

A
Minor Project Report on
“COVID-19 OUTBREAK PREDICTION USING
MACHINE LEARNING”

In partial fulfillment of requirements for the degree of
Bachelor of Technology (B. Tech.)

in
Computer Science and Engineering



Submitted by
Ms. Ritika Joshi (170327)

Under the Guidance of
Mr. Siddhanta Kumar Singh

Computer Science and Engineering
SCHOOL OF ENGINEERING AND TECHNOLOGY
Mody University and Science and Technology
Lakshmangarh, Distt. Sikar-332311

December 2020

A C K N O W L E D G E M E N T

I sincerely express my gratitude to my guide for his benevolent guidance in completing the report on **Covid-19 Outbreak Prediction using Machine Learning**. His kindness and help have been the source of encouragement for me.

I am grateful to him for the guidance, inspiration and constructive suggestions that helped me in the preparation of this project.

Ritika Joshi

170327

CERTIFICATE

This is to certify that the minor project report entitled “Covid-19 Outbreak Prediction using Machine Learning” submitted by Ms. Ritika Joshi, as a partial fulfillment for the requirement of B. Tech. VII Semester examination of the School of Engineering and Technology, Mody University of Science and Technology, Lakshmangarh for the academic session 2019-2020 is an original project work carried out under the supervision and guidance of Dr. Siddhanta Kumar Singh has undergone the requisite duration as prescribed by the institution for the project work.

PROJECT GUIDE:

Approval Code: AUT_20_CSE_F19_02

Name: Mr. Siddhanta Kumar Singh

Date: 27/12/2020

HEAD OF DEPARTMENT

Name: Dr. A. Senthil

Date: 27/12/2020

EXAMINER-I:

Name: Dr. Sunil Kumar Jangir

Dept: CSE

EXAMINER-II

Name: Dr. Ajay Kumar Singh

Dept: CSE

ABSTRACT

The aim of this project is to make a model which will the forecast the number of confirmed cases covid-19 virus in the upcoming days. Covid-19 is a infectious disease which is affecting a huge number of people all around the world.

This virus was first identified in Wuhan, china and later spread throughout the world causing a pandemic which forced most of the countries to go into lockdown.

Various machine learning models and time series forecasting models.

The predictive model will be created using machine learning and using the dataset obtained from Kaggle. Machine learning automates the formation of analytical models. It is a branch of artificial intelligence focused on the principle that data can be learned from processes, It can find pattern and take decisions.

Time series forecasting will be used which is a type of predictive model. Time series forecasting is the use of a model centered on earlier observed values to evaluate future values.

Table of Contents

Sr.no.	Topics	Page no.
1.	Introduction	1
1.1	<i>-Present System</i>	1
1.2	<i>-Proposed System</i>	2
2.	System Design	3
2.1	<i>-System flowchart</i>	3
2.2	<i>-Dataset</i>	4
3.	Hardware and Software details	5
4.	Implementation Work Details	6
4.1	<i>-Real life applications</i>	6
4.2	<i>-Data implementation and program execution</i>	6
5.	Source Code	7
6.	Input/output Screens/ Model's Photograph	23
7.	System Testing	27
8.	Conclusion	28
8.1	<i>-Limitations</i>	28
8.2	<i>-Scope for future work</i>	28
9.	Bibliography	29
10.	Annexures	30
	<i>- Plagiarism Report</i>	

List of Figures

Fig no.	Title	Page no.
6.1	Growth of different types of cases in India	23
6.2	Confirmed cases Linear Regression Prediction	23
6.3	Polynomial Regression Prediction for confirmed cases	24
6.4	SVM regressor Prediction for confirmed cases	24
6.5	Holts Linear Model Prediction for confirmed cases	25
6.6	Holt's Winter model prediction for confirmed cases	25
6.7	AR model prediction for confirmed cases	26
6.8	SARIMA model Prediction for confirmed cases	26

Chapter1: Introduction

1. INTRODUCTION

The aim of this project is to make a predictive model which will predict the trajectory of the outbreak of the covid-19 virus in the upcoming days. Covid-19 is a infectious disease which is affecting a huge number of people all around the world. It was first identified in Wuhan, china and then later spread all over the world causing a pandemic.

Since no vaccine is developed which can be available all throughout the world, we have to take preventive measures which can stop the spread of the disease. Since lockdown cannot last forever, we have to know how fast the spread is and how much more people will be infected.

The predictive model will be created using machine learning and using the dataset obtained from Kaggle. Machine learning automates the formation of analytical models. It is a branch of artificial intelligence focused on the principle that data can be learned from processes, It can find pattern and take decisions.

Time series forecasting will be used which is a type of predictive model. Time series forecasting is the use of a model centered on earlier observed values to evaluate future values.

1.1 PRESENT SYSTEM

Various work on this problem related to covid-19 is being done. Officials all over the world are using several outbreak predictions models for covid-19 to make informed decisions and implement relevant control measures. Simple statistical models have received greater attention from authorities among the standard models for covid-19 global pandemic prediction. One of the works suggests using SEIR models. SEIR means susceptible-exposed-infected-recovered model.

This model aims to forecast factors like the spread of a disease, the total number of infected and the span of an outbreak, and estimate different epidemiological parameters like the number of reproductive. Such models can illustrate how the outcome of the disease can be affected by various public health measures.

PROPOSED SYSTEM

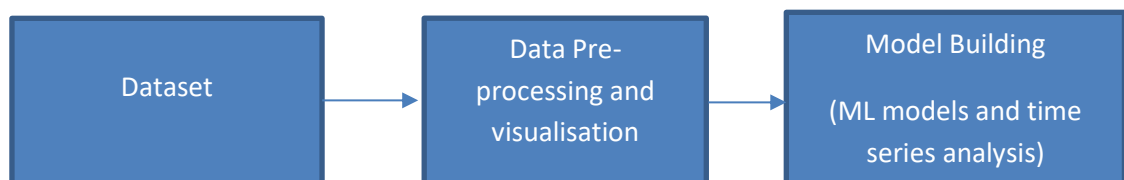
In this project, we will first collect and evaluate the dataset. We will transform the raw data into an accessible format and visualize it using data preprocessing. The various machine learning algorithms such as Linear regression, polynomial regression, SVM, holt's linear model, Holt's winter model, AR model, ARIMA model, SARIMA model is used. The tools used in this project are mainly sklearn for model selection, NumPy library which is used to work with the arrays and pandas that use a key data structure called a dataframe that allows us to store and manipulate tabular data in observation rows and variable columns, matplotlib is a library of plotting that is used to plot graphs. After implementing the model, the model with the least mean square error will be considered the best fit model.

Chapter 2: System Design

2. System Design

The dataset is first preprocessed and visualized so that it is in a useable format for analysis. After this we model the data using as Linear regression, polynomial regression, SVM, holt's linear model, Holt's winter model, AR model, ARIMA model, SARIMA. Then we evaluate the model and choose the best one according to its root mean square.

2.1 System flowchart



The flowchart depicts the following

1.Dataset

The dataset involves the collection of data from various sources.

2. Data Pre-processing and visualization

In order to obtain accurate results data preprocessing is done to check if there is any inconsistency in the data, if there is it is handled accordingly. We then visualize the data to study the pattern and trends in the data.

3. Model Building

Various models are used in this project-:

Linear Regression

Polynomial Regression

SVM

Holt's Linear

Holt's Winter Model

Auto Regressive Model (AR)

Moving Average Model (MA)

ARIMA Model

SARIMA Model

2.2 DATASET

In this project the dataset is taken from Kaggle which is the Novel Corona Virus 2019 Dataset and the goal is to study the effect and spread of COVID-19 in the coming days, conduct predictions and time series forecasting.

Chapter 3: Hardware and Software Details

3.1 Hardware Details

Processor: Intel Core i5

RAM: 8.00 GB

64 bit operating system

3.2 Software Details

Python 3.7(64 bit)

Jupyter notebook

Chapter 4: Implementation work details

4. Implementation work details

First the data is pre-processed and visualization is done and analyzed. Afterwards various models are used to train the data and the model with the least root mean squared error is selected as the best fit model. Various machine learning models are used and time series forecasting models such as holt's linear model and ARIMA model is used. The dataset is obtained from Kaggle.

4.1 Real life applications

It can be used by government for predicting the extend of the spread of the infectious disease and take actions accordingly.

4.3 Data implementation and program execution

The data is analyzed and visualized afterwards. On different models, the data is trained and the one with the least mean square error is considered to be the best fit model and can be used for forecasting. The program is executed on Jupyter notebook.

Chapter 5: Source Code

```
import warnings

warnings.filterwarnings('ignore')

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


import plotly.express as px

import plotly.graph_objects as go

from plotly.subplots import make_subplots

import numpy as np

import datetime as dt

from datetime import timedelta

from sklearn.model_selection import GridSearchCV

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import silhouette_score,silhouette_samples

from sklearn.linear_model import LinearRegression,Ridge,Lasso

from sklearn.svm import SVR

from sklearn.metrics import mean_squared_error,r2_score

import statsmodels.api as sm

from statsmodels.tsa.api import Holt,SimpleExpSmoothing,ExponentialSmoothing

from sklearn.preprocessing import PolynomialFeatures

from statsmodels.tsa.stattools import adfuller

from pmdarima import auto_arima

std=StandardScaler()
```

```

covid=pd.read_csv(r"C:\Users\Ritika\Desktop\covid_19_data.csv")

covid.head()

print("Shape of the dataset: ",covid.shape)

print("Checking for null values:\n",covid.isnull().sum())

print("Checking Data-type of each column:\n",covid.dtypes)

#Dropping column as SNo is of no use, and "Province/State" contains too many missing
values

covid.drop(["SNo"],1,inplace=True)

#Converting "Observation Date" into Datetime format

covid["ObservationDate"]=pd.to_datetime(covid["ObservationDate"])

grouped_country=covid.groupby(["Country/Region","ObservationDate"]).agg({"Confirmed":'sum',"Recovered":'sum',"Deaths":'sum'})

grouped_country["Active Cases"]=grouped_country["Confirmed"]-
grouped_country["Recovered"]-grouped_country["Deaths"]

grouped_country["log_confirmed"]=np.log(grouped_country["Confirmed"])

grouped_country["log_active"]=np.log(grouped_country["Active Cases"])

#Grouping different types of cases as per the date

datewise=covid.groupby(["ObservationDate"]).agg({"Confirmed":'sum',"Recovered":'sum',"Deaths":'sum'})

datewise["Days Since"]=datewise.index-datewise.index.min()

print("Basic Information")

```

```

print("Total number of countries with Disease Spread:
",len(covid["Country/Region"].unique()))

print("Total number of Confirmed Cases around the World: ",datewise["Confirmed"].iloc[-1])

print("Total number of Recovered Cases around the World: ",datewise["Recovered"].iloc[-1])

print("Total number of Deaths Cases around the World: ",datewise["Deaths"].iloc[-1])

print("Total number of Active Cases around the World: ",(datewise["Confirmed"].iloc[-1]-
datewise["Recovered"].iloc[-1]-datewise["Deaths"].iloc[-1]))

print("Total number of Closed Cases around the World: ",datewise["Recovered"].iloc[-1]+
datewise["Deaths"].iloc[-1])

print("Number of Confirmed Cases in last 24 hours: ",datewise["Confirmed"].iloc[-1]-
datewise["Confirmed"].iloc[-2])

print("Number of Recovered Cases in last 24 hours: ",datewise["Recovered"].iloc[-1]-
datewise["Recovered"].iloc[-2])

print("Number of Death Cases in last 24 hours: ",datewise["Deaths"].iloc[-1]-
datewise["Deaths"].iloc[-2])

fig=px.bar(x=datewise.index,y=datewise["Confirmed"]-datewise["Recovered"]-
datewise["Deaths"])

fig.update_layout(title="Distribution of Number of Active Cases",
                  xaxis_title="Date",yaxis_title="Number of Cases",)

fig.show()

india_data=covid[covid["Country/Region"]=="India"]

datewise_india=india_data.groupby(["ObservationDate"]).agg({"Confirmed":'sum',"Recovered":'sum',"Deaths":'sum'})

print(datewise_india.iloc[-1])

print("Total Active Cases: ",datewise_india["Confirmed"].iloc[-1]-
datewise_india["Recovered"].iloc[-1]-datewise_india["Deaths"].iloc[-1])

```

```

print("Total Closed Cases: ",datewise_india["Recovered"].iloc[-
1]+datewise_india["Deaths"].iloc[-1])

fig=go.Figure()

fig.add_trace(go.Scatter(x=datewise_india.index, y=datewise_india["Confirmed"],
                        mode='lines+markers',
                        name='Confirmed Cases'))

fig.add_trace(go.Scatter(x=datewise_india.index, y=datewise_india["Recovered"],
                        mode='lines+markers',
                        name='Recovered Cases'))

fig.add_trace(go.Scatter(x=datewise_india.index, y=datewise_india["Deaths"],
                        mode='lines+markers',
                        name='Death Cases'))

fig.update_layout(title="Growth of different types of cases in India",
                  xaxis_title="Date",yaxis_title="Number of
Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()


datewise["Days Since"]=datewise.index-datewise.index[0]
datewise["Days Since"]=datewise["Days Since"].dt.days


train_ml=datewise.iloc[:int(datewise.shape[0]*0.95)]
valid_ml=datewise.iloc[int(datewise.shape[0]*0.95):]
model_scores=[]


lin_reg=LinearRegression(normalize=True)

```



```

lin_reg.fit(np.array(train_ml["Days Since"]).reshape(-
1,1),np.array(train_ml["Confirmed"]).reshape(-1,1))

prediction_valid_linreg=lin_reg.predict(np.array(valid_ml["Days Since"]).reshape(-1,1))

model_scores.append(np.sqrt(mean_squared_error(valid_ml["Confirmed"],prediction_valid
_linreg)))

print("Root Mean Square Error for Linear Regression:
",np.sqrt(mean_squared_error(valid_ml["Confirmed"],prediction_valid_linreg)))

plt.figure(figsize=(11,6))

prediction_linreg=lin_reg.predict(np.array(datewise["Days Since"]).reshape(-1,1))

linreg_output=[]

for i in range(prediction_linreg.shape[0]):

    linreg_output.append(prediction_linreg[i][0])

fig=go.Figure()

fig.add_trace(go.Scatter(x=datewise.index, y=datewise["Confirmed"],
                        mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add_trace(go.Scatter(x=datewise.index, y=linreg_output,
                        mode='lines',name="Linear Regression Best Fit Line",
                        line=dict(color='black', dash='dot'))))

fig.update_layout(title="Confirmed Cases Linear Regression Prediction",
                  xaxis_title="Date",yaxis_title="Confirmed
Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

train_ml=datewise.iloc[:int(datewise.shape[0]*0.95)]

```

```

valid_ml=datewise.iloc[int(datewise.shape[0]*0.95):]

poly = PolynomialFeatures(degree = 8)

train_poly=poly.fit_transform(np.array(train_ml["Days Since"]).reshape(-1,1))
valid_poly=poly.fit_transform(np.array(valid_ml["Days Since"]).reshape(-1,1))
y=train_ml["Confirmed"]

linreg=LinearRegression(normalize=True)
linreg.fit(train_poly,y)

prediction_poly=linreg.predict(valid_poly)
rmse_poly=np.sqrt(mean_squared_error(valid_ml["Confirmed"],prediction_poly))
model_scores.append(rmse_poly)
print("Root Mean Squared Error for Polynomial Regression: ",rmse_poly)


comp_data=poly.fit_transform(np.array(datewise["Days Since"]).reshape(-1,1))
plt.figure(figsize=(11,6))
predictions_poly=linreg.predict(comp_data)

fig=go.Figure()
fig.add_trace(go.Scatter(x=datewise.index, y=datewise["Confirmed"],
                        mode='lines+markers',name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=datewise.index, y=predictions_poly,
                        mode='lines',name="Polynomial Regression Best Fit",

```

```

        line=dict(color='black', dash='dot'))

fig.update_layout(title="Confirmed Cases Polynomial Regression Prediction",

                  xaxis_title="Date",yaxis_title="Confirmed Cases",

                  legend=dict(x=0,y=1,traceorder="normal"))

fig.show()


new_prediction_poly=[]

for i in range(1,18):

    new_date_poly=poly.fit_transform(np.array(datewise["Days Since"].max()+i).reshape(-
1,1))

    new_prediction_poly.append(linreg.predict(new_date_poly)[0])


train_ml=datewise.iloc[:int(datewise.shape[0]*0.95)]
valid_ml=datewise.iloc[int(datewise.shape[0]*0.95):]


#Intializing SVR Model

svm=SVR(C=1,degree=6,kernel='poly',epsilon=0.01)


#Fitting model on the training data

svm.fit(np.array(train_ml["Days Since"]).reshape(-
1,1),np.array(train_ml["Confirmed"]).reshape(-1,1))


prediction_valid_svm=svm.predict(np.array(valid_ml["Days Since"]).reshape(-1,1))


model_scores.append(np.sqrt(mean_squared_error(valid_ml["Confirmed"],prediction_valid
_svm)))

print("Root Mean Square Error for Support Vectore Machine:
",np.sqrt(mean_squared_error(valid_ml["Confirmed"],prediction_valid_svm)))

```

```

plt.figure(figsize=(11,6))

prediction_svm=svm.predict(np.array(datewise["Days Since"]).reshape(-1,1))

fig=go.Figure()

fig.add_trace(go.Scatter(x=datewise.index, y=datewise["Confirmed"],
                        mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add_trace(go.Scatter(x=datewise.index, y=prediction_svm,
                        mode='lines',name="Support Vector Machine Best fit Kernel",
                        line=dict(color='black', dash='dot'))))

fig.update_layout(title="Confirmed Cases Support Vectore Machine Regressor Prediction",
                  xaxis_title="Date",yaxis_title="Confirmed
Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()


new_date=[]

new_prediction_lr=[]

new_prediction_svm=[]

for i in range(1,18):

    new_date.append(datewise.index[-1]+timedelta(days=i))

    new_prediction_lr.append(lin_reg.predict(np.array(datewise["Days
Since"].max()+i).reshape(-1,1))[0][0])

    new_prediction_svm.append(svm.predict(np.array(datewise["Days
Since"].max()+i).reshape(-1,1))[0])


pd.set_option('display.float_format', lambda x: '%.6f' % x)

model_predictions=pd.DataFrame(zip(new_date,new_prediction_lr,new_prediction_poly,new_prediction_svm),

```

```

        columns=["Dates","Linear Regression Prediction","Polynonmial
Regression Prediction","SVM Prediction"])

model_predictions.head()


model_train=datewise.iloc[:int(datewise.shape[0]*0.95)]

valid=datewise.iloc[int(datewise.shape[0]*0.95):]

y_pred=valid.copy()


holt=Holt(np.asarray(model_train["Confirmed"])).fit(smoothing_level=0.4,
smoothing_slope=0.4,optimized=False)


y_pred["Holt"]=holt.forecast(len(valid))

model_scores.append(np.sqrt(mean_squared_error(y_pred["Confirmed"],y_pred["Holt"])))

print("Root Mean Square Error Holt's Linear Model:
",np.sqrt(mean_squared_error(y_pred["Confirmed"],y_pred["Holt"])))


fig=go.Figure()

fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
                        mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],
                        mode='lines+markers',name="Validation Data for Confirmed Cases",))

fig.add_trace(go.Scatter(x=valid.index, y=y_pred["Holt"],
                        mode='lines+markers',name="Prediction of Confirmed Cases",))

fig.update_layout(title="Confirmed Cases Holt's Linear Model Prediction",
                  xaxis_title="Date",yaxis_title="Confirmed
Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

```

```

holt_new_date=[]
holt_new_prediction=[]
for i in range(1,18):
    holt_new_date.append(datewise.index[-1]+timedelta(days=i))
    holt_new_prediction.append(holt.forecast((len(valid)+i))[-1])

model_predictions["Holt's Linear Model Prediction"]=holt_new_prediction
model_predictions.head()

model_train=datewise.iloc[:int(datewise.shape[0]*0.95)]
valid=datewise.iloc[int(datewise.shape[0]*0.95):]
y_pred=valid.copy()

es=ExponentialSmoothing(np.asarray(model_train['Confirmed']),seasonal_periods=14,trend
='add', seasonal='mul').fit()

y_pred["Holt's Winter Model"]=es.forecast(len(valid))

model_scores.append(np.sqrt(mean_squared_error(y_pred["Confirmed"],y_pred["Holt's
Winter Model"])))

print("Root Mean Square Error for Holt's Winter Model:
",np.sqrt(mean_squared_error(y_pred["Confirmed"],y_pred["Holt's Winter Model"])))

fig=go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
                        mode='lines+markers',name="Train Data for Confirmed Cases"))
fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],
                        mode='lines+markers',name="Validation Data for Confirmed Cases",))

```

```

fig.add_trace(go.Scatter(x=valid.index, y=y_pred["Holt's Winter Model"],
                        mode='lines+markers',name="Prediction of Confirmed Cases",))

fig.update_layout(title="Confirmed Cases Holt's Winter Model Prediction",
                  xaxis_title="Date",yaxis_title="Confirmed
Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()


holt_winter_new_prediction=[]

for i in range(1,18):

    holt_winter_new_prediction.append(es.forecast((len(valid)+i))[-1])

model_predictions["Holt's Winter Model Prediction"]=holt_winter_new_prediction

model_predictions.head()


model_train=datewise.iloc[:int(datewise.shape[0]*0.95)]

valid=datewise.iloc[int(datewise.shape[0]*0.95):]

y_pred=valid.copy()


model_ar= auto_arima(model_train["Confirmed"],trace=True, error_action='ignore',
start_p=0,start_q=0,max_p=4,max_q=0,

                    suppress_warnings=True,stepwise=False,seasonal=False)

model_ar.fit(model_train["Confirmed"])

prediction_ar=model_ar.predict(len(valid))

y_pred["AR Model Prediction"]=prediction_ar


model_scores.append(np.sqrt(mean_squared_error(y_pred["Confirmed"],y_pred["AR
Model Prediction"])))

```

```

print("Root Mean Square Error for AR Model:
",np.sqrt(mean_squared_error(y_pred["Confirmed"],y_pred["AR Model Prediction"])))

fig=go.Figure()

fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
                        mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],
                        mode='lines+markers',name="Validation Data for Confirmed Cases",))

fig.add_trace(go.Scatter(x=valid.index, y=y_pred["AR Model Prediction"],
                        mode='lines+markers',name="Prediction of Confirmed Cases",))

fig.update_layout(title="Confirmed Cases AR Model Prediction",
                  xaxis_title="Date",yaxis_title="Confirmed
Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

AR_model_new_prediction=[]

for i in range(1,18):

    AR_model_new_prediction.append(model_ar.predict(len(valid)+i)[-1])

model_predictions["AR Model Prediction"]=AR_model_new_prediction

model_predictions.head()

model_train=datewise.iloc[:int(datewise.shape[0]*0.95)]

valid=datewise.iloc[int(datewise.shape[0]*0.95):]

y_pred=valid.copy()

model_sarima= auto_arima(model_train["Confirmed"],trace=True, error_action='ignore',
                        start_p=0,start_q=0,max_p=2,max_q=2,m=7,

```



```

        suppress_warnings=True,stepwise=True,seasonal=True)

model_sarima.fit(model_train["Confirmed"])

model_ma= auto_arima(model_train["Confirmed"],trace=True, error_action='ignore',
start_p=0,start_q=0,max_p=0,max_q=2,

suppress_warnings=True,stepwise=False,seasonal=False)

model_ma.fit(model_train["Confirmed"])

prediction_ma=model_ma.predict(len(valid))

y_pred["MA Model Prediction"]=prediction_ma

model_scores.append(np.sqrt(mean_squared_error(valid["Confirmed"],prediction_ma)))

print("Root Mean Square Error for MA Model:
",np.sqrt(mean_squared_error(valid["Confirmed"],prediction_ma)))

fig=go.Figure()

fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
                        mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],
                        mode='lines+markers',name="Validation Data for Confirmed Cases",))

fig.add_trace(go.Scatter(x=valid.index, y=y_pred["MA Model Prediction"],
                        mode='lines+markers',name="Prediction for Confirmed Cases",))

fig.update_layout(title="Confirmed Cases MA Model Prediction",
                  xaxis_title="Date",yaxis_title="Confirmed
Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

MA_model_new_prediction=[]

```

```

for i in range(1,18):

    MA_model_new_prediction.append(model_ma.predict(len(valid)+i)[-1])

model_predictions["MA Model Prediction"]=MA_model_new_prediction

model_predictions.head()


model_train=datewise.iloc[:int(datewise.shape[0]*0.95)]

valid=datewise.iloc[int(datewise.shape[0]*0.95):]

y_pred=valid.copy()


model_arima= auto_arima(model_train["Confirmed"],trace=True, error_action='ignore',
start_p=1,start_q=1,max_p=3,max_q=3,

                        suppress_warnings=True,stepwise=False,seasonal=False)

model_arima.fit(model_train["Confirmed"])


prediction_arima=model_arima.predict(len(valid))

y_pred["ARIMA Model Prediction"]=prediction_arima


model_scores.append(np.sqrt(mean_squared_error(valid["Confirmed"],prediction_arima)))

print("Root Mean Square Error for ARIMA Model:
",np.sqrt(mean_squared_error(valid["Confirmed"],prediction_arima)))


fig=go.Figure()

fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],
                        mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],
                        mode='lines+markers',name="Validation Data for Confirmed Cases",))

fig.add_trace(go.Scatter(x=valid.index, y=y_pred["ARIMA Model Prediction"],

```

```

        mode='lines+markers',name="Prediction for Confirmed Cases",))

fig.update_layout(title="Confirmed Cases ARIMA Model Prediction",

                  xaxis_title="Date",yaxis_title="Confirmed
Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

```

```

ARIMA_model_new_prediction=[]

for i in range(1,18):

    ARIMA_model_new_prediction.append(model_arima.predict(len(valid)+i)[-1])

model_predictions["ARIMA Model Prediction"]=ARIMA_model_new_prediction

model_predictions.head()

```

```

prediction_sarima=model_sarima.predict(len(valid))

y_pred["SARIMA Model Prediction"]=prediction_sarima

```

```

model_scores.append(np.sqrt(mean_squared_error(y_pred["Confirmed"],y_pred["SARIMA
Model Prediction"])))

```

```

print("Root Mean Square Error for SARIMA Model:
",np.sqrt(mean_squared_error(y_pred["Confirmed"],y_pred["SARIMA Model
Prediction"])))

```

```

fig=go.Figure()

fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Confirmed"],

                        mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],

```

```

        mode='lines+markers',name="Validation Data for Confirmed Cases",))

fig.add_trace(go.Scatter(x=valid.index, y=y_pred["SARIMA Model Prediction"],

        mode='lines+markers',name="Prediction for Confirmed Cases",))

fig.update_layout(title="Confirmed Cases SARIMA Model Prediction",

        xaxis_title="Date",yaxis_title="Confirmed
Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

```

```

SARIMA_model_new_prediction=[]

for i in range(1,18):

    SARIMA_model_new_prediction.append(model_sarima.predict(len(valid)+i)[-1])

model_predictions["SARIMA Model Prediction"]=SARIMA_model_new_prediction

model_predictions.head()

```

```

model_names=["Linear Regression","Polynomial Regression","Support Vector Machine
Regressor","Holt's Linear","Holt's Winter Model",

        "Auto Regressive Model (AR)","Moving Average Model (MA)","ARIMA
Model","SARIMA Model"]

model_summary=pd.DataFrame(zip(model_names,model_scores),columns=["Model
Name","Root Mean Squared Error"]).sort_values(["Root Mean Squared Error"])

model_summary

```

Chapter 6: Output Screens

Growth of different types of cases in India

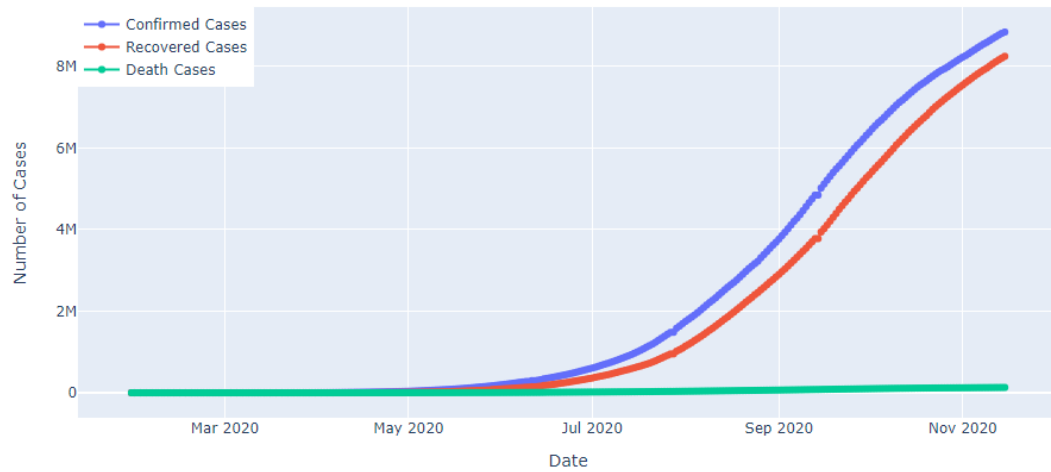


Fig 6.1.Growth of different types of cases in India

Confirmed Cases Linear Regression Prediction

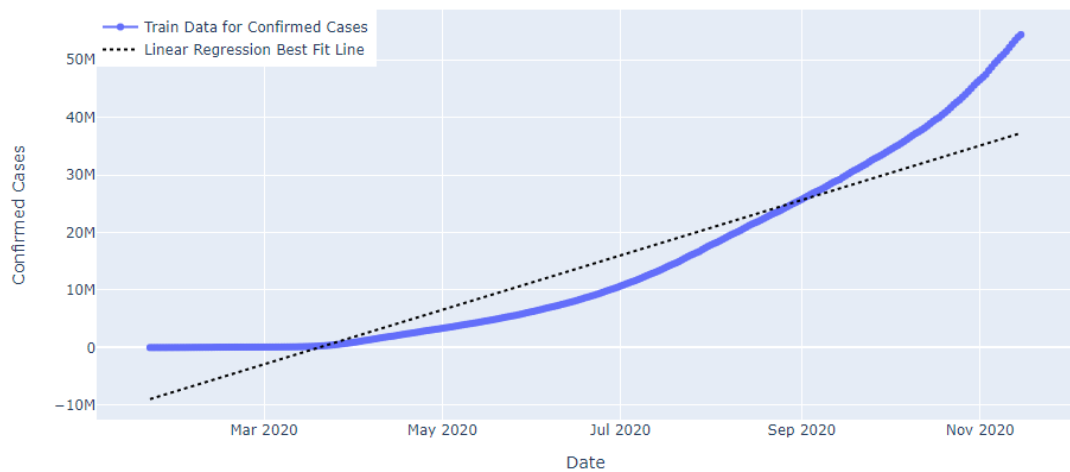


Fig 6.2 Confirmed cases Linear Regression Prediction

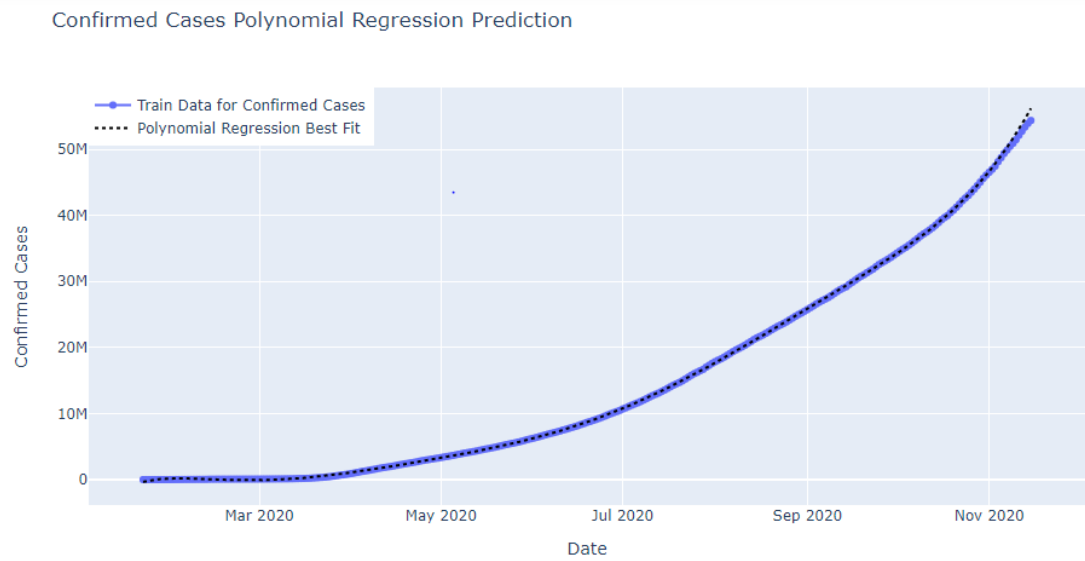


Fig 6.3. Polynomial Regression Prediction for confirmed cases

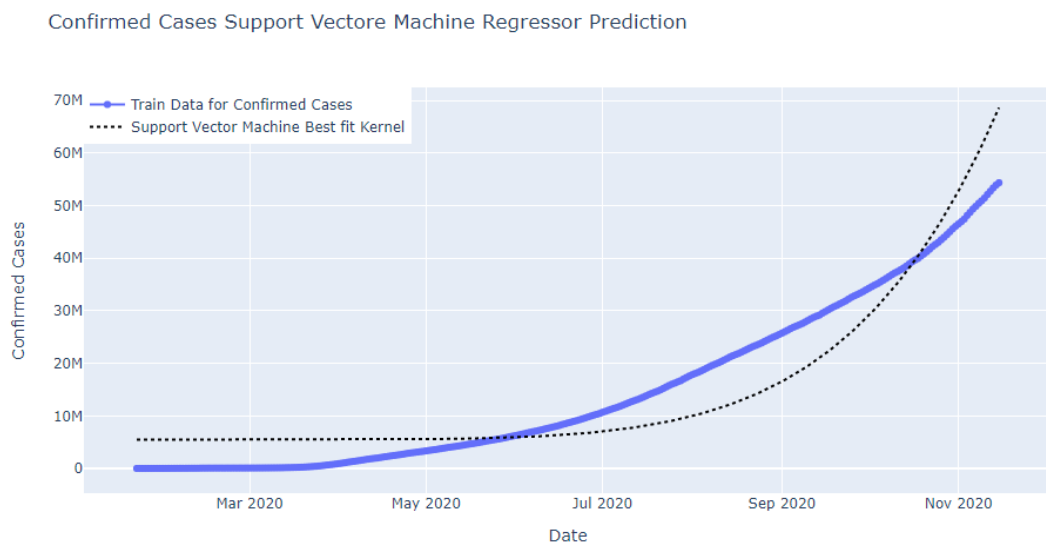


Fig 6.4. SVM regressor Prediction for confirmed cases

Confirmed Cases Holt's Linear Model Prediction

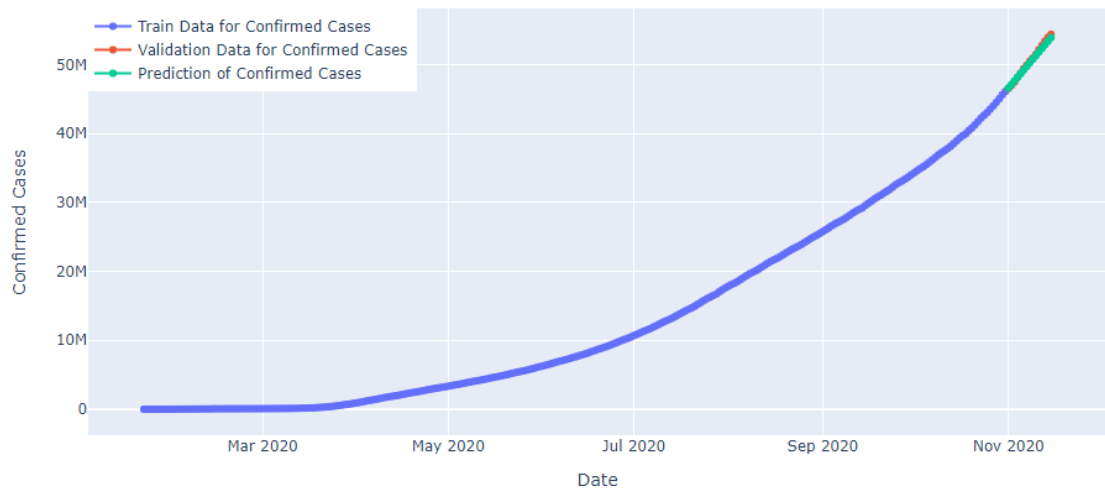


Fig 6.5 Holts Linear Model Prediction for confirmed cases

Confirmed Cases Holt's Winter Model Prediction

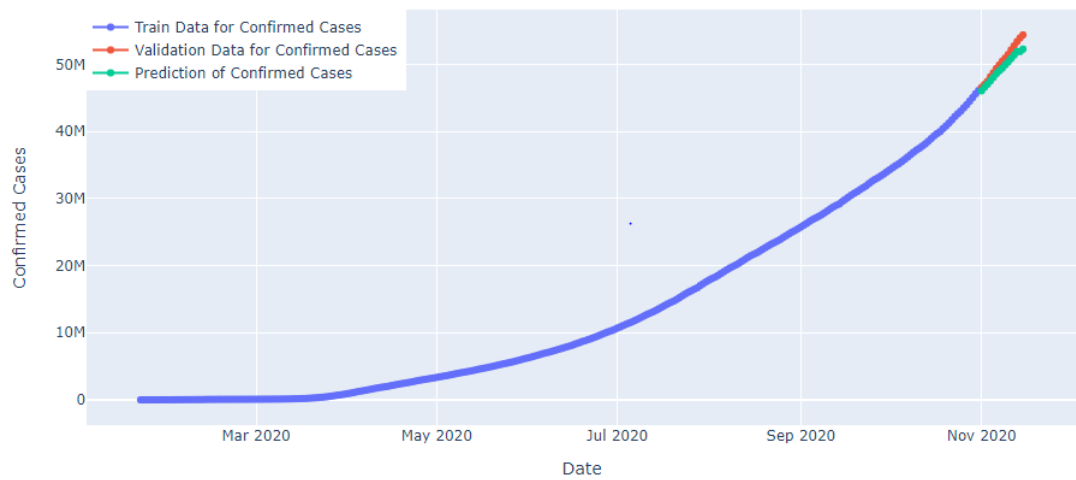


Fig 6.6. Holt's Winter model prediction for confirmed cases

Confirmed Cases AR Model Prediction

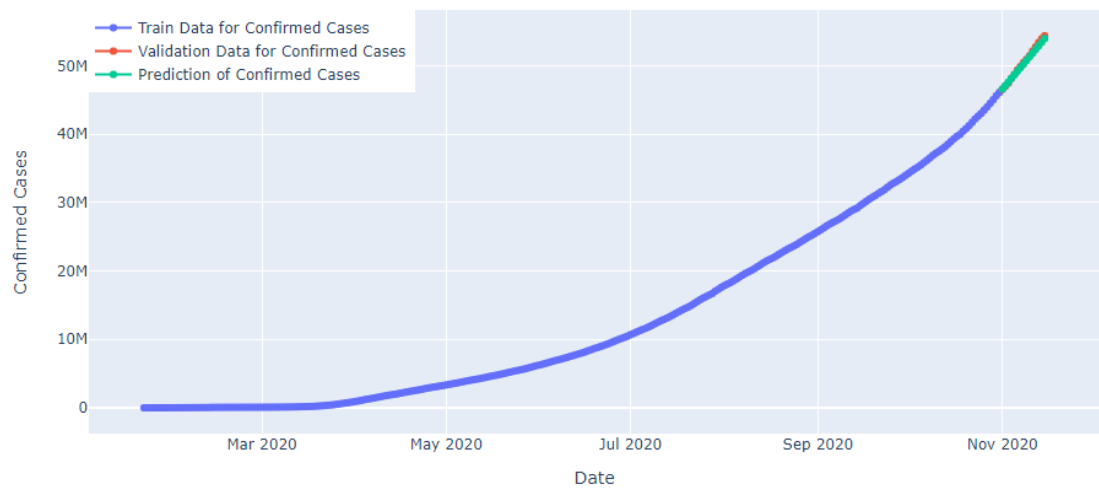


Fig 6.7. AR model prediction for confirmed cases

Confirmed Cases SARIMA Model Prediction

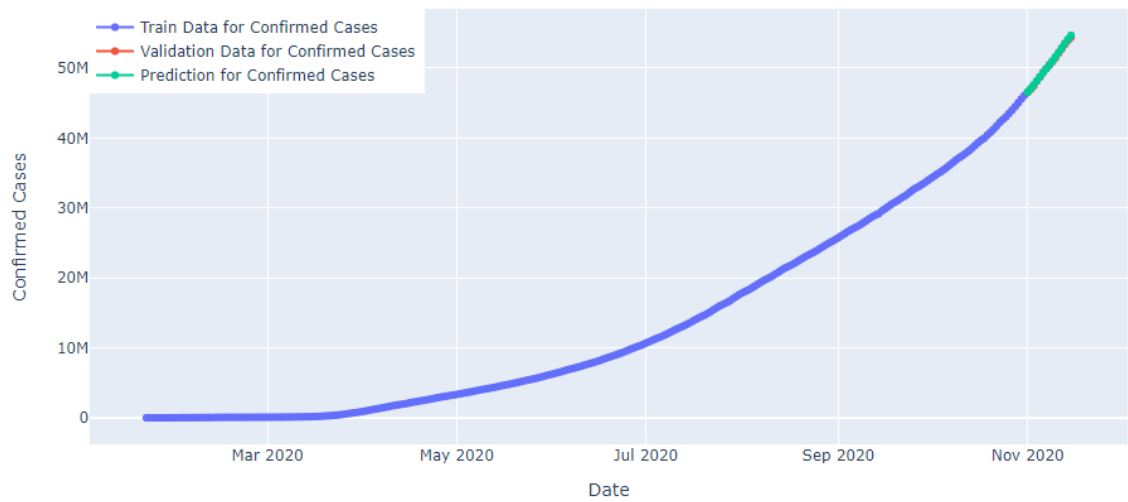


Fig 6.8. SARIMA model Prediction

Chapter 7: System Testing

In this project the model evaluation part is very important as by the means of it we can identify which model can the best fit for the problem.

Here the models are evaluated on the basis of its root mean square error(rmse).

The root-mean-square variance (RMSD) or root-mean-square error (RMSE) is a commonly used calculation of the differences expected by the model or estimator between values (sample or population values) and the values observed.

According to the rmse values of all the models tested in the project the one with the least rmse value was SARIMA model. So it can be considered as the best fit model for this problem.

Chapter 8: Conclusion

Conclusion

It is concluded that machine learning models can be used to forecast the spread of infectious diseases like Covid-19. In the project we used various algorithms to forecast the rise of confirmed cases. It was observed among all the algorithms used, SARIMA had the least rmse so it was considered the best fit model for the data that was available.

Limitations

It is a new virus so only a year worth of dataset is available. Generally, the more data we have the better accuracy we get and we have to keep updating the data.

Scope for future work

It can be implemented such that it can update its graphs or predictions according to the real time values.

Chapter 9: Bibliography

1. G. Bontempi, S. B. Taieb and Y.-A. Le Borgne, "Machine learning strategies for time series forecasting", *Proc. Eur. Bus. Intell. Summer School*, pp. 62-77, 2012.
2. F. Petropoulos and S. Makridakis, "Forecasting the novel coronavirus COVID-19", *PLoS ONE*, vol. 15, no. 3, Mar. 2020.
3. G. Grasselli, A. Pesenti and M. Cecconi, "Critical care utilization for the COVID-19 outbreak in lombardy italy: Early experience and forecast during an emergency response", *JAMA*, vol. 323, no. 16, pp. 1545, Apr. 2020.
4. S. Makridakis, E. Spiliotis and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward", *PLoS ONE*, vol. 13, Mar. 2018.


Chapter 10: Annexures



Document Information

Analyzed document	Ritika Minor Report.pdf (D90627390)
Submitted	12/27/2020 4:57:00 AM
Submitted by	Siddhanta Kumar Singh
Submitter email	sksingh.set@modyuniversity.ac.in
Similarity	7%
Analysis address	sksingh.set.modyun@analysis.urkund.com

Sources included in the report

W	URL: https://www.contentintelligence.net/en/ci/machine-learning-this-unknown-being Fetched: 12/27/2020 4:58:00 AM	  2
W	URL: https://www.preprints.org/manuscript/202004.0311/v1 Fetched: 12/27/2020 4:58:00 AM	  1