COVID-19 DATA DASHBOARD

MAJOR PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

ABHISHEK JAGLAN	DAKSH TREHAN	PRIYANSH SINGHAL
Enrolment No: 00215602717	Enrolment No: 01115602717	Enrolment No: 03815602717

Guided by

Niti Yadav
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DR. AKHILESH DAS GUPTA INSTITUTE OF TECHNOLOGY & MANAGEMENT,
(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI)
NEW DELHI – 110053

JULY 2021

CANDIDATE'S DECLARATION

It is hereby certified that the work which is being presented in the B. Tech Minor Project Report entitled **"Covid-19 Data Dashboard"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** and submitted in the **Department of Computer Science & Engineering** of **Dr. Akhilesh Das Gupta Institute of Technology & Management, New Delhi (Affiliated to Guru Gobind Singh Indraprastha University, Delhi)** is an authentic record of our own work carried out during a period from **March 2021 to July 2021** under the guidance of **Niti Yadav, Assistant Professor.**

The matter presented in the B. Tech Minor Project Report has not been submitted by me for the award of any other degree of this or any other Institute.

Abhishek Jaglan	**Daksh Trehan**	**Priyansh Singhal**
00215602717	**01115602717**	**03815602717**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge. He/She/They are permitted to appear in the External Major Project Examination.

Niti Yadav
Assistant Professor

Prof. (Dr) Anupam Kumar Sharma
Head, Department of CSE

The B. Tech. Minor Project Viva-Voce Examination of **Abhishek Jaglan (Enrolment No: 00215602717) ,** has been held on ___ July, 2021.

Pinky
Project Coordinator

(Signature of External Examiner)

ABSTRACT

With rise of the highlight of the decade and information related to it in an unorganised and obscene manner, the need for a centralised platform to gather information from on an international front doesn't seem to be far fetched. The data dashboard is formed with help of data taken from reliable sources to portrait it in an interactive and easy to consume format with salient feature like a chatbot, cases prediction and projection of data in numerous formats updated daily.

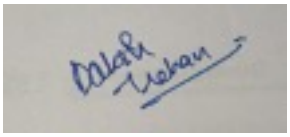
ACKNOWLEDGEMENT

We express our deep gratitude to **Nti Yadav**, Assistant Professor, Department of Computer Science & Engineering for his valuable guidance and suggestion throughout my project work. We are thankful to **Ms. Pinky Yadav,** Project Coordinator for their valuable guidance.

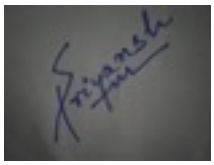
We would like to extend my sincere thanks to **Prof. (Dr) Anupam Kumar Sharma, Head of Department,** for her time-to-time suggestions to complete my project work. I am also thankful to **Prof. (Dr.) Sanjay Kumar, Director** for providing me the facilities to carry out my project work.

Note: Students may thank to the persons whom they would like to acknowledge.

Abhishek Jaglan
00215602717



Daksh Trehan
01115602717



PriyashSinghal
03815602717

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

ML	Machine Learning
MVC	Model View Controller
HTML	Hype Text Markup Language
B/W	Between
CSS	Cascading Style Sheets
RAM	Random Access Memory
GB	Giga Byte
PC	Personal Computer
KPI	Key Performance Indicators
SVM	Support Vector Machine

LIST OF SYMBOLS

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

As it is quite evident from the current scenario that the most current, concerning and relatable topic is Coronavirus. It is natural for anyone to be concerned about the way things are happening and with the speed it is taking place in regards to coronavirus, all this instils a state of confusion and caution among people about the spread. Therefore, the main reason behind selecting the particular topic of coronavirus for this project remains to solve these confused state of mind with best to our ability projection of facts and figures in a way which is conceivable to everyone. The project 'COVID-19 DATA DASHBOARD' aims towards providing a better understanding of the coronavirus data by using correct and efficient visual methods assembled in an interactive form. The data used is taken from John Hopkins University and comprises of active, total confirmed, recovered and total deaths all over world distributed by countries. The project also makes an attempt towards predicting the recoveries of the major hit countries the next day using machine learning. Linear regression is the machine learning technique/algorithm used to predict the number of recoveries going to take place the next day.

Linear Regression is operated on two continuous variables to find relationship between them. Linear Polynomial Regression can be regarded as extended version of Linear Regression, it is implemented on related but non-linear data. It is supervised in nature, and handles non-linear data efficiently. Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity. When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors. We used Ridge Regression as well for prediction but better accuracy was struck using Linear Polynomial Regression. Therefore, in the end Linear Polynomial Regression was employed.

The dashboard created is hosted as a website and the appropriate visuals used (graphs, plots etc) are interactive and very flexible for better understanding of the current scenario. The website and its visuals are created using the Django implemented on visual studio. The javascript, CSS and HTML code is used inside the Django framework to produce the desired visual website with help of the MVC architecture.

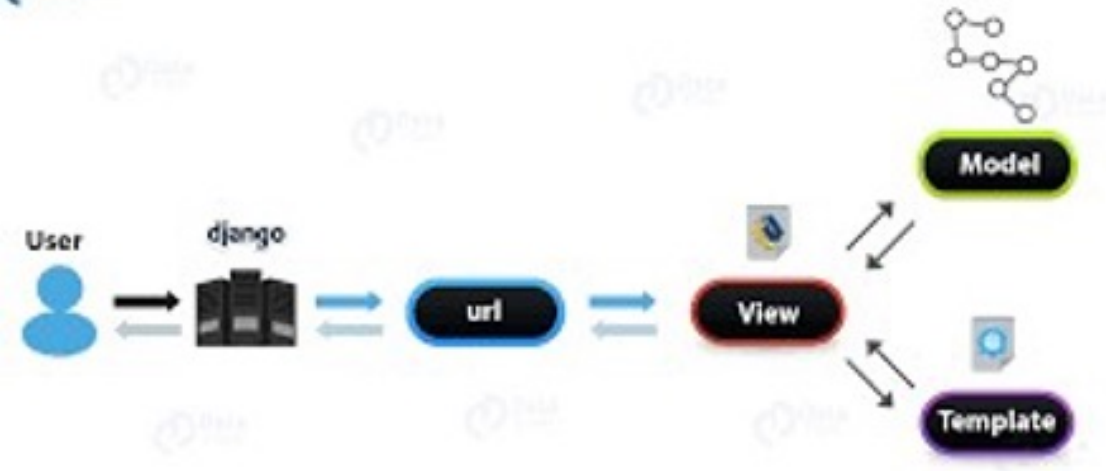


Figure 1.1

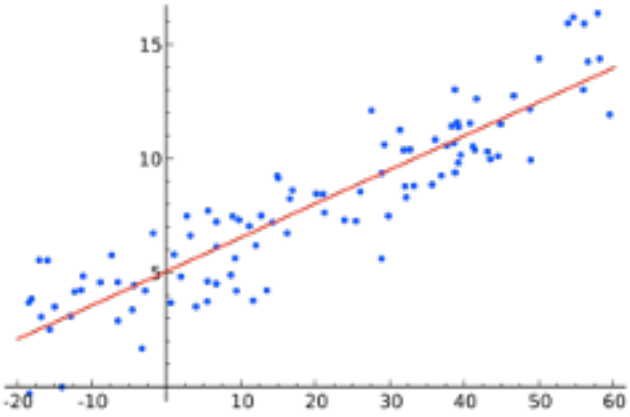


Figure 1.2

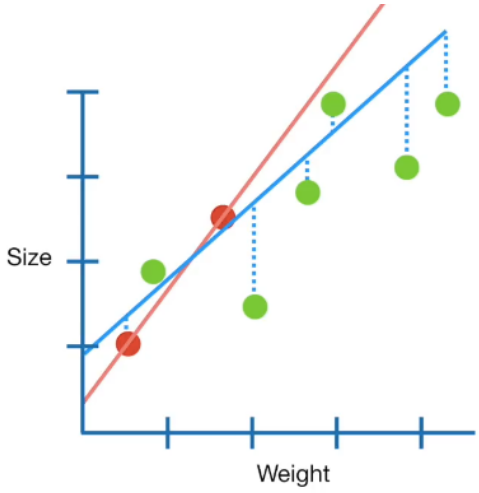


Figure 1.3

1.2 MOTIVATION

With Coronavirus being the highlight of this decade and information related to it flying all around the place, consuming and processing all the facts at once increases the complexity.
In this project we have tried to provide all the necessary facts and numbers related to coronavirus at one place for easy access and processing in a more interactive and consumable way.

1.3 OBJECTIVE

a) To provide easy access to all the facts at one place in form of a live data dashboard.
b) To provide predicted number of cases at one place as whole and as per each country for understanding future spread and impact.
c) To provide assistance to queries of users using the chat bot.
d) To showcase total number of cases and total number of recovered cases for each country.
e) To create awareness about the coronavirus and its effect around the world.

CHAPTER 2: DESCRIPTION

2.1 DESCRIPTION OF THE PROJECT

This project is one of the coronavirus related theme projects. It is a machine learning based website for a data dashboard. A data dashboard is an information management tool that visually tracks, analyses and displays key performance indicators (KPI), metrics and key data points to monitor the specific process. The dashboard consists of two fronts: front and back. The back end consists of data gathering, data preparation, data analysis, chat bot and machine learning, all of which is implemented using Python. The front end consists of making the website, converting the processed information at backend to a consumable form, and deploying all these features online.

At the back-end data for prediction and showcasing data for different purposes was gathered from the official repository of John Hopkins University. For chatbot the data was taken from cdc.gov.in to fetch questions, and answer to faqs.

At the front-end the files were processed into consumable material for website building purposes using python based open-source framework Django. The website was made presentable and interactive using CSS and HTML.

All these features combined formed a live data dashboard as a website updating itself daily, showing total number of cases for each country separately and in form of a world map for better relative understanding of the situation. It also portraits the recovering and infected cases for each countries in a graphical form for detailed view. The dashboard gives you an option to put in your queries and get the answers to them in form of a chatbot along with giving prediction of total number of cases for each country in near future. The website also offers you a feature to download the data in four different form (png, svg vector image and pdf, jpeg).

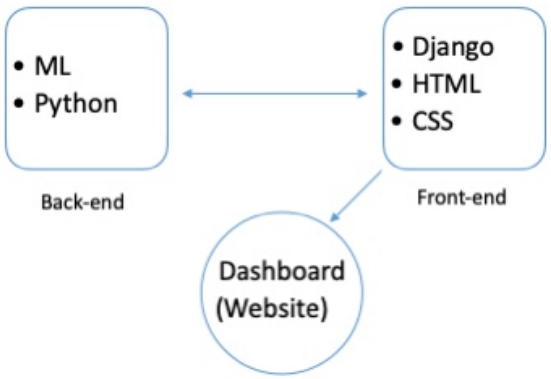


Figure 2.1

2.2 HARDWARE REQUIREMENT

Any working PC/Laptop with *4 GB RAM* and at least *i3 processor* to process the data together and work the website at same time.

2.3 SOFTWARE REQUIREMENT

The list mentioned here is subjective and can be implemented with more and less softwares depending on the requirement of the dashboard.

- Python 3.7
- Django 3.8
- Jupyter Notebook
- Visual Studio
- CSS
- HTML
- Heroku

2.4 PROS OF DATA DASHBOARD

A data dashboard is an easy-to-understand, visualised summary of data analysis that provides an at-a-glance overview of multiple areas of the field. There are many significant benefits of implementing data dashboards into your operations including:

- **Enhanced Visibility**

Data dashboards provide greater visibility with information available whenever it is required to ensure you are better placed to respond to changing conditions.

- **Timesaving Efficiency**

With data dashboards, you are no longer wasting valuable time generating reports from multiple systems. Instead, data is drawn from a centralised source and displayed as an easy to interpret visual overview.

- **Better Forecasting**

With greater insight into the buying cycle of each customer, future demand can be more accurately predicted using historic information. Businesses can more effectively plan for demand fluctuations for the next business cycle, setting measurable goals and deliverables for greater success.

• Key Performance Indicators

Data dashboards source data from multiple areas displaying the information as easy to understand visuals in real-time. This provides you with an overview of current KPIs to assess different areas of performance while creating actionable insights.

• Better Decision Making

Whether you're providing reporting and analysis for the entire organisation or functional areas of the business, a data dashboard allows you to analyse key data quickly and meticulously. Visualised interactivity serves to deliver overwhelming amounts of data in a way that is easy to understand. With the ability to easily identify what the data really means; better decisions can be made relevant to the business.

2.5 CONS OF DATA DASHBOARD

• Flashy or cluttered design, with users attempting to incorporate too much information without understanding constraints or considering their specific needs from the range of different measurable detailed data analyses provides.
• Difficulty in attaching supporting data to a dashboard and the failure of data to refresh automatically means that both these tasks must be done manually.
• The technology used in the development of data dashboards differs from other software solutions already employed in organisations and can be initially difficult to understand.
• The problems have no predetermined rules and hierarchies for how dashboard metrics are used. This means each user can use the metrics in different ways, resulting in a diverse set of data being reported.

2.6 BACK-END

2.6.1 Python

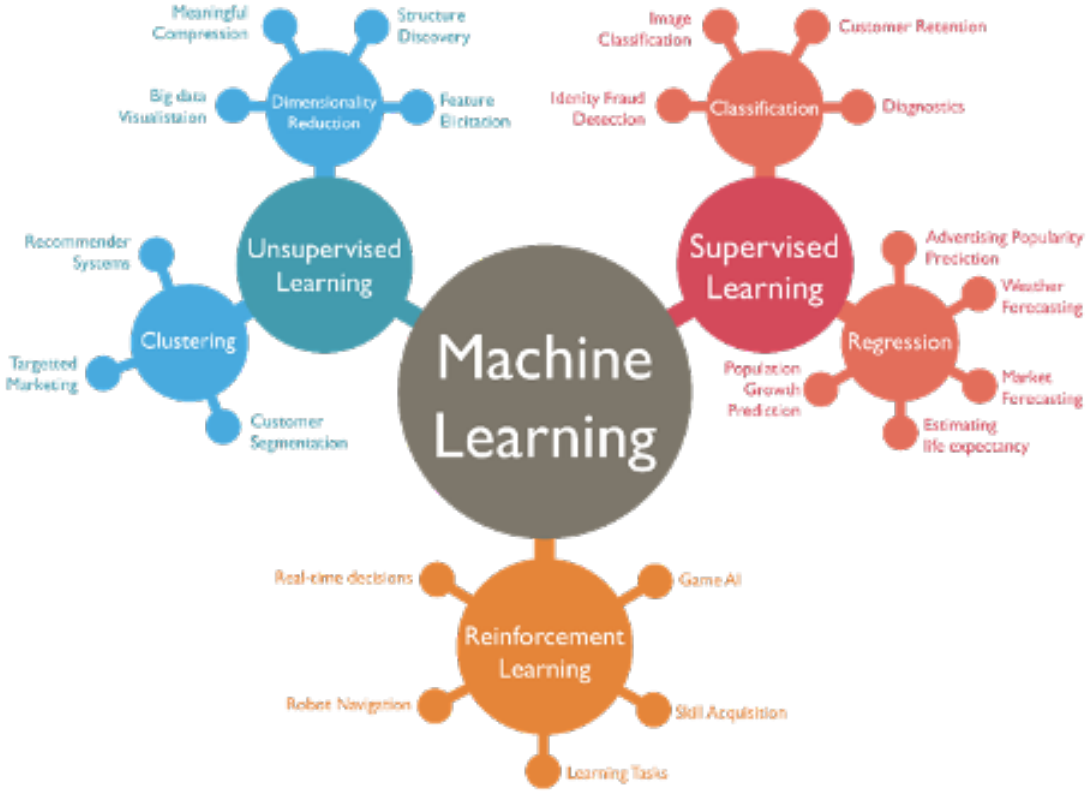
Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasises code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Figure 2.2

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

2.6.2 Machine Learning

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.



2.7 FRONT-END

2.7.1 Django

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasises reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models. It also acts as an integration language b/w front and back end.

Figure 2.4

2.7.2 HTML

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

Figure 2.5

CHAPTER 3: IMPLEMENTATION

3.1 BACK-END

3.1.1 Data Preparation

The dataset retrieved from the official repository of Johns Hopkins University. This data consists of daily case reports and daily time series summary tables. In the study, we have selected time-series summary tables in CSV format having three tables for confirmed, death, and recovered cases of COVID-19 with six properties.

Importing required libraries:-

```
import pandas as pd
import numpy as np
from sklearn import preprocessing,cross_decomposition,model_selection,metrics
import matplotlib.pyplot as plt
import seaborn as sns
import xgboost as xgb
%matplotlib inline
import dateutil
from tqdm import tqdm
from sklearn import linear_model
import datetime
import numpy as np
import pandas as pd

import plotly.graph_objects as go
from ipywidgets import widgets
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels import tsa
import numpy as np
import pandas as pd

import plotly.graph_objects as go
from ipywidgets import widgets
```

```
confirmedGlobal=pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv',encoding='utf-8',na_values=None)
confirmedGlobal
```

Taking in account, confirmed cases, active cases, death cases.
Getting values for each country:-

```
import pandas as pd
import numpy as np
from sklearn import preprocessing,cross_decomposition,model_selection,metrics
import matplotlib.pyplot as plt
import seaborn as sns
import xgboost as xgb
%matplotlib inline
import dateutil
from tqdm import tqdm
from sklearn import linear_model
import datetime
import numpy as np
import pandas as pd

import plotly.graph_objects as go
from ipywidgets import widgets
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels import tsa
import numpy as np
import pandas as pd

import plotly.graph_objects as go
from ipywidgets import widgets
```

```
confirmedGlobal=pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv',encoding='utf-8',na_values=None)
confirmedGlobal
```

Creating a data-map:-

```
dataForMap=[]
for i in uniqueCountryName:
    try:
        tempdf=df3[df3['name']==i]
        temp={}
        temp["code3"]=list(tempdf['code3'].values)[0]
        temp["name"]=i
        temp["value"]=df2[df2['Country']==i]['values'].sum()
        temp["code"]=list(tempdf['code'].values)[0]
        dataForMap.append(temp)
    except:
        pass
```

```
dataForMap
```

```
[{'code3': 'AFG', 'name': 'Afghanistan', 'value': 38129, 'code': 'AF'},
 {'code3': 'ALB', 'name': 'Albania', 'value': 9083, 'code': 'AL'},
 {'code3': 'DZA', 'name': 'Algeria', 'value': 43816, 'code': 'DZ'},
 {'code3': 'AND', 'name': 'Andorra', 'value': 1698, 'code': 'AD'},
 {'code3': 'AGO', 'name': 'Angola', 'value': 2415, 'code': 'AO'},
 {'code3': 'ATG', 'name': 'Antigua and Barbuda', 'value': 94, 'code': 'AG'},
 {'code3': 'ARG', 'name': 'Argentina', 'value': 380292, 'code': 'AR'},
 {'code3': 'ARM', 'name': 'Armenia', 'value': 43270, 'code': 'AM'},
 {'code3': 'AUS', 'name': 'Australia', 'value': 25448, 'code': 'AU'},
 {'code3': 'AUT', 'name': 'Austria', 'value': 26361, 'code': 'AT'},
 {'code3': 'AZE', 'name': 'Azerbaijan', 'value': 35844, 'code': 'AZ'},
 {'code3': 'BHR', 'name': 'Bahrain', 'value': 50756, 'code': 'BH'},
 {'code3': 'BGD', 'name': 'Bangladesh', 'value': 384583, 'code': 'BD'},
```

Preparing data Country-wise

Country-wise

```
countryDataSpe=pd.DataFrame(confirmedGlobal[confirmedGlobal['Country/Region']=='China'][confirmedGlobal.columns[4:-1]].sum()).reset_index
()
countryDataSpe.columns=['country','values']
countryDataSpe['lagVal']=countryDataSpe['values'].shift(1).fillna(0)
countryDataSpe['incrementVal']=countryDataSpe['values']-countryDataSpe['lagVal']
countryDataSpe['rollingMean']=countryDataSpe['incrementVal'].rolling(window=4).mean()
countryDataSpe=countryDataSpe.fillna(0)
datasetsForLine=[{'label':'Daily Cumulated Data','data':countryDataSpe["values"].values.tolist()},
                 {'label':'Rolling Mean 4 days','data':countryDataSpe["rollingMean"].values.tolist()}]
```

```
countryDataSpe=pd.DataFrame(confirmedGlobal[confirmedGlobal['Country/Region']=='China'][confirmedGlobal.columns[4:-1]].sum()).reset_index
()
countryDataSpe.columns=['country','values']
```

```
confirmedGlobal
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	6/18/20	6/19/20	6/20/20	6/21/20	6/2
0	NaN	Afghanistan	33.939110	67.709953	0	0	0	0	0	0	...	37599	37599	37656	37694	379
1	NaN	Albania	41.153300	20.168300	0	0	0	0	0	0	...	7654	7812	7967	8119	827
2	NaN	Algeria	28.033900	1.659600	0	0	0	0	0	0	...	39444	39847	40258	40667	410
3	NaN	Andorra	42.506300	1.521800	0	0	0	0	0	0	...	1005	1024	1024	1045	104
4	NaN	Angola	-11.202700	17.873900	0	0	0	0	0	0	...	1966	2015	2044	2060	213
5	NaN	Antigua and Barbuda	17.060800	-61.796400	0	0	0	0	0	0	...	93	94	94	94	94
6	NaN	Argentina	-38.416100	-63.616700	0	0	0	0	0	0	...	305966	312659	320884	329043	336
7	NaN	Armenia	40.069100	45.038200	0	0	0	0	0	0	...	41846	42056	42319	42477	426

Preparing HeatMap

```
dataForheatMap=[]
for i in contryNames:
    try:
        tempdf=df3[df3['Country/Region']==i]
        temp={}
        temp["name"]=i
        temp["data"]=[{'x':j,'y':k} for j,k in zip(tempdf[tempdf.columns[1:]].sum().index,tempdf[tempdf.columns[1:]].sum().values)]
        dataForheatMap.append(temp)
    except:
        pass
dataForheatMap
```

```
[{'name': 'US',
  'data': [{'x': '8/23/20', 'y': 5701645},
   {'x': '8/24/20', 'y': 5739536},
   {'x': '8/25/20', 'y': 5777710},
   {'x': '8/26/20', 'y': 5821819},
   {'x': '8/27/20', 'y': 5867785}]},
 {'name': 'Brazil',
  'data': [{'x': '8/23/20', 'y': 3605783},
   {'x': '8/24/20', 'y': 3622861},
   {'x': '8/25/20', 'y': 3660995},
   {'x': '8/26/20', 'y': 3717156},
```

Predicting Recoveries:-

Country-wise

```
countryDataSpe=pd.DataFrame(confirmedGlobal[confirmedGlobal['Country/Region']=='China'][confirmedGlobal.columns[4:-1]].sum()).reset_index
()
countryDataSpe.columns=['country','values']
countryDataSpe['lagVal']=countryDataSpe['values'].shift(1).fillna(0)
countryDataSpe['IncrementVal']=countryDataSpe['values']-countryDataSpe['lagVal']
countryDataSpe['rollingMean']=countryDataSpe['IncrementVal'].rolling(window=4).mean()
countryDataSpe=countryDataSpe.fillna(0)
datasetsForLine=[{'label':'Daily Cumulated Data','data':countryDataSpe['values'].values.tolist()},
                 {'label':'Rolling Mean 4 days','data':countryDataSpe['rollingMean'].values.tolist()}]
```

```
countryDataSpe=pd.DataFrame(confirmedGlobal[confirmedGlobal['Country/Region']=='China'][confirmedGlobal.columns[4:-1]].sum()).reset_index
()
countryDataSpe.columns=['country','values']
```

```
confirmedGlobal
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	8/18/20	8/19/20	8/20/20	8/21/20	8/2
0	NaN	Afghanistan	33.939110	67.709953	0	0	0	0	0	0	...	37599	37599	37656	37654	379
1	NaN	Albania	41.153330	20.168300	0	0	0	0	0	0	...	7654	7612	7967	8115	827
2	NaN	Algeria	28.033900	1.659600	0	0	0	0	0	0	...	39444	39647	40258	40667	410
3	NaN	Andorra	42.506300	1.521800	0	0	0	0	0	0	...	1005	1024	1024	1045	104
4	NaN	Angola	-11.202700	17.873900	0	0	0	0	0	0	...	1968	2015	2044	2068	210
5	NaN	Antigua and Barbuda	17.060800	-61.796400	0	0	0	0	0	0	...	93	94	94	94	94
6	NaN	Argentina	-38.416100	-63.616700	0	0	0	0	0	0	...	305966	312659	320884	329043	336
7	NaN	Armenia	40.069100	45.038200	0	0	0	0	0	0	...	41846	42056	42319	42477	426

```
x=input("Enter a country: ")
```

```
Enter a country: Sint Maarten
```

```
x
```

```
'Sint Maarten'
```

```
final=recoverGlobal1[recoverGlobal1['Province/State']==x]
final
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	8/18/20	8/19/20	8/20/20	8/21/20	8/22/20	8/23/20
176	Sint Maarten	Netherlands	18.0425	-63.0548	0	0	0	0	0	0	...	107	133	146	146	147	147

1 rows × 223 columns

```
data=[['1', int(final.iloc[:,-11])], ['2', int(final.iloc[:,-10])], ['3', int(final.iloc[:,-9])], ['4', int(final.iloc[:,-8])], ['5', int
(final.iloc[:,-7])], ['6', int(final.iloc[:,-6])], ['7', int(final.iloc[:,-5])], ['8', int(final.iloc[:,-4])], ['9', int(final.iloc[:,-
3])], ['10', int(final.iloc[:,-2])] ]
```

```
df = pd.DataFrame(data, columns = ['Day', 'Cases'])
df
```

	Day	Cases
0	1	107
1	2	107
2	3	133

3.1.2 Prediction of Total Cases

Dataset
The dataset retrieved from the official repository of Johns Hopkins University.

Prediction and Analysis
Coronavirus spread has conducted the society under the edge of loss in social lives. Additionally, it is crucial to investigate the transmission growth ahead and predict the future occurrences of the transmission. In concurrent, state-of-the-art mathematical models are chosen based on machine learning for a computational process to predict the spread of the virus, for instance:

- **Support Vector Regression**[5] (SVR)
- **Polynomial Regression**[6] (PR)

Machine learning and deep learning strategies are performed using the **python library** to predict the total number of confirmed, recovered, and death cases extensively. This prediction will allow undertaking specific determinations based on transmission growth, such as expanding the lockdown phase, performing the sanitation plan, and providing daily support and supplies.

Regression Analysis

Regression analysis is a section of machine learning algorithms. It is the leading machine learning algorithm. Think of straight equation line combining any two variables X and Y, which can be declared algebraically as:

$$Y = aX + b$$

Where ***b*** is declared the intercept on the y-axis, and ***a*** is called the slope of the line. Here, ***a*** and ***b*** are also called the parameters of regression analysis. These parameters should learn throughout proper learning methods.

Regression analysis contains a set of machine learning methods that enable us to predict a continuous result variable (Y) based on the value of the one or multiple predictor variables(X). It pretends a continuing connection between the result and the predictor variables.

Correlation coefficients

The correlation coefficient interpreted as the strength of a linear relationship between two variables. Karl Pearson emphasizes that the coefficient of correlation is a weight or degree of the linear correlation between two variables. He also has been generated a formula known as Correlation Coefficient. The correlation coefficient between two random variables X and Y, generally indicated by a numerical measure of the linear dependence between those variables and is defined as:

$$r(X,Y) = \frac{Cov(X,Y)}{\sigma x \sigma y}$$

$$Cov(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

$$\sigma x = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

$$\sigma y = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2$$

Where, *i = 1, 2, 3, 4, ...N*, is the collection of input and output variables. Some prediction is given below:

1. If the value of the correlation coefficient is equal to zero, it indicates no correlation between input variables X and output variable Y.
2. If the value of the correlation coefficient is equal to positive one, it indicates there is a strong relationship between the input variable and the output variable. In other words, if the input variable is increased then the output variable is also increased.
3. If the value of the correlation coefficient is equal to negative, it indicates the input variable is increased then the output variable is also decreased and so on.

Those variables that have a small or no linear correlation might have a strong nonlinear relationship. On the other hand, estimating linear correlation before fitting a model is a valuable way to recognise variables with a simple relationship. In this proposed study, we have measured the correlation coefficient between date and number of confirmed cases of COVID-2019 spread nationwide.

After fetching the data, the data has been prepared and analysed based on certain trends.

The data has been accumulated over past days to get real-time total active, recovered and death cases.

```
# Fill with the dataset
for i in dates:
    confirmed_sum = confirmed[i].sum()
    death_sum = deaths[i].sum()
    recovered_sum = recoveries[i].sum()

    world_cases.append(confirmed_sum)
    total_deaths.append(death_sum)
    total_recovered.append(recovered_sum)
    total_active.append(confirmed_sum-death_sum-recovered_sum)

    mortality_rate.append(death_sum/confirmed_sum)
    recovery_rate.append(recovered_sum/confirmed_sum)
```

```
# Fill with the dataset
for i in dates:
    confirmed_sum = confirmed[i].sum()
    death_sum = deaths[i].sum()
    recovered_sum = recoveries[i].sum()

    world_cases.append(confirmed_sum)
    total_deaths.append(death_sum)
    total_recovered.append(recovered_sum)
    total_active.append(confirmed_sum-death_sum-recovered_sum)

    mortality_rate.append(death_sum/confirmed_sum)
    recovery_rate.append(recovered_sum/confirmed_sum)
```

To get more about the data, we have calculated increase in the cases so as to get over the trends.

The final data which we can use to predict can be shown as something of this sort.

```
country_df = pd.DataFrame({'Country Name': unique_countries, 'Number of Confirmed Cases': confirmed_by_country,
                           'Number of Deaths': death_by_country, 'Number of Recoveries' : recovery_by_country,
                           'Number of Active Cases' : active_by_country,
                           'Mortality Rate': mortality_rate_by_country})
# number of cases per country/region

country_df.style.background_gradient(cmap='Blues')
```

	Country Name	Number of Confirmed Cases	Number of Deaths	Number of Recoveries	Number of Active Cases	Mortality Rate
0	US	3501466	137539	1075882	2288045	0.0392804
1	Brazil	1966748	75366	1350098	541284	0.0383201
2	India	968857	24914	612768	331175	0.0257148
3	Russia	745197	11753	522375	211069	0.0157717
4	Peru	337751	12417	226400	98934	0.0367638
5	Chile	321205	7186	292085	21934	0.022372
6	Mexico	317635	36906	252368	28361	0.11619
7	South Africa	311049	4453	160693	145903	0.0143161
8	United Kingdom	293469	45138	1386	246945	0.153808
9	Iran	264561	13410	227561	23590	0.0506877
10	Pakistan	257914	5426	178737	73751	0.021038
11	Spain	257494	28413	150376	78705	0.110344
12	Italy	243506	34997	196016	12493	0.143721
13	Saudi Arabia	240474	2325	183048	55101	0.0096684

Similar approach is used to get the data province-vise.

	Province/State Name	Country	Number of Confirmed Cases	Number of Deaths	Number of Recoveries	Mortality Rate
0	New York	US	404006	32427	0	0.0802637
1	Sao Paulo	Brazil	393176	18640	246941	0.0474088
2	California	US	364885	7375	0	0.0207814
3	Florida	US	301810	4521	0	0.0149796
4	Texas	US	291058	3622	0	0.0124443
5	Maharashtra	India	275640	10928	152613	0.0396459
6	England	United Kingdom	250885	40462	0	0.161277
7	Metropolitana	Chile	241345	5957	224418	0.0246825
8	Moscow	Russia	231270	4234	167810	0.0183076
9	Lima	Peru	181131	5763	0	0.0318168
10	New Jersey	US	176278	15634	0	0.0886895
11	Illinois	US	157825	7427	0	0.0470585

PREDICTION

Prediction

```
days_since_1_1 = np.array([i for i in range(len(dates))]).reshape(-1,1)
world_cases = np.array(world_cases).reshape(-1,1)
total_deaths = np.array(total_deaths).reshape(-1, 1)
total_recovered = np.array(total_recovered).reshape(-1, 1)
```

```
days_since_1_1.shape
```

```
(111, 1)
```

```
days_in_future = 30
future_forecast = np.array([i for i in range(len(dates)+days_in_future)]).reshape(-1, 1)
adjusted_dates = future_forecast[:-30]
print(future_forecast)
```

```
[[  0]
 [  1]
```

We have taken the past data from 1 Jan 2020 till date and we are targeting to predict the future cases for 30 days.

```
start = '1/1/2020'
start_date = datetime.datetime.strptime(start, '%m/%d/%Y')
future_forecast_dates = []
for i in range(len(future_forecast)):
    future_forecast_dates.append((start_date + datetime.timedelta(days=i)).strftime('%m/%d/%Y'))
```

```
X_train_confirmed, X_test_confirmed, y_train_confirmed, y_test_confirmed = train_test_split(days_since_1_22, world_cases, test_
shuffle=False)
```

```
# transform data for polynomial regression
poly = PolynomialFeatures(degree=2)
poly_X_train_confirmed = poly.fit_transform(X_train_confirmed)
poly_X_test_confirmed = poly.fit_transform(X_test_confirmed)
poly_future_forecast = poly.fit_transform(future_forecast)
```

```
# polynomial regression
linear_model = LinearRegression(normalize=True, fit_intercept=False)
linear_model.fit(poly_X_train_confirmed, y_train_confirmed)
test_linear_pred = linear_model.predict(poly_X_test_confirmed)
linear_pred = linear_model.predict(poly_future_forecast)
print('MAE:', mean_absolute_error(test_linear_pred, y_test_confirmed))
print('MSE:',mean_squared_error(test_linear_pred, y_test_confirmed))
```

```
MAE: 4613740.780833273
MSE: 34531881633301.33
```

```
plt.plot(y_test_confirmed)
plt.plot(test_linear_pred)
plt.legend(['Test Data', 'Polynomial Regression Predictions'])
linear_model.score(poly_X_test_confirmed,y_test_confirmed)
```

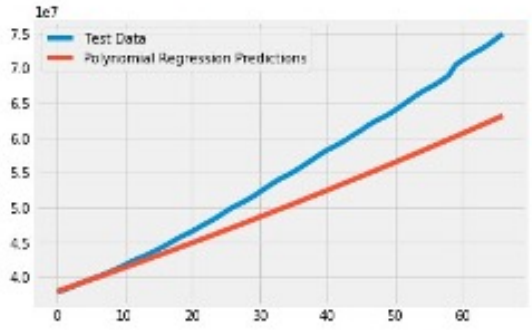
Implying Linear Polynomial Regression for prediction.

```
# Polynomial Regression
linear_model = LinearRegression(normalize=True, fit_intercept=False)
linear_model.fit(poly_X_train_confirmed, y_train_confirmed)
test_linear_pred = linear_model.predict(poly_X_test_confirmed)
linear_pred = linear_model.predict(poly_future_forecast)
print('MAE:', mean_absolute_error(test_linear_pred, y_test_confirmed))
print('MSE:',mean_squared_error(test_linear_pred, y_test_confirmed))
```

```
MAE: 4613740.780833273
MSE: 34531881633301.33
```

```
plt.plot(y_test_confirmed)
plt.plot(test_linear_pred)
plt.legend(['Test Data', 'Polynomial Regression Predictions'])
linear_model.score(poly_X_test_confirmed,y_test_confirmed)
```

```
0.7162579228503609
```



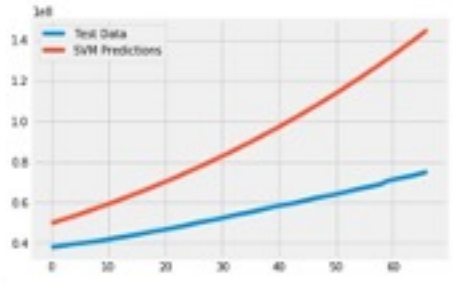
We further tried to use SVM to predict the cases for better accuracy, but Linear Regression was working better.

```
svm_confirmed = SVR(shrinking=True, kernel='poly',gamma=0.01, epsilon=1,degree=5, C=0.1)
svm_confirmed.fit(X_train_confirmed, y_train_confirmed)
svm_pred = svm_confirmed.predict(future_forecast)
```

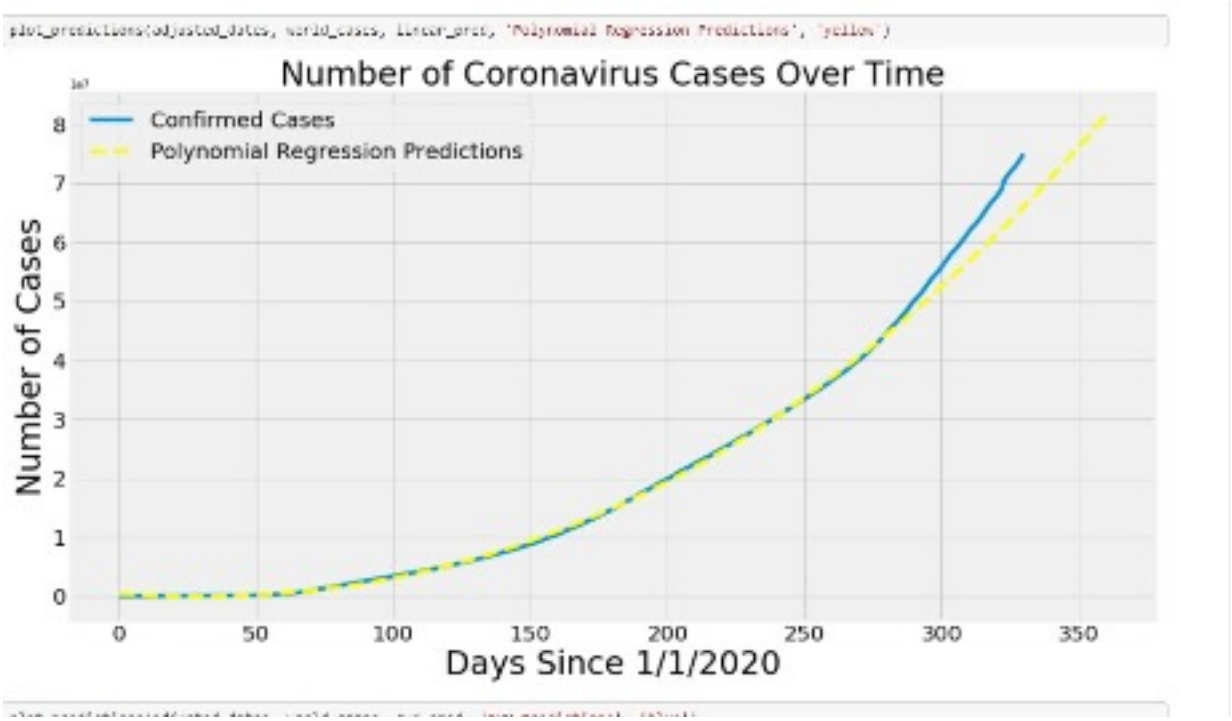
```
c:\program files\python37\lib\site-packages\sklearn\utils\validation.py:724: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

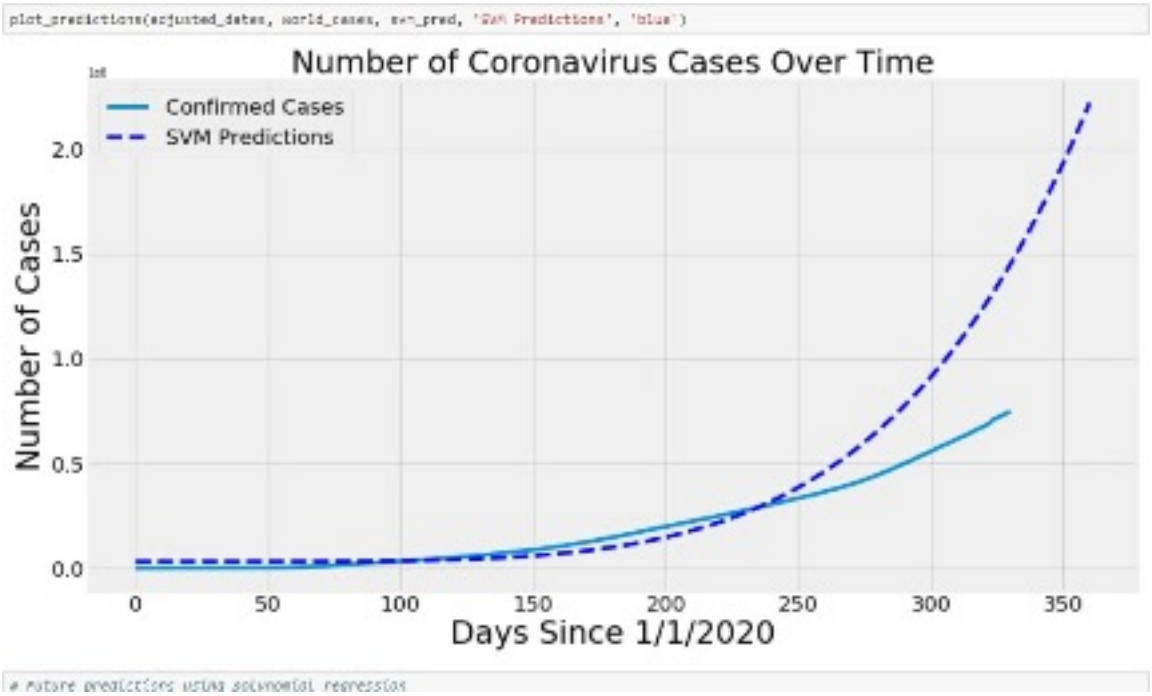
```
svm_test_pred = svm_confirmed.predict(X_test_confirmed)
plt.plot(y_test_confirmed)
plt.plot(svm_test_pred)
plt.legend(['Test Data', 'SVM Predictions'])
print('MAE:', mean_absolute_error(svm_test_pred, y_test_confirmed))
print('MSE:',mean_squared_error(svm_test_pred, y_test_confirmed))
```

```
MAE: 35763206.801073655
MSE: 1502323032934197.8
```



The final output:





```
# Future predictions using polynomial regression
linear_pred = linear_pred.reshape(1,-1)[0]
poly_df = pd.DataFrame({'Date': future_forecast_dates[-20:], 'Predicted number of Confirmed Cases Worldwide': np.round(linear_pred[-2
0:])})
poly_df
```

	Date	Predicted number of Confirmed Cases Worldwide
0	12/07/2020	68007644.0
1	12/08/2020	68454946.0
2	12/09/2020	68903710.0
3	12/10/2020	69353939.0
4	12/11/2020	69805632.0
5	12/12/2020	70258788.0
6	12/13/2020	70713408.0
7	12/14/2020	71169492.0
8	12/15/2020	71627040.0
9	12/16/2020	72086051.0
10	12/17/2020	72546526.0
11	12/18/2020	73008465.0
12	12/19/2020	73471868.0
13	12/20/2020	73936735.0
14	12/21/2020	74403065.0
15	12/22/2020	74870859.0
16	12/23/2020	75340117.0
17	12/24/2020	75810839.0
18	12/25/2020	76283025.0
19	12/26/2020	76756674.0

```
# Future predictions using SVM
svm_df = pd.DataFrame({'Date': future_forecast_dates[-20:], 'SVM Predicted # of Confirmed Cases Worldwide': np.round(svm_pred[-20:])})
svm_df
```

	Date	SVM Predicted # of Confirmed Cases Worldwide
0	12/07/2020	170144684.0
1	12/08/2020	172578581.0
2	12/09/2020	175067484.0
3	12/10/2020	177612563.0
4	12/11/2020	180137137.0
5	12/12/2020	182716443.0
6	12/13/2020	185325740.0
7	12/14/2020	187965280.0
8	12/15/2020	190635363.0
9	12/16/2020	193335190.0
10	12/17/2020	196065865.0
11	12/18/2020	198831283.0
12	12/19/2020	201628059.0
13	12/20/2020	204452886.0
14	12/21/2020	207311431.0
15	12/22/2020	210202572.0
16	12/23/2020	213125377.0
17	12/24/2020	216083128.0
18	12/25/2020	219073102.0
19	12/26/2020	222096574.0

3.1.3 Chatbot

To implement chatbot, we have taken in account the data from cdc.gov.in to fetch answers and questions.

```
import requests
from bs4 import BeautifulSoup
import re
import pandas as pd
import json

url = 'https://www.cdc.gov/coronavirus/2019-ncov/faq.html'
response = requests.get(url)
print(response)
# make sure we got a valid response
if(response.ok):
  # get the full data from the response
  data = response.text
  print(data)
```

Getting questions & answers and implementing a data frame for both of them.

Dumping the final file together.

```
with open('answers.json', 'w') as f:
  json.dump(answers, f)
with open('questions.json', 'w') as f:
  json.dump(questions, f)

answers = pd.read_json('/content/drive/My Drive/Colab Datasets/COVID Chat Bot/answers.json')
questions = pd.read_json('/content/drive/My Drive/Colab Datasets/COVID Chat Bot/questions.json')
print(len(questions))
print(len(answers))

with open("test.json", "w") as f:
  json.dump(q_and_a, f)

with open("test.json", "r") as f:
  new_data = json.load(f)
with open('answers.json', 'w') as f:
  json.dump(answers, f)
with open('questions.json', 'w') as f:
  json.dump(questions, f)

q_and_a

df = pd.DataFrame(q_and_a)

df.to_csv('cdc_qa.csv', index=False)
```

To implement the chatbot, we are going to use TF-IDF vector r. tf-idf stands for “term frequency – inverse document” frequency and it measures how important a word in a document is relative to the whole corpus. Without going into too much detail, documents that have similar content will have similar tf-idf vectors. Intuitively, if a context and a response have similar words they are more likely to be a correct pair. At least more likely than random. Many libraries out there (such as scikit-learn) come with built-in tf-idf functions, so it’s very easy to use.

```
with open('answers.json', 'w') as f:
  json.dump(answers, f)
with open('questions.json', 'w') as f:
  json.dump(questions, f)

answers = pd.read_json('/content/drive/My Drive/Colab Datasets/COVID Chat Bot/answers.json')
questions = pd.read_json('/content/drive/My Drive/Colab Datasets/COVID Chat Bot/questions.json')
print(len(questions))
print(len(answers))

with open("test.json", "w") as f:
  json.dump(q_and_a, f)

with open("test.json", "r") as f:
  new_data = json.load(f)
with open('answers.json', 'w') as f:
  json.dump(answers, f)
with open('questions.json', 'w') as f:
  json.dump(questions, f)

q_and_a

df = pd.DataFrame(q_and_a)

df.to_csv('cdc_qa.csv', index=False)
```

Giving answers to common queries.

```
COVIDbot('what is coronavirus?')
```

```
'A novel coronavirus is a new coronavirus that has not been previously identified. The virus causing coronavirus disease 2019 (COVID-19),
is not the same as the coronaviruses that commonly circulate among humans\xaband cause mild illness, like the common cold.\nA diagnosis w
ith coronavirus 229E, NL63, OC43, or HKU1 is not the same as a COVID-19 diagnosis. Patients with COVID-19 will be evaluated and cared for
differently than patients with common coronavirus diagnosis.\n'
```

Replying to greeting by users.

```
welcome_input = ("hello", "hi", "greetings", "sup", "what's up","hey",)
welcome_response = ["hi", "hey", "*nods*", "hi there", "hello", "I am glad! You are talking to me"]
def welcome(user_response):
    for word in user_response.split():
        if word.lower() in welcome_input:
            return random.choice(welcome_response)
```

Replying to user queries:

```
flag=True
print("Greetings! I am a chatbot and I will try to answer your questions about COVID-19. If you want to exit, type Bye!")
while(flag==True):
    user_response = input()
    user_response = user_response.lower()
    if(user_response not in ['bye','shutdown','exit', 'quit']):
        if(user_response=='thanks' or user_response=='thank you'):
            flag=False
            print("Chatbot : You are welcome..")
        else:
            if(welcome(user_response)!=None):
                print("Chatbot : "+welcome(user_response))
            else:
                print("Chatbot : ",end="")
                print(COVIDbot(user_response))
    else:
        flag=False
        print("Chatbot : Bye!!! ")
```

Final output:

```
Greetings! I am a chatbot and I will try to answer your questions about COVID-19. If you want to exit, type Bye!
what is covid
Chatbot : This is an emerging, rapidly evolving situation and CDC will continue to provide updated information as it becomes available. C
DC works 24/7 to protect people's health. More information about CDC's response to COVID-19 is available online.

how to cure covid
Chatbot : People can fight stigma and help, not hurt, others by providing social support. Counter stigma by learning and sharing facts. C
ommunicating the facts that viruses do not target specific racial or ethnic groups and how COVID-19 actually spreads can help stop stigm
a.

symptoms of covid
Chatbot : No. The symptoms of COVID-19 are similar in children and adults. However, children with confirmed COVID-19 have generally prese
nted with mild symptoms. Reported symptoms in children include cold-like symptoms, such as fever, runny nose, and cough. Vomiting and dia
rrhea have also been reported. It's not known yet whether some children may be at higher risk for severe illness, for example, children w
ith underlying medical conditions and special healthcare needs. There is much more to be learned about how the disease impacts children.

what is novel coronavirus
Chatbot : A novel coronavirus is a new coronavirus that has not been previously identified. The virus causing coronavirus disease 2019 (C
OVID-19), is not the same as the coronaviruses that commonly circulate among humans and cause mild illness, like the common cold.
A diagnosis with coronavirus 229E, NL63, OC43, or HKU1 is not the same as a COVID-19 diagnosis. Patients with COVID-19 will be evaluated
and cared for differently than patients with common coronavirus diagnosis.

How can people help stop stigma related to COVID-19?
Chatbot : People can fight stigma and help, not hurt, others by providing social support. Counter stigma by learning and sharing facts. C
ommunicating the facts that viruses do not target specific racial or ethnic groups and how COVID-19 actually spreads can help stop stigm
a.

how does the virus originated
Chatbot : This virus was first detected in Wuhan City, Hubei Province, China. The first infections were linked to a live animal market, b
ut the virus is now spreading from person-to-person. It's important to note that person-to-person spread can happen on a continuum. Some
viruses are highly contagious (like measles), while other viruses are less so.
The virus that causes COVID-19 seems to be spreading easily and sustainably in the community ("community spread") in some affected geogra
```

3.1.4 Vaccination

To implement the vaccination part of the dashboard, we have taken data from the follow source : https://ourworldindata.org/covid-vaccinations .

The data used contains the following attributes and information regarding it:

- Country - this is the country for which the vaccination information is provided;
- Country ISO Code - ISO code for the country;
- Date- date for the data entry; for some of the dates we have only the daily vaccinations, for others, only the (cumulative) total;
- Total number of vaccinations - this is the absolute number of total immunizations in the country;
- Total number of people vaccinated - a person, depending on the immunization scheme, will receive one or more (typically 2) vaccines; at a certain moment, the number of vaccination might be larger than the number of people;
- Total number of people fully vaccinated - this is the number of people that received the entire set of immunization according to the immunization scheme (typically 2); at a certain moment in time, there might be a certain number of people that received one vaccine and another number (smaller) of people that received all vaccines in the scheme;
- Daily vaccinations (raw) - for a certain data entry, the number of vaccination for that date/country;
- Daily vaccinations - for a certain data entry, the number of vaccination for that date/country;
- Total vaccinations per hundred - ratio (in percent) between vaccination number and total population up to the date in the country;
- Total number of people vaccinated per hundred - ratio (in percent) between population immunized and total population up to the date in the country;
- Total number of people fully vaccinated per hundred - ratio (in percent) between population fully immunized and total population up to the date in the country;
- Number of vaccinations per day - number of daily vaccination for that day and country;
- Daily vaccinations per million - ratio (in ppm) between vaccination number and total population for the current date in the country;
- Vaccines used in the country - total number of vaccines used in the country (up to date);
- Source name - source of the information (national authority, international organization, local organization etc.);
- Source website - website of the source of information;

Implementation process:

- We initialize the Python packages required for data ingestion, preparation and visualization. We mostly use Plotly for visualization.

- Then we read the data file and aggregate the data on few fields (country, iso_code and vaccines - that is the vaccination scheme used in a certain country).

Objectives of the exploration will be:

- What vaccination schemes are used in various countries;
- Total number of distinct vaccinations and it's percentage proportion;
- Daily vaccinations and daily vaccinations per million;
- Total people vaccinated and percent of people vaccinated against population;
- The dataset also contains the information about the total number of people completely vaccinated (and percent of them).

We visualize the latest (maximum) values and as well for the variation in time of the above mentioned values.

3.2 FRONT-END

As we know Django is python based open-source framework which follows MVC(Model View Controller) architectural pattern used in the rapid development of the website with clean design without worrying too much about setting up an environment to start.

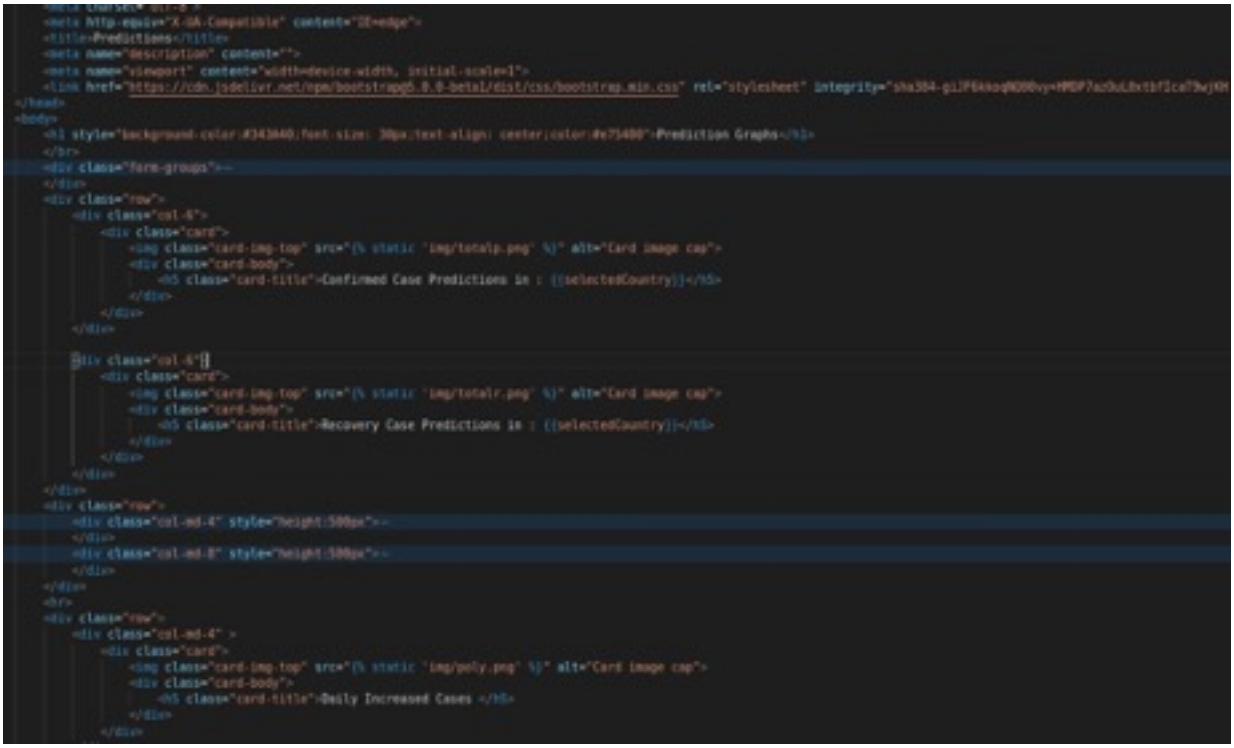
For Covid-19 Dashboard data fetched online server is analysed and used many approaches to serialises the data which is used in views.py files in Django to further process the data and represent it to the user.

Data fetched is sent to html(front-end) file for representation in the form of maps, texts as well as charts and to display these chart special Library- Chart.js is used . It also involves the part of Chatbot which compare the text entered by user and similar results present to user's questions in file data fetched from online server.

All the data fetched from backend is displayed in front end with the help of styling using CSS and JavaScript. Main goal of the project is to make site user attractive and solving queries using Chatbot. Django worked as an intermediate between data analysed using ML and Representation of data to user.

3.2.1 Prediction (Front-end)

For processing of prediction data on front-end in Django we've used HTML and CSS.



3.2.2 Chatbot

Here the files fetched from backend are stored in a single variable which is processed through different functions using bootstrap and jquery.

3.3.3 Graphs

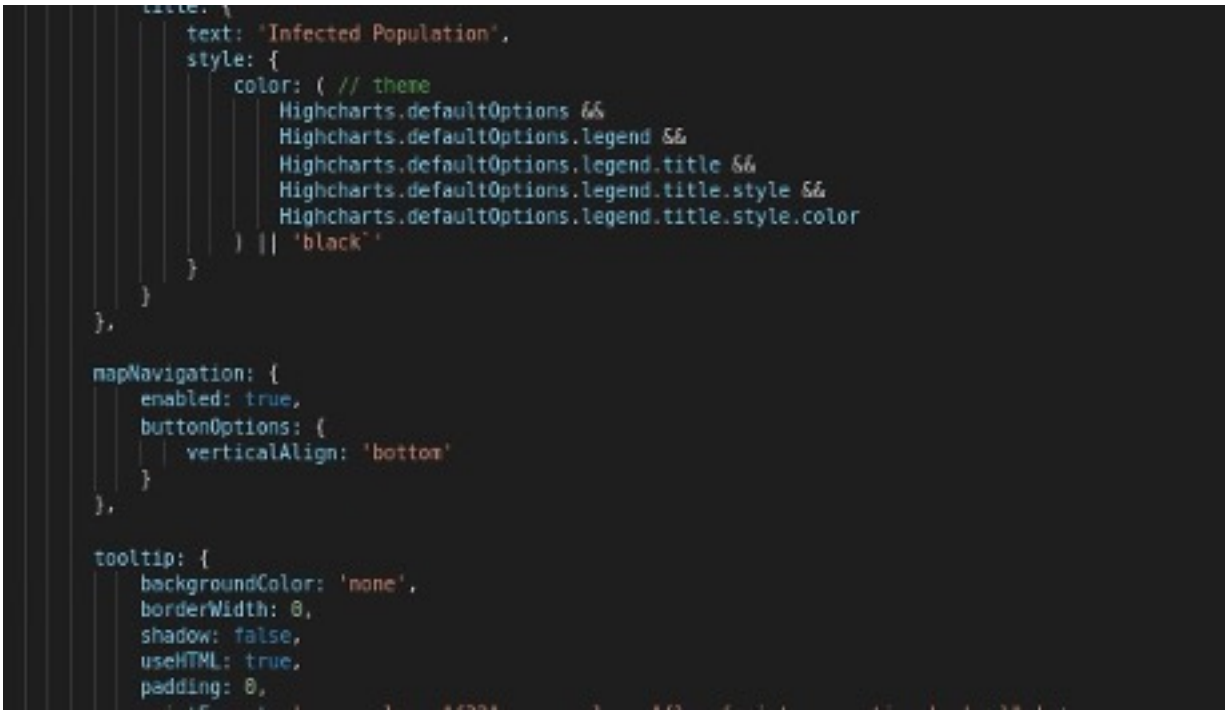
The following graph represents the code of two functions for the heat-map and the projection of last six days of data on the dashboard using Django.

```
        try:
            tempdf = df3[df3['name'] == i]
            temp = {}
            temp["code3"] = list(tempdf['code3'].values)[0]
            temp["name"] = i
            temp["value"] = barPlotData[barPlotData['Country/Region']
                                        == i]['values'].sum()
            temp["code"] = list(tempdf['code'].values)[0]
            dataForMap.append(temp)
        except:
            pass
    return dataForMap


def getHeatMapData(confirmedGlobal, countryNames):
    df3 = confirmedGlobal[list(
        confirmedGlobal.columns[1:2])+list(list(confirmedGlobal.columns.values)[-6:-1])]
    dataForheatMap = []
    for i in countryNames:
        try:
            tempdf = df3[df3['Country/Region'] == i]
            temp = {}
            temp["name"] = i
            temp["data"] = [{'x': j, 'y': k} for j, k in zip(
                tempdf[tempdf.columns[1:]].sum().index, tempdf[tempdf.columns[1:]].sum().values)]
            dataForheatMap.append(temp)
```

The following graph represents code for the number of cases on the given day for each country in form of a horizontal bar graph.

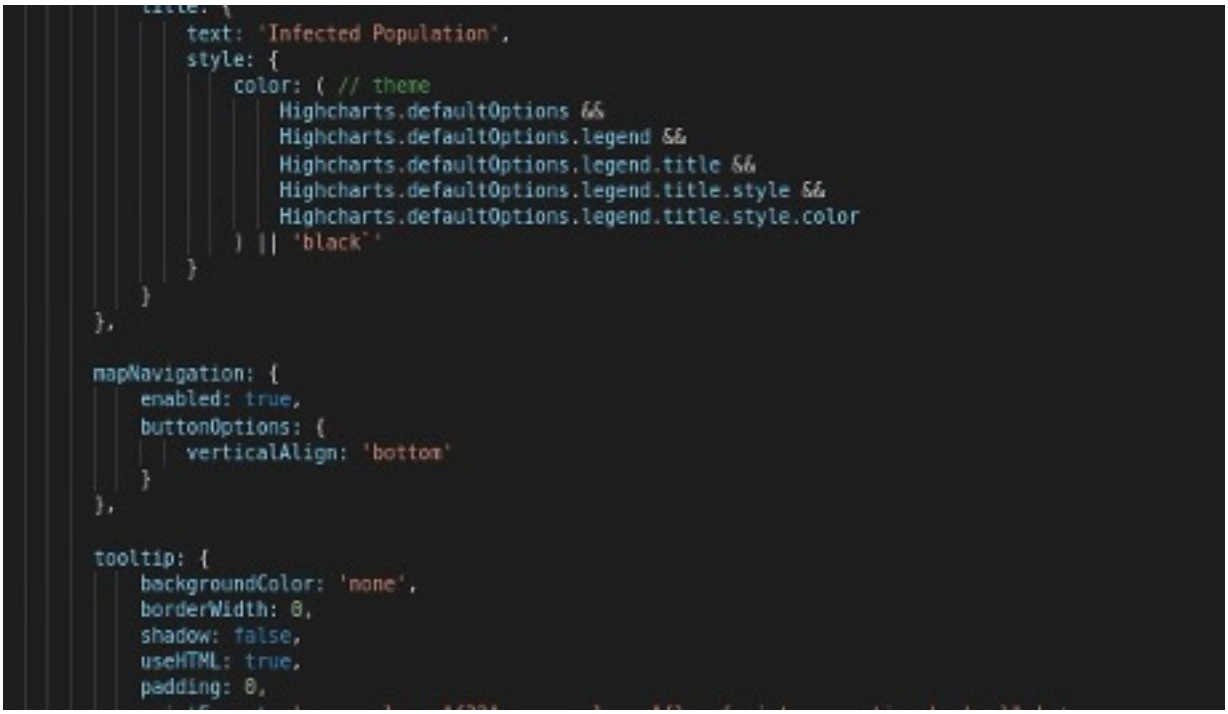
```
var chartOptions = {
                    legend: {
                        display: false
                        },
                    scales: {
                        xAxes: [{
                            barPercentage:1
                            }],
                        yAxes: [
                                // {barPercentage: 1},
                                // {gridLines:{drawBorder: false}},
                                {ticks: {display:false}},
                                // {ticks: {mirror: true}},
                                {display: false},

                                ],
                    elements: {
                        rectangle: {
                        borderSkipped: 'left',
                        }
                    }
                    }
                };
var chart = new Chart(ctx, {
  type: 'horizontalBar',
  data: {
    labels: {{countryNames|safe}},
    datasets: [
      {
        label: "Infected Counts",
        backgroundColor: 'rgb(50, 50,221 )',
        borderColor: "rgb(255, 99, 132)",
        data: {{barPlotVals|safe}},
      },
    ],
  },
  options: chartOptions,
```

The following graph represents the code for the actual total number of days and rolling mean of four days for each country.



```
        text: 'Infected Population',
        style: {
            color: ( // there
                Highcharts.defaultOptions &&
                Highcharts.defaultOptions.legend &&
                Highcharts.defaultOptions.legend.title &&
                Highcharts.defaultOptions.legend.title.style &&
                Highcharts.defaultOptions.legend.title.style.color
            ) || 'black'
        }
    }
},

mapNavigation: {
    enabled: true,
    buttonOptions: {
        verticalAlign: 'bottom'
    }
},

tooltip: {
    backgroundColor: 'none',
    borderWidth: 0,
    shadow: false,
    useHTML: true,
    padding: 0,
```

3.3.4 Prediction (Back-end)



```
        text: 'Infected Population',
        style: {
            color: ( // there
                Highcharts.defaultOptions &&
                Highcharts.defaultOptions.legend &&
                Highcharts.defaultOptions.legend.title &&
                Highcharts.defaultOptions.legend.title.style &&
                Highcharts.defaultOptions.legend.title.style.color
            ) || 'black'
        }
    }
},

mapNavigation: {
    enabled: true,
    buttonOptions: {
        verticalAlign: 'bottom'
    }
},

tooltip: {
    backgroundColor: 'none',
    borderWidth: 0,
    shadow: false,
    useHTML: true,
    padding: 0,
```

3.3.5 Vaccination

Data is fetched from live server through machine learning which interact directly from an open source python django framework, helping it in representing data with front-end web based structure using various technologies such as HTML, CSS, javascript and many more . Data is passed using python language in views.py files and fetched directly using MVC (Model View Architecture), making it user interactive using libraries such as bootstrap and various others. After representing it in local host; it can be deployed to any server based applications like heroku, pythonanywhere, etc. In this specific case, heroku is used as a live server and can directly work on any of the operating device.

CHAPTER 4: RESULT AND SNAPSHOT OF PROJECT

4.1 RESULT

The result of this project is a live data dashboard on coronavirus that updates itself daily and depicts the data in different forms. The different forms in which the data is represented are heat-map, a table which shows total number of cases in last 6 days for each country, total number of cases and rolling mean of four days for each country and a horizontal bar graph showing number of cases for each country.

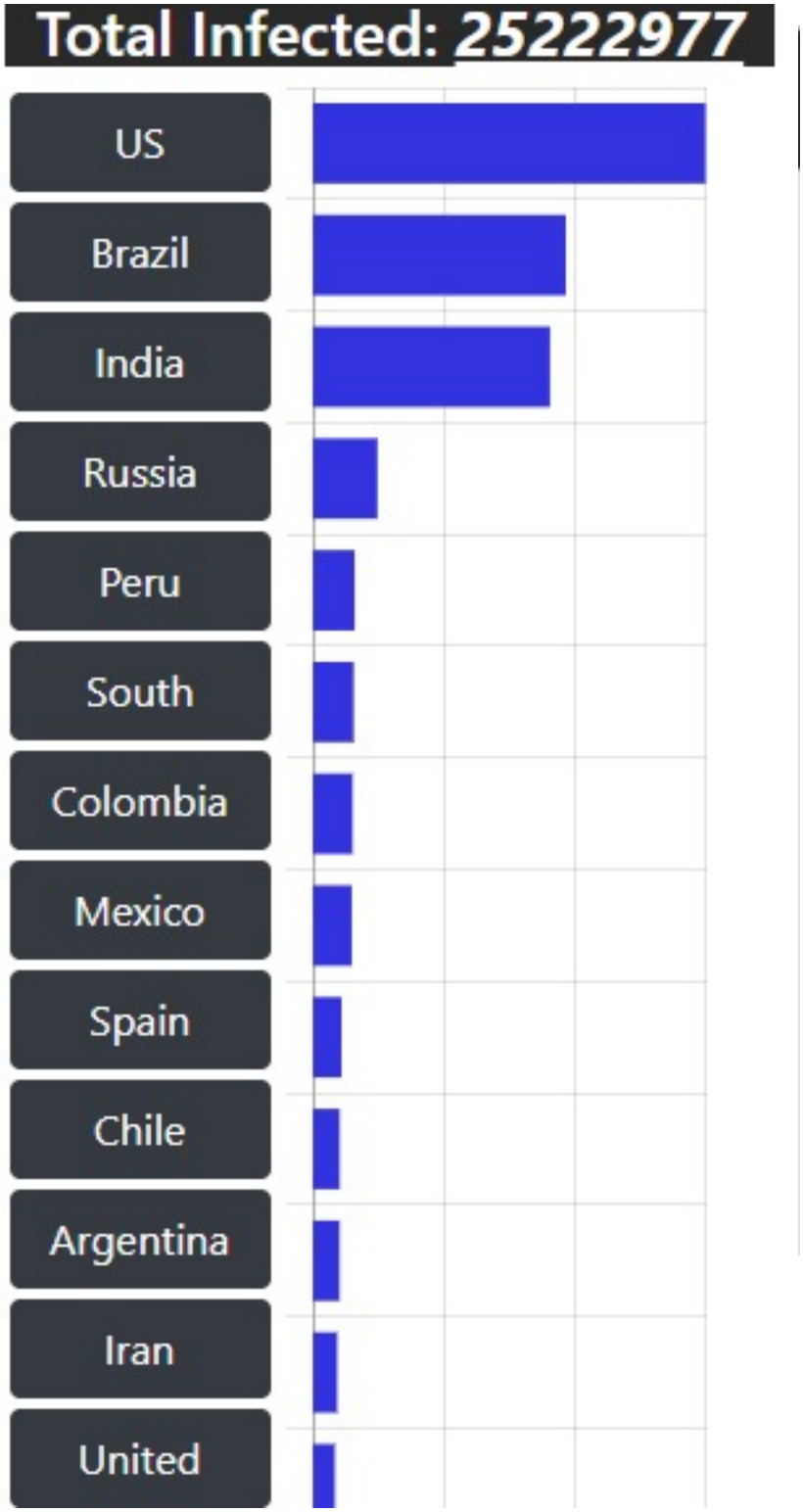
The dashboard also allows you to download the data in four different forms. You can also converse with a chatbot regarding different queries and a prediction algorithm which displays a list of predicted number of cases for each country.

4.2 SNAPSHOTS OF PROJECT

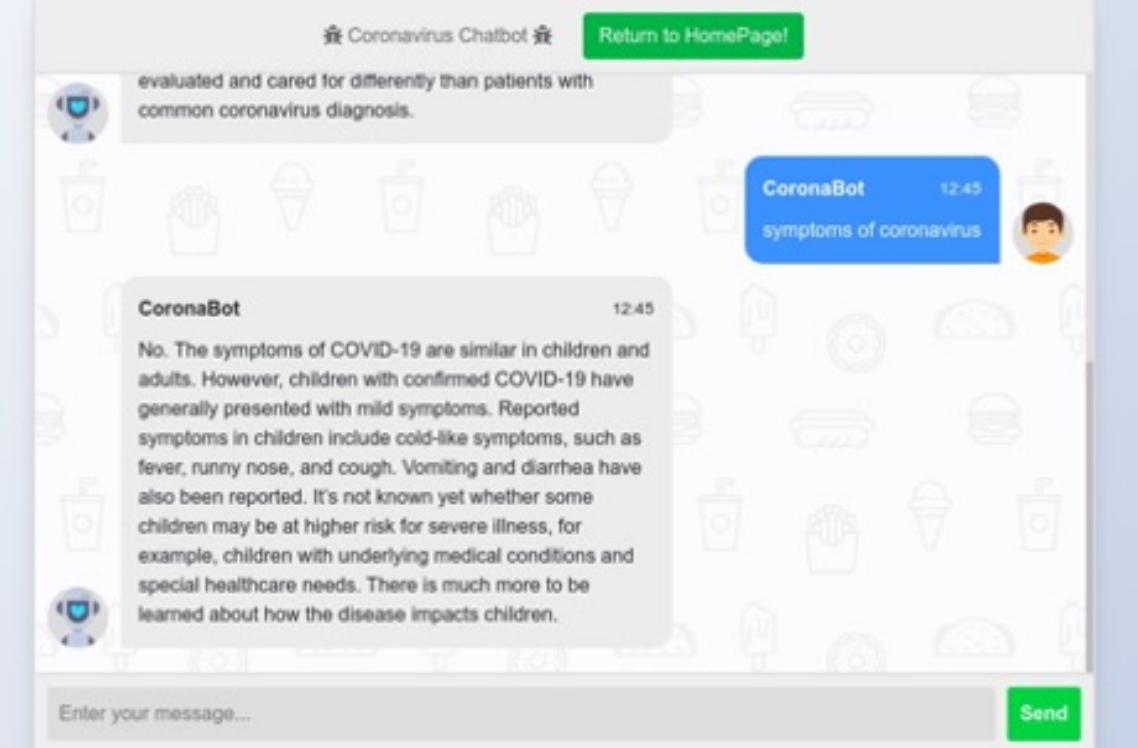
4.2.1 Home page

The homepage displays the heat map of world in centre of the screen, buttons for chatbot (top-right), prediction (top-right), last six days count (right) and horizontal bar graph for each country (left).

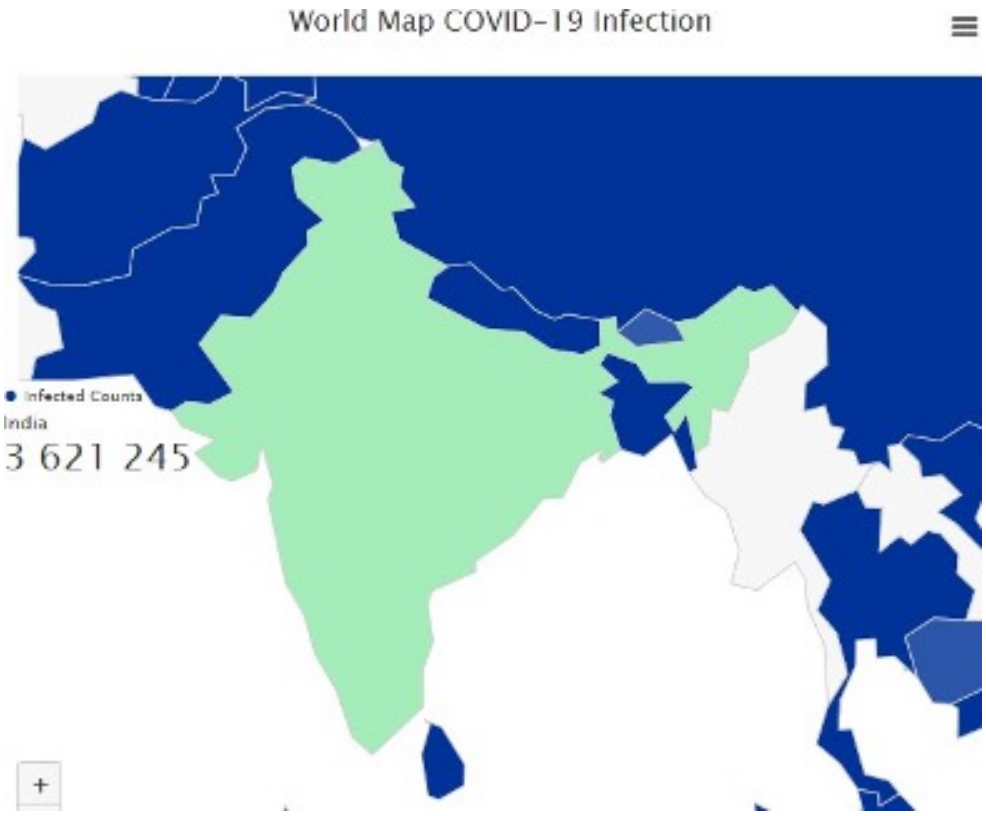
4.2.2 Last Six Days Count and Downloading Format



4.2.3 Chatbot



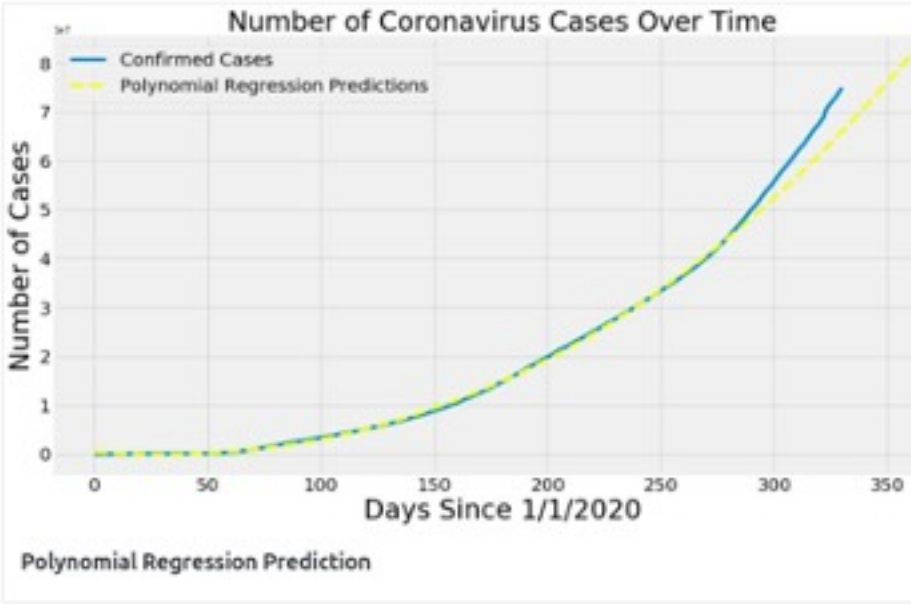
4.2.4 World-map



4.2.5 Prediction

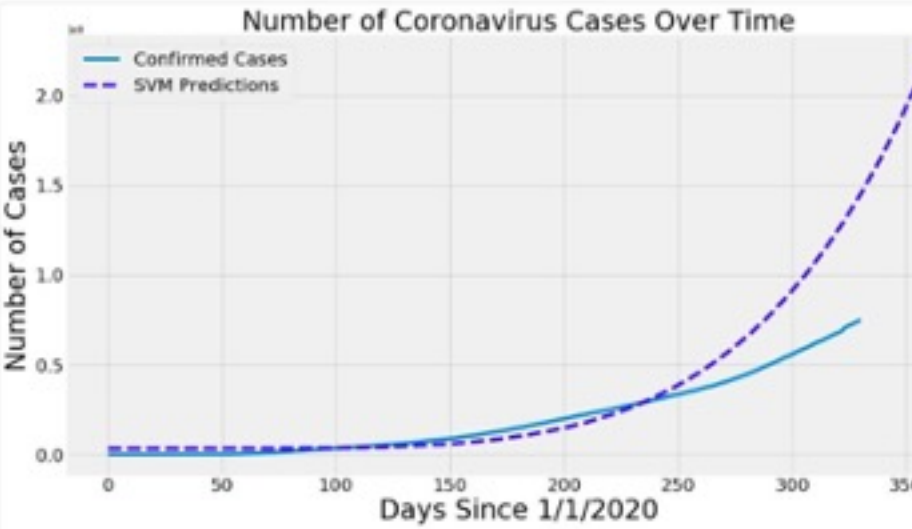
	Date	Predicted number of Confirmed Cases Worldwide
0	12/07/2020	68007644.0
1	12/08/2020	68454946.0
2	12/09/2020	68903710.0
3	12/10/2020	69353939.0
4	12/11/2020	69805632.0
5	12/12/2020	70258788.0
6	12/13/2020	70713408.0
7	12/14/2020	71169492.0
8	12/15/2020	71627040.0
9	12/16/2020	72086051.0
10	12/17/2020	72546526.0
11	12/18/2020	73008465.0
12	12/19/2020	73471868.0
13	12/20/2020	73936735.0
14	12/21/2020	74403065.0
15	12/22/2020	74870859.0
16	12/23/2020	75340117.0
17	12/24/2020	75810839.0
18	12/25/2020	76283025.0
19	12/26/2020	76756674.0

Daily Increased Cases



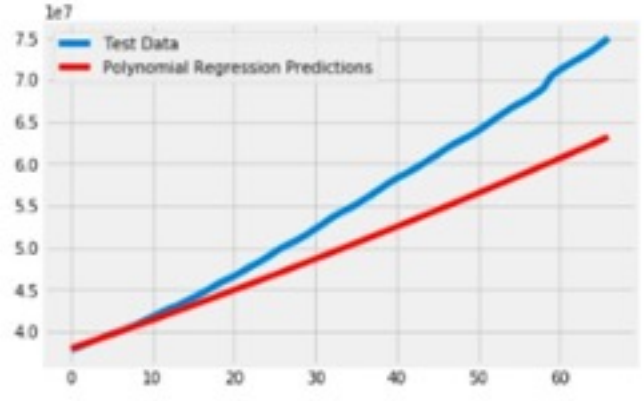
Polynomial Regression Prediction

	Date	SVM Predicted # of Confirmed Cases Worldwide
0	12/07/2020	170114664.0
1	12/08/2020	172576581.0
2	12/09/2020	175067464.0
3	12/10/2020	177587563.0
4	12/11/2020	180137137.0
5	12/12/2020	182716443.0
6	12/13/2020	185325740.0
7	12/14/2020	187965290.0
8	12/15/2020	190635355.0
9	12/16/2020	193336198.0
10	12/17/2020	196068085.0
11	12/18/2020	198831283.0
12	12/19/2020	201626059.0
13	12/20/2020	204452685.0
14	12/21/2020	207311431.0
15	12/22/2020	210202572.0
16	12/23/2020	213126377.0
17	12/24/2020	216083129.0
18	12/25/2020	219073102.0
19	12/26/2020	222096574.0



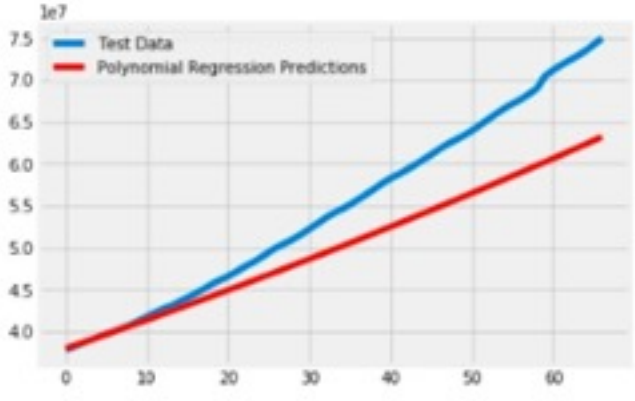
SVM Prediction

12	12/30/2020	73479898.0
13	12/31/2020	73944864.0
14	01/01/2021	74411294.0
15	01/02/2021	74879188.0
16	01/03/2021	75348547.0
17	01/04/2021	75819369.0
18	01/05/2021	76291657.0
19	01/06/2021	76765408.0

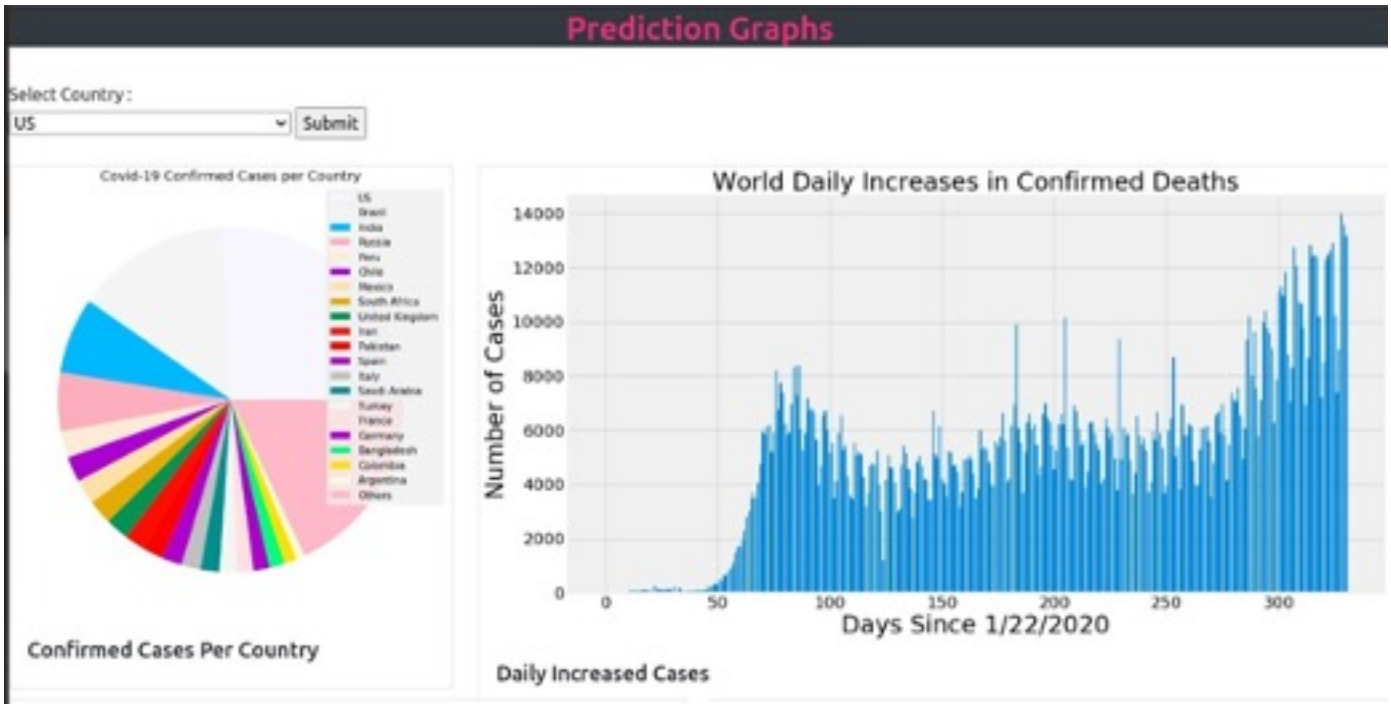


}
Confirmed Case Predictions in : Worldwide

12	12/30/2020	73479898.0
13	12/31/2020	73944864.0
14	01/01/2021	74411294.0
15	01/02/2021	74879188.0
16	01/03/2021	75348547.0
17	01/04/2021	75819369.0
18	01/05/2021	76291657.0
19	01/06/2021	76765408.0



Recovery Case Predictions in : Worldwide



CHAPTER 5: CONCLUSION

The consequence of this venture is a live information dashboard on Covid that refreshes itself every day and portrays the information in various structures. The various structures in which the information is spoken to are heat-map, a table which shows complete number of cases in most recent 6 days for every nation, absolute number of cases and moving mean of four days for every nation and a level reference chart demonstrating number of cases for every nation.

The dashboard additionally permits you to download the information in four unique structures. You can likewise talk with a chatbot with respect to various inquiries and an expectation calculation which shows a rundown of anticipated number of cases for every nation.

REFERENCES

[1] COVID-19 Outbreak Prediction using Machine Learning Algorithm | by Wie Kiang H. | Towards Data Science

[2] Regression Analysis of COVID-19 using Machine Learning Algorithms - IEEE Conference PublicationDeep Learning For Chatbots, Part 2 – Implementing A Retrieval-Based Model In TensorFlow (kdnuggets.com)

[3] Covid-19-Dashboard/Covid-19 Chatbot.ipynb at master · dakshtrehan/Covid-19-Dashboard (github.com)

[4] Kaggle - platform used to accumulate/retrieve data from. Data was taken from John Hopkins University's regular data update.

[5] Google - the search engine was used to connect to different websites and answer queries.

[6] StackOverflow - the website was used to resolve doubts and errors encountered during implementation of programme.

[7] Python documentation website - used to research and quantifiably give attention to specific attributes to increase the accuracy.

[8] Django Packages website - this website was used the same way as python documentation website for 45better performance of the Django programming.