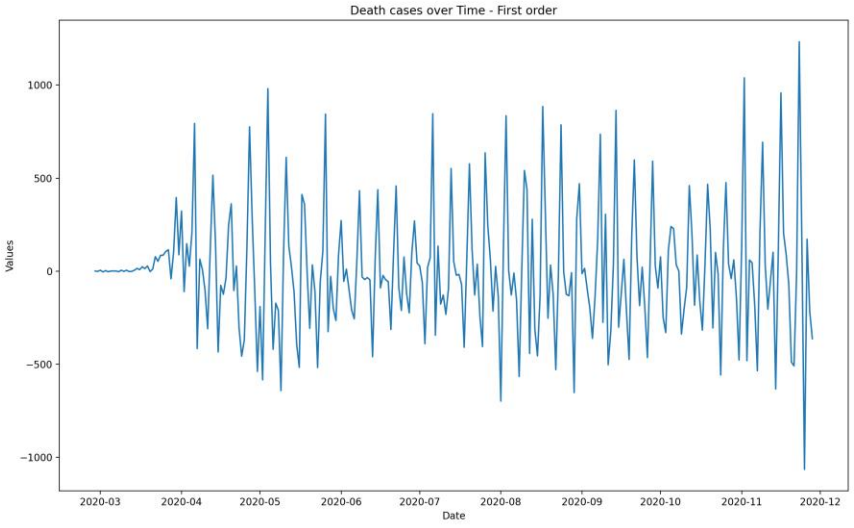
To make it stationary, we first make Log transformation of the data and then plot Rolling average and Standard deviation.

A. Log of time series



We can see that data is still not stationary from plot. Further we take 1st order differencing of the data and then try to plot it.

B. 1st Order Differencing of time series



After performing first order differencing of Data we perform Dickey fuller test to check if data is stationary or not and we observe the following:

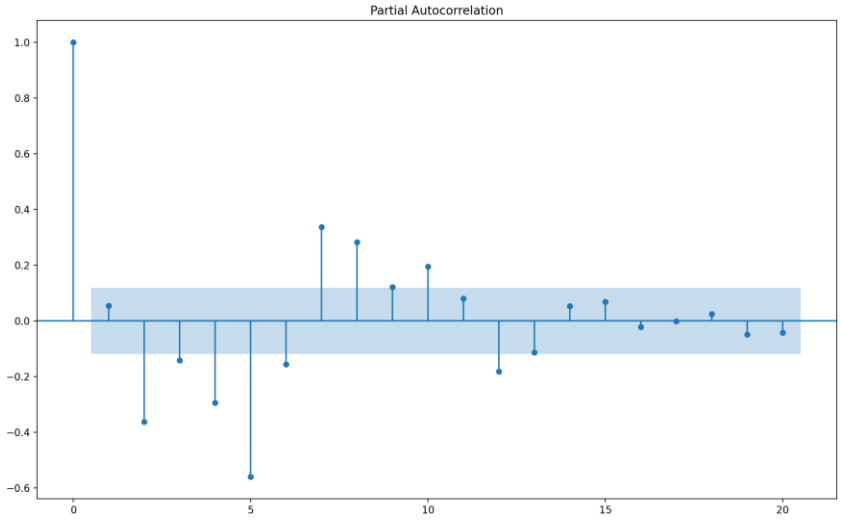
```
Dickey Fuller Test for Log Data
ADF Statistic: -5.424395
p-value: 0.000003
Critical Values:
    1%: -3.456
    5%: -2.873
    10%: -2.573
```

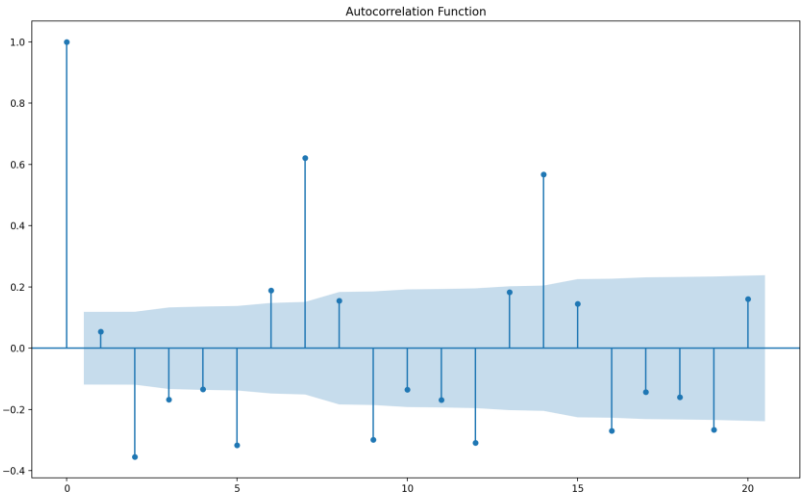
p - Value <= 0.05: Reject the null hypothesis (H0), the data does not have a unit root and is stationary. When the test statistic is lower than the critical value shown, you reject the null hypothesis and infer that the time series is stationary.

From above Results we can say that our time series is stationary now by observing p value. Also ADF statistics is less than critical values.

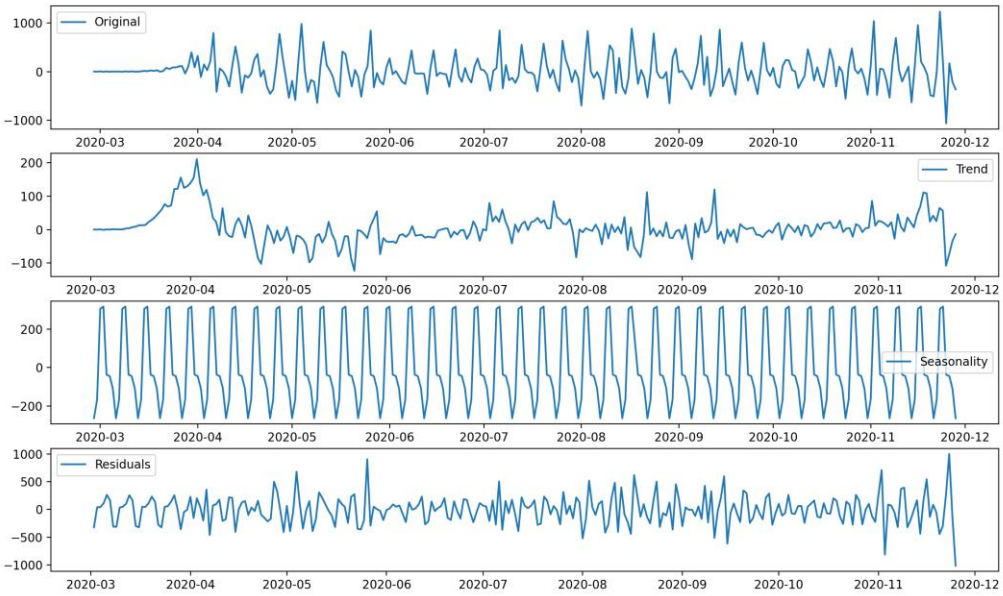
C. ACF and PACF for First order differencing

From ACF and PACF we find that there is seasonality in the Time series. Now we remove seasonality from the time series. Also Time series decomposition tells us more about the seasonality in the series.


Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
5
10
15
20


Autocorrelation Function
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
5
10
15
20

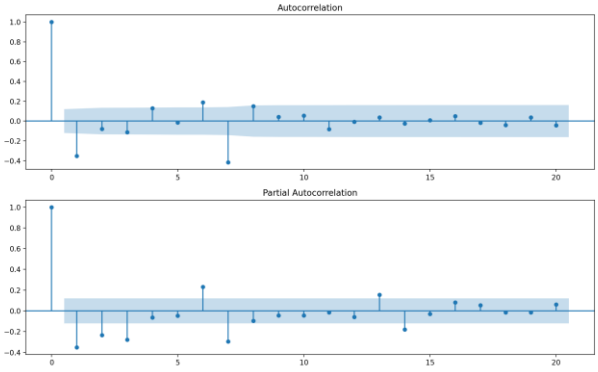
D. STL component



STL decomposition tells us about seasonality – Trend and Residual.

E. Remove seasonality

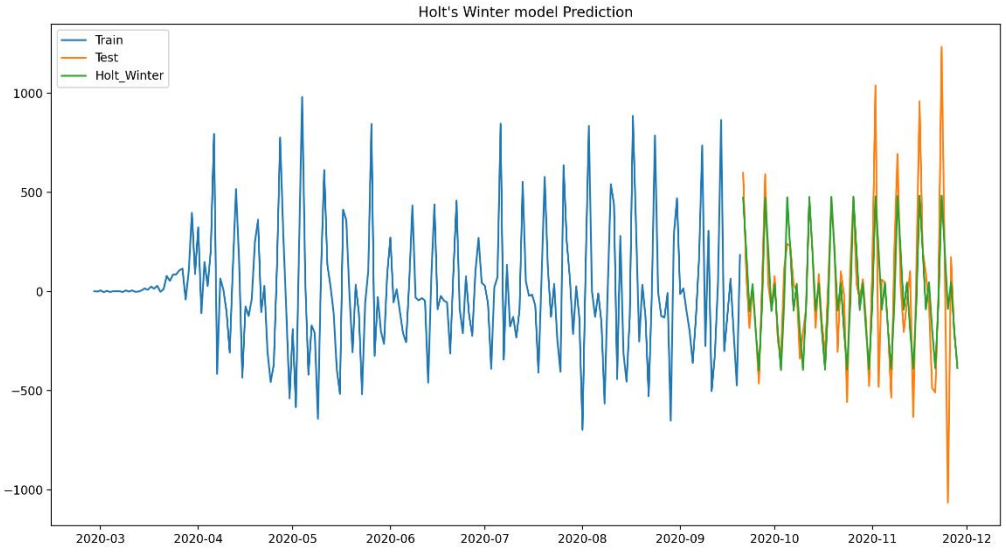
After we remove seasonality, we can is that in ACF plot, there is no seasonality. Now our time series is clean and we can use it for forecasting.



5) Perform Prediction on Time series

i) Holts Winter Model

Holt-Winters forecasting is a way to model and predict the behavior of a sequence of values over time—a time series. Holt-Winters is one of the most popular forecasting techniques for time series. Its decades old, but it's still ubiquitous in many applications, including monitoring, where it's used for purposes such as anomaly detection and capacity planning.



We can see that Holt's winter model fairly predicted the values, but it was only able to predict with 79% accuracy from MAPE value.

ii) ARIMA MODEL

ARIMA, short for 'Auto Regressive Integrated Moving Average' 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.

An ARIMA model is characterized by 3 terms: p, d, q where,

p is the order of the AR term

q is the order of the MA term

d is the number of differencing required to make the time series stationary

If a time series, has seasonal patterns, then you need to add seasonal terms and it becomes SARIMA, short for 'Seasonal ARIMA'. More on that once we finish ARIMA.

So, what does the 'order of AR term' even mean? Before we go there, let's first look at the 'd' term.

When we feed 1st order differencing data to Auto arima model ARIMA, it suggests us following:

We perform Auto Arima on our time series and get best model as ARIMA (5,1,5) based on AIC results. We then try to Fit the model and check how it performs.

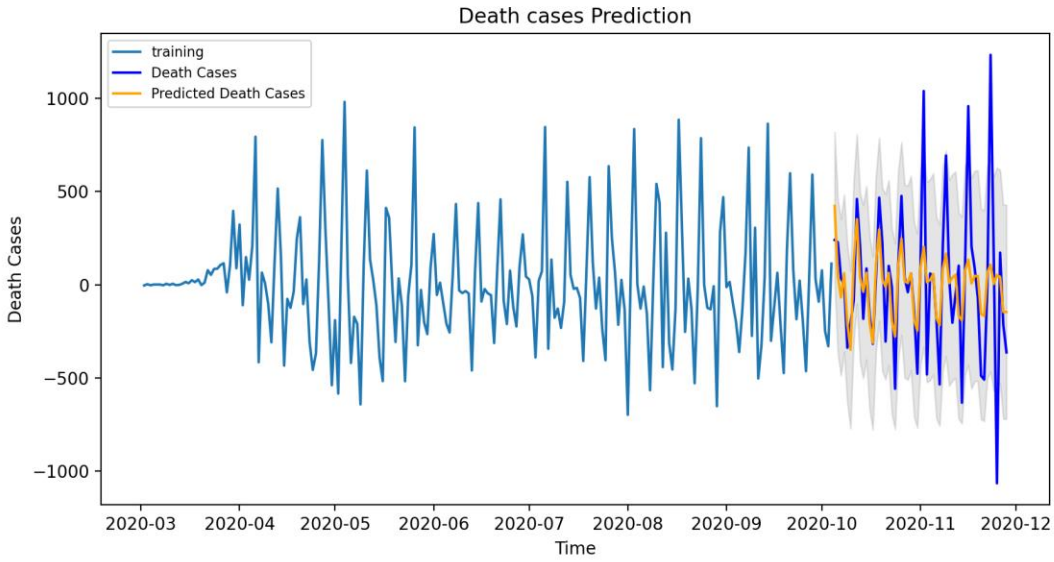
```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.15 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=4118.793, Time=0.01 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=4097.983, Time=0.02 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.07 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=4116.797, Time=0.00 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=4044.612, Time=0.03 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=4031.662, Time=0.11 sec
 ARIMA(4,1,0)(0,0,0)[0] intercept   : AIC=4031.120, Time=0.04 sec
 ARIMA(5,1,0)(0,0,0)[0] intercept   : AIC=3938.115, Time=0.08 sec
 ARIMA(5,1,1)(0,0,0)[0] intercept   : AIC=3785.103, Time=0.39 sec
 ARIMA(4,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.29 sec
 ARIMA(5,1,2)(0,0,0)[0] intercept   : AIC=3739.957, Time=0.42 sec
 ARIMA(4,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.25 sec
 ARIMA(5,1,3)(0,0,0)[0] intercept   : AIC=3733.271, Time=0.45 sec
 ARIMA(4,1,3)(0,0,0)[0] intercept   : AIC=3892.388, Time=0.48 sec
 ARIMA(5,1,4)(0,0,0)[0] intercept   : AIC=3729.554, Time=0.47 sec
 ARIMA(4,1,4)(0,0,0)[0] intercept   : AIC=3804.401, Time=0.50 sec
 ARIMA(5,1,5)(0,0,0)[0] intercept   : AIC=3713.018, Time=0.65 sec
 ARIMA(4,1,5)(0,0,0)[0] intercept   : AIC=3738.256, Time=0.53 sec
 ARIMA(5,1,5)(0,0,0)[0]             : AIC=3710.396, Time=0.56 sec
 ARIMA(4,1,5)(0,0,0)[0]             : AIC=3735.268, Time=0.47 sec
 ARIMA(5,1,4)(0,0,0)[0]             : AIC=3727.269, Time=0.40 sec
 ARIMA(4,1,4)(0,0,0)[0]             : AIC=3800.752, Time=0.42 sec

Best model:  ARIMA(5,1,5)(0,0,0)[0]
```

After fitting the model we get below prediction with accuracy as follows:

MSE: 106433.27367762715
MAE: 210.00300175446796
RMSE: 326.241128120945

Such data can be complex to predict perfectly and hence it is important that various models are applied to check how accurate the model is.


Death cases Prediction
training
Death Cases
Predicted Death Cases
1000
500
0
−500
−1000
Death Cases
2020-03
2020-04
2020-05
2020-06
2020-07
2020-08
2020-09
2020-10
2020-11
2020-12
Time

Summary

Predicting Death cases is difficult and it needs precision and trial and Error Method. We simply cannot find an optimum model and fit the data. Since from Mid-November the cases worldwide have increased, it has become difficult to use normal models. Hybrid and customs models are needed to predict accurately these results.

Further Scope

- We can develop hybrid models where we combine multiple models and can try to check for the accuracy of the models. This might me more accurate than existing model.
- We can use a cron job to automate the daily manual load. Currently we are loading data by running script daily. Cron job can be placed on Amazon AWS server.

Citations

- https://www.nature.com/articles/d41586-020-03165-9
- https://www.who.int/health-topics/coronavirus#tab=tab_1
- https://orangematter.solarwinds.com
- https://github.com/CSSEGISandData/COVID-19
- https://otexts.com/fpp2/holt-winters.html
- https://www.redcross.org/get-help/how-to-prepare-for-emergencies/types-of-emergencies/coronavirus-safety.html