15CSE401/481 MLDM- CASE STUDY- FINAL REVIEW

COVID-19 DATA ANALYSIS

Group Number : 03

Group Members:

1. Aradhana J. -CB.EN.U4CSE17308
2. Arnab Datta -CB.EN.U4CSE17309
3. Ashwin B. -CB.EN.U4CSE17310
4. Niveth Saran V. J. -CB.EN.U4CSE17337

Problem Definition

This project is a drive to analyse and predict effects of the current pandemic in the coming days based on existing an d recently generated survey reports and extracted informati on using rich machine learning components, deep learning components such as ARIMA and
LSTM models to support organisations and institutions over anticipation of resource requirement in the later phases of the pandemic.

Objectives

1. Predict the number of COVID cases using Time Series Dataset
2. Use 2 different algorithms for prediction to get more insights.

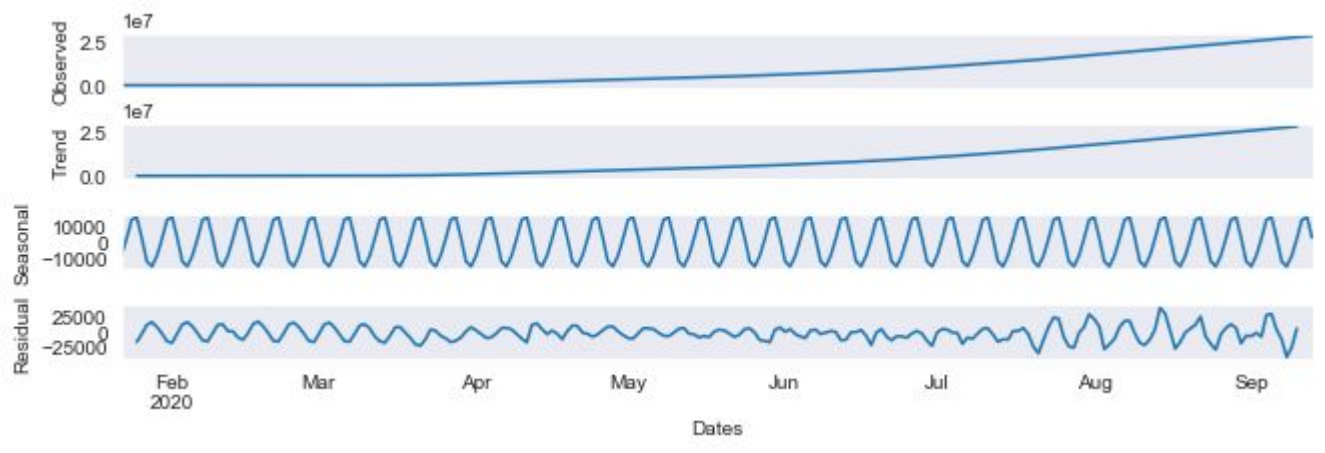
Dataset

- Dataset 1
  - Coronavirus India Dataset (https://github.com/imdevskp/covid-19-india-data/blob/master/nation_level_daily.csv)
  - This dataset contains daily data of Indian COVID deaths, recoveries and cases.
  - Attributes: 191 rows, 7 columns
- Dataset 2
  - Novel-Coronavirus-Dataset(https://github.com/Ashwin1999/COVID-19-Data-Mining/tree/master/COVID-Time%20Series%20Data/COVID-Time%20Series%20Data%20-%20Refined)
  - This dataset contains daily data of Global COVID deaths, recoveries and cases during the time period of 22nd Jan 2020 - 14th Sept 2020
  - Attributes: (Confirmed: 3606 rows, 240 columns; Deaths: 3606 rows, 240 columns, Recovered: 253 rows, 240 columns)

EDA- Outline

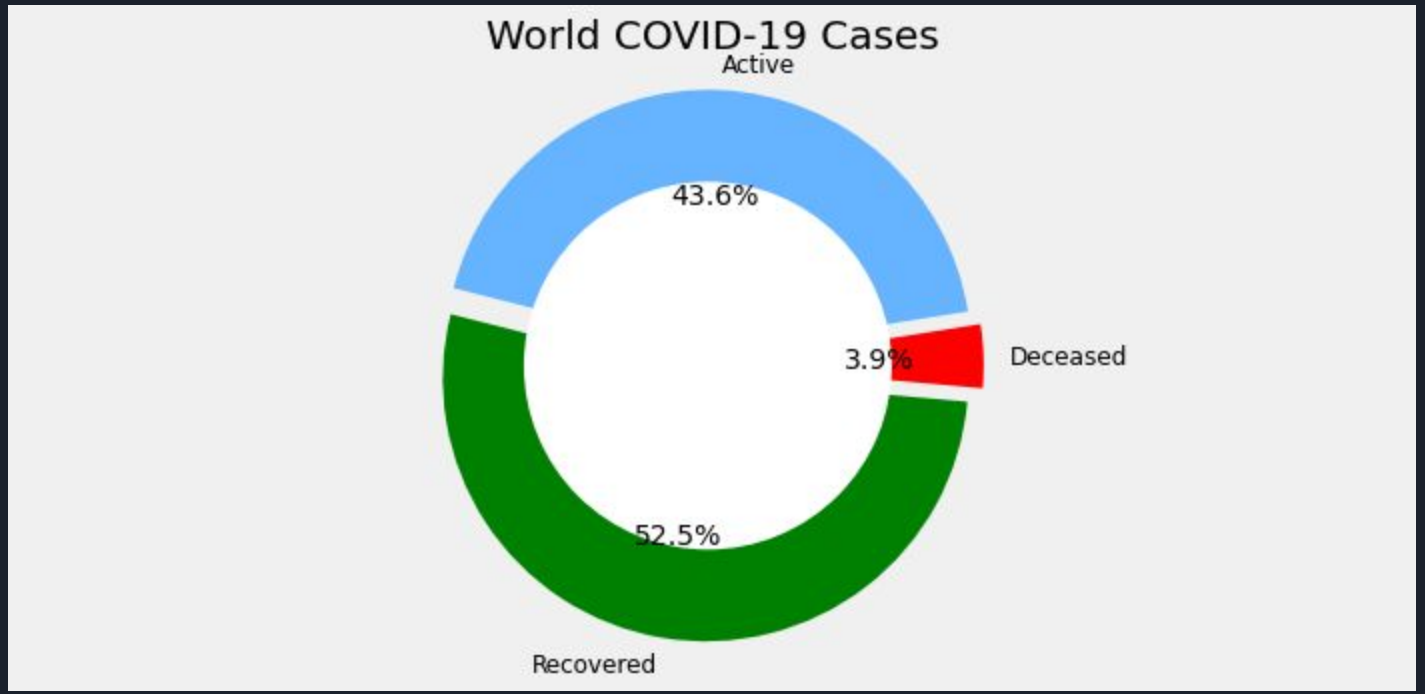
1. Analysis on Global COVID Data
    - Removing NA Values from table, Extracting proper data from noisy data
    - Converting the format of the date column
    - Perform AD Fuller tests and Seasonal Decompose
2. Analysis on Indian COVID Data
    - Removing NA Values from table, Extracting proper data from noisy data
    - Converting the format of the date column
    - Perform AD Fuller tests and Seasonal Decompose

ETS Decomposition



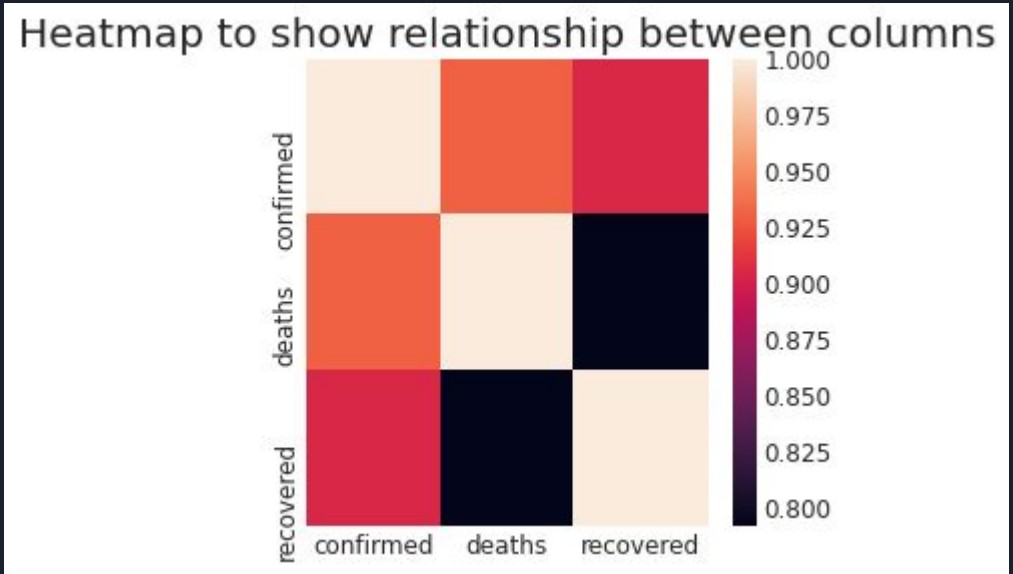
- Decomposed data shows an increased trend in cases
- Presence of seasonality in the dataset is also observed
- Some residual errors/noise exist in the data

EDA on Global COVID Data



Visualisation of the active,recovered and death cases on the global data

EDA on Global COVID Data



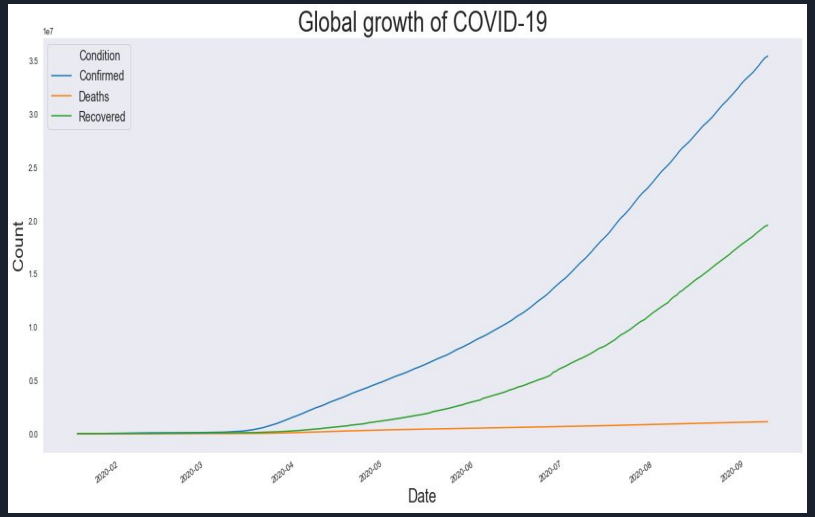
Correlation map shows that confirmed cases are positively correlated with the death cases,whereas the a similar correlation is present with the confirmed cases and the recovered cases as well

COVID-19 Global growth

Global confirmed cases of COVID-19 is at the 35 million range.

Global recovery from COVID-19 is at 20 million.

Global COVID-19 related death is somewhere around 1.2 million.

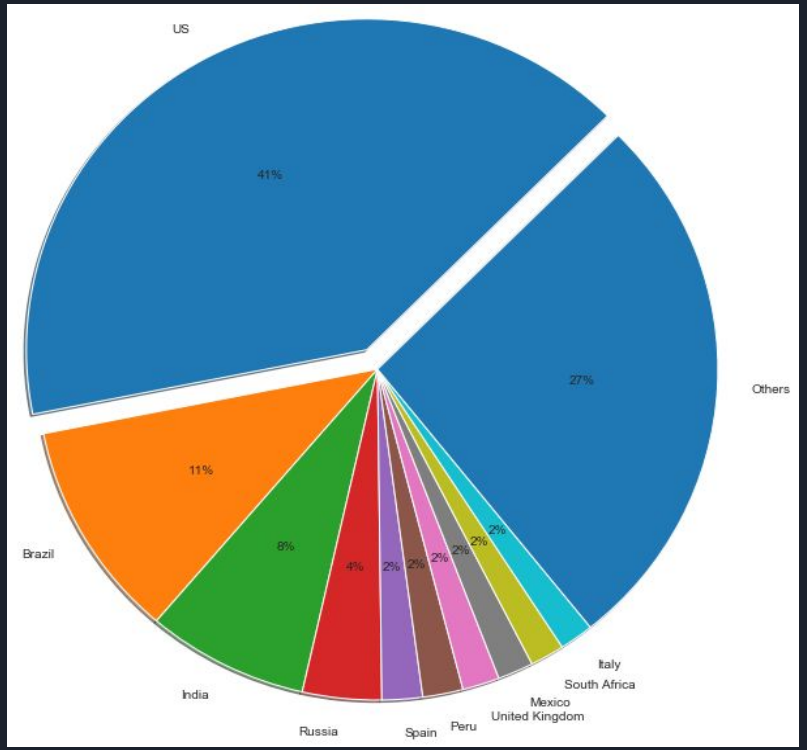


Country Wise report - Confirmed cases

US has the highest number of confirmed cases.

Around 41% of total confirmed COVID-19 cases in the world is from US.

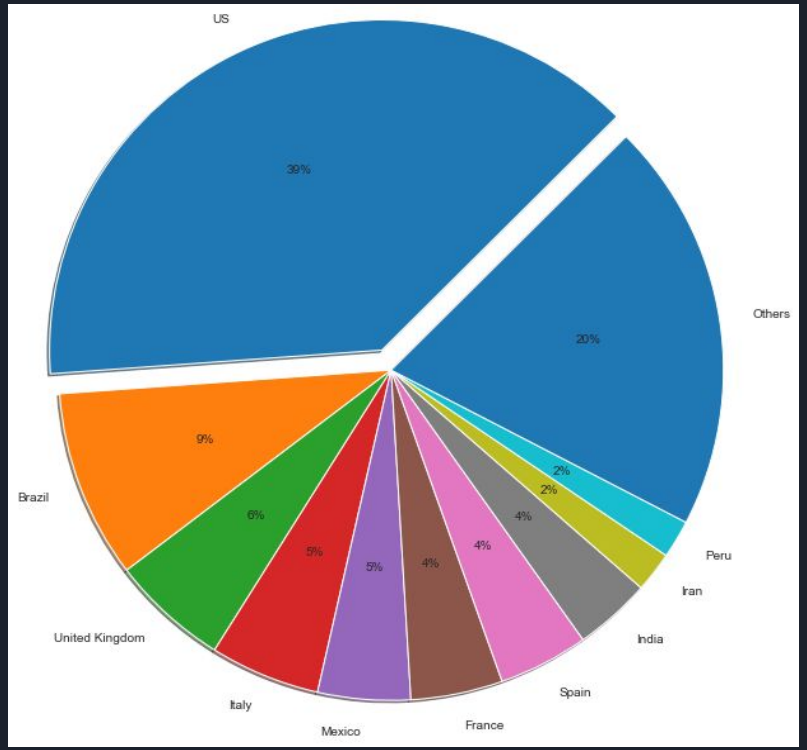
It is followed by Brazil, India and Russia.



Country Wise report - Deaths

Again US has the highest number of COVID-19 related death.

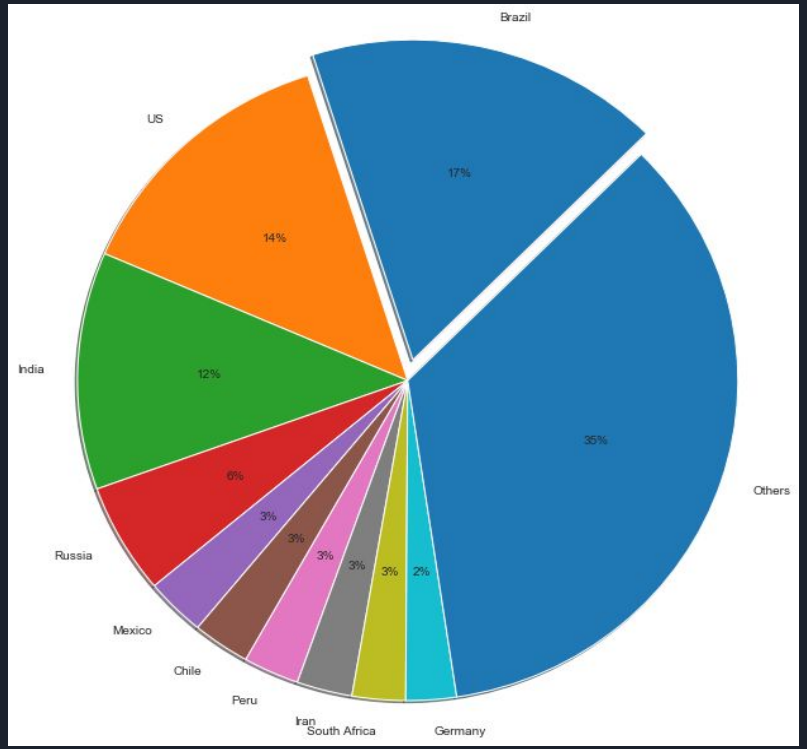
It is followed by Brazil, UK and Italy.



Country Wise report - Recovery

17% of all COVID-19 recovery is from Brazil.

Other countries with high recovery are US, India and Russia.



Global Mortality and Recovery Rate

Since the dataset used over here has the confirmed, deaths and recovered features, we can derive two other features from them.

They are:

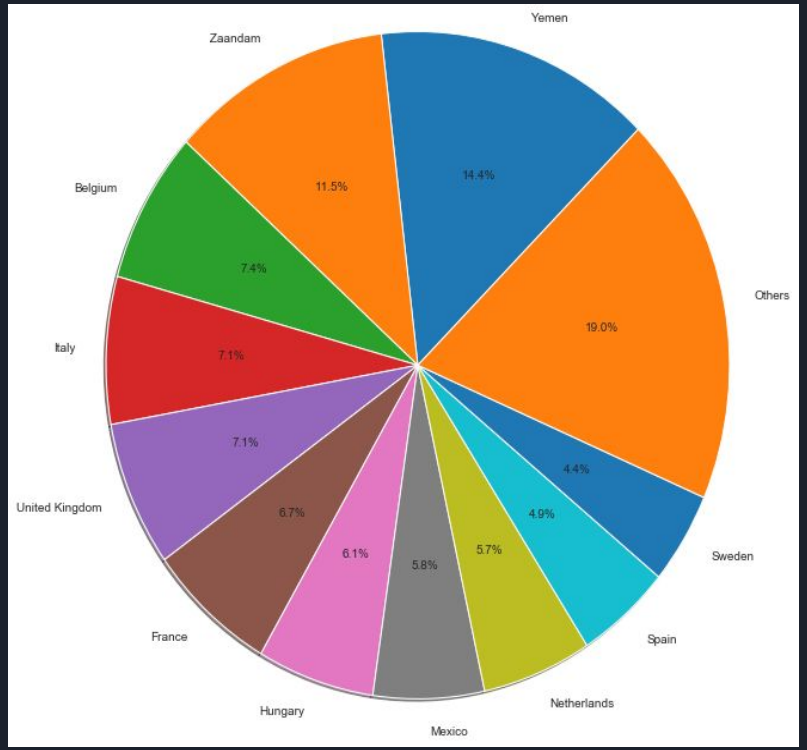
1. Mortality rate
2. Recovery rate

```
combined_countries['Mortality_rate'] = combined_countries['Deaths']/combined_countries['Confirmed']
combined_countries['Recovery_rate'] = combined_countries['Recovered']/combined_countries['Confirmed']

combined_countries.head()
```

	Country	Confirmed	Deaths	Recovered	Mortality_rate	Recovery_rate
0	Afghanistan	3745342	114595	2139148	0.030597	0.571149
1	Albania	583139	17360	329905	0.029770	0.565740
2	Algeria	3088876	145505	2089806	0.047106	0.676559
3	Andorra	145841	7952	111054	0.054525	0.761473
4	Angola	125569	5483	45985	0.043665	0.366213

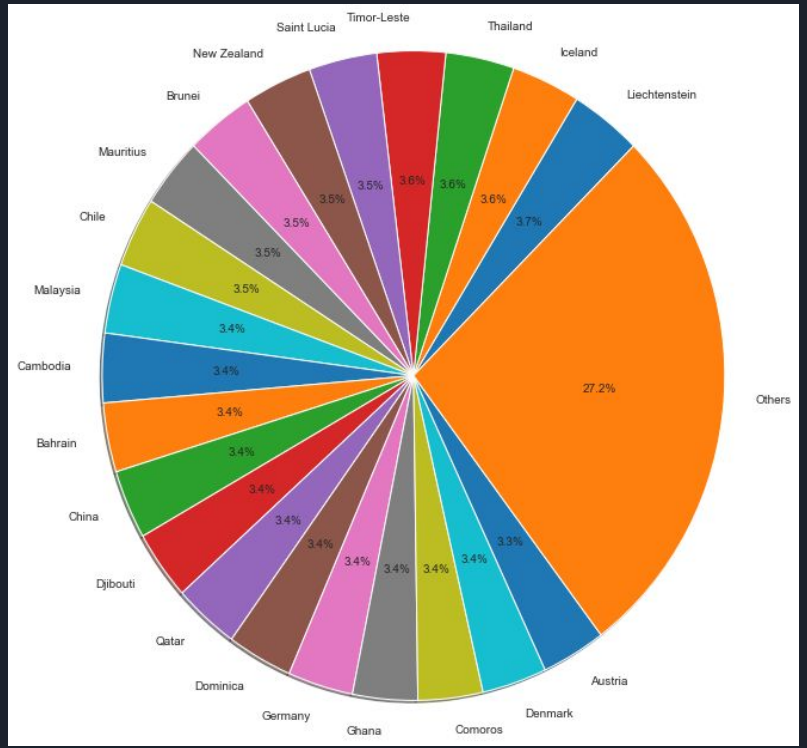
Global Mortality Rate

From the chart, Yemen has the highest global COVID-19 mortality rate.



Global Recovery Rate

Liechtenstein has the highest global COVID-19 recovery rate, followed by Iceland and Thailand.

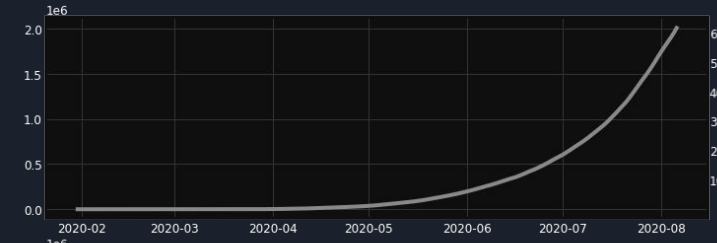


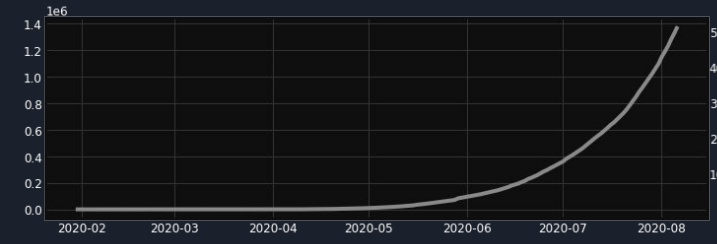
EDA on Indian COVID Data

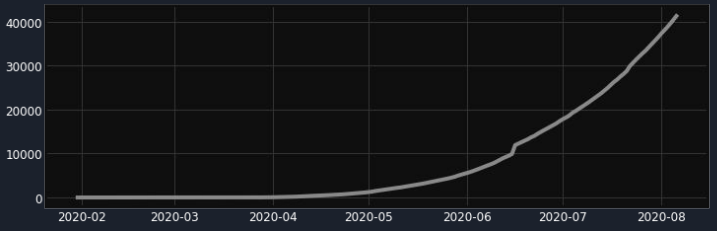
Cases

Deaths

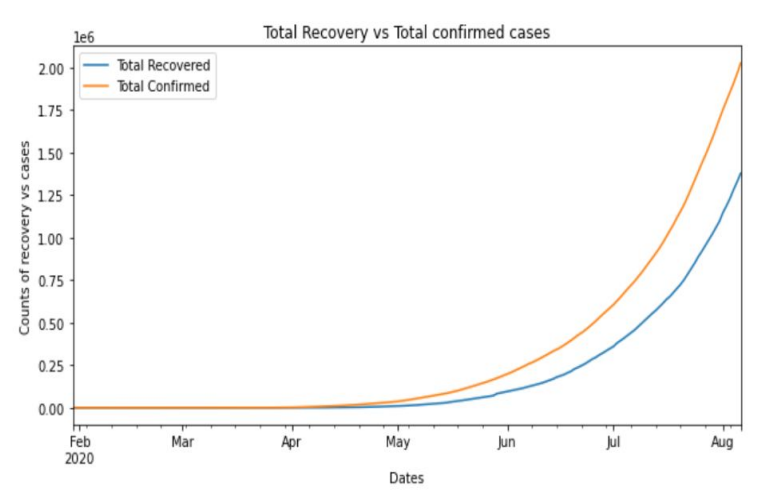
Recoveries

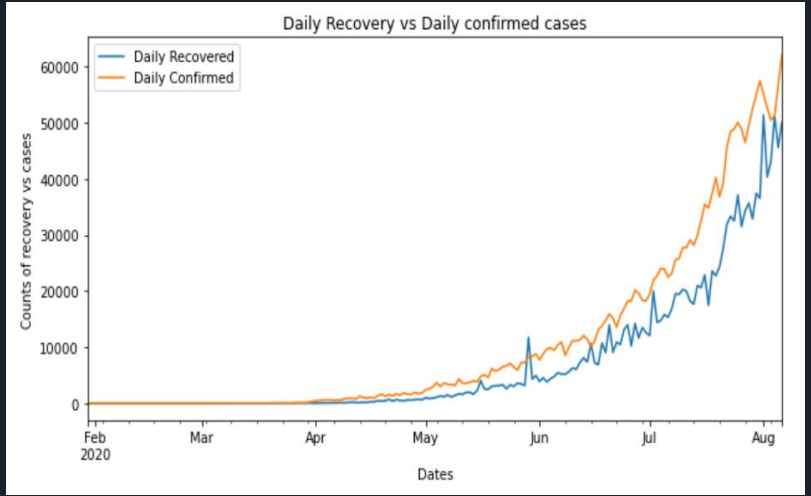




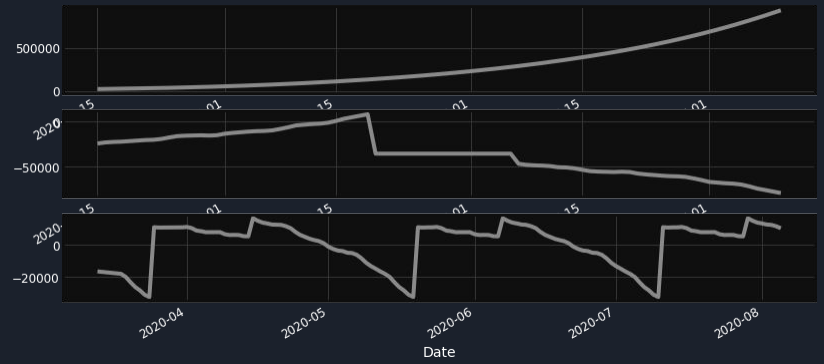


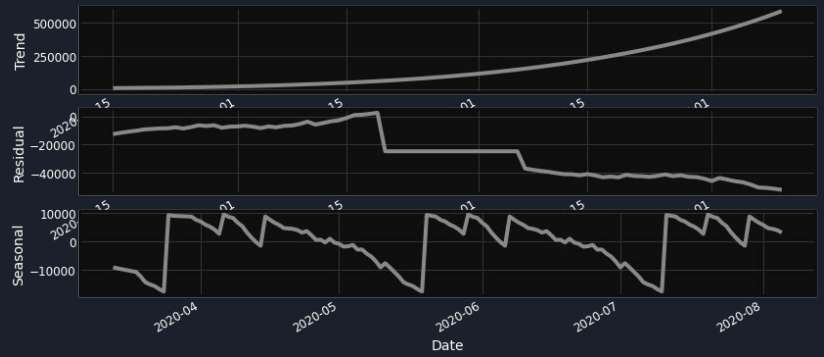
EDA on Indian COVID Data





EDA on Indian COVID Data





EDA on Indian COVID Data

```
indiadaily=pd.read_csv('https://raw.githubusercontent.com/Ashwin1999/COVID-19-Data-Mining/Arnab/COVID-DataTimeSeries(India)/nation_level_daily.csv')
indiadaily['Date'] = indiadaily['Date'].str.slice(0,6) +' 2020'
indiadaily['Date']= indiadaily['Date'].astype(str).apply(lambda x: datetime.datetime.strptime(x,'%d %b %Y'))
indiadaily['Date']= pd.to_datetime(indiadaily['Date'])
indiadaily.rename(columns={'Daily Confirmed':'casesdaily',
                           'Total Confirmed':'cases',
                           'Daily Recovered':'recoverydaily',
                           'Total Recovered':'recovery',
                           'Daily Deceased' :'deathdaily',
                           'Total Deceased':'death'},inplace=True)
print(indiadaily.dtypes)
indiadaily.tail()
```

EDA on Indian COVID Data

```
[ ]  1  result = adfuller(indiacovidfinal['cases'])
     2  print('ADF Statistic: %f' % result[0])
     3  print('p-value: %f' % result[1])
     4  for key, value in result[4].items():
     5      print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: 3.495026
p-value: 1.000000
        1%: -3.483
        5%: -2.885
        10%: -2.579
```

```
     1  result = adfuller(indiacovidfinal['death'])
     2  print('ADF Statistic: %f' % result[0])
     3  print('p-value: %f' % result[1])
     4  for key, value in result[4].items():
     5      print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: 18.247325
p-value: 1.000000
        1%: -3.479
        5%: -2.883
        10%: -2.578
```

EDA on Indian COVID Data

```
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(indiacovidfinal)
```

```
# split into train and test sets
train_size = int(len(dataset) * 0.80)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset
),:]
print(len(train), len(test))
print(train.shape)
print(test.shape)
```

```
110 28
(110, 7)
(28, 7)
```

```
trainX, trainY = train[:,0:6],train[:,6]
testX, testY = test[:,0:6],test[:,6]
```

Training Time Series Models Outline

We have used ARIMA and LSTM to train the models. Both models have been applied to the two datasets so that we can compare the results.

Contents

1. LSTM on Global Data
2. ARIMA on Global Data
3. LSTM on Indian Data
4. ARIMA on India Data

LSTM on Global Data

LSTMs are a type of recurrent neural network that can learn the context required to make predictions in time series forecasting problems, rather than having this context pre-specified and fixed.

The main difference between LSTM and a simple RNN is that they have the ability to remember long-term trends which is something the simple RNN tends to forget.

LSTM also gets rid of problems like vanishing/exploding gradients that is frequent in simple RNN.

It is used here for forecasting the number of confirmed COVID-19 cases.

The model summary is given in the next slide...

LSTM on Global Data - Model summary

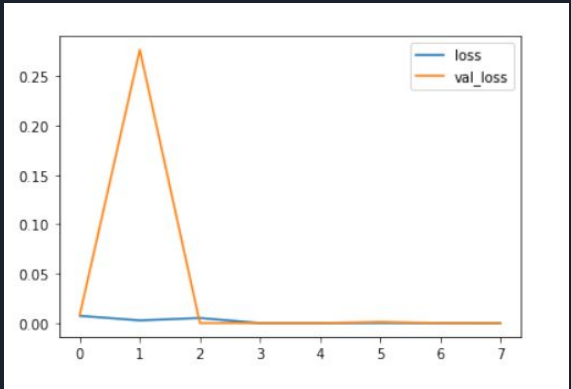
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

# Bulding the model
model = Sequential()
model.add(LSTM(256, activation='relu', input_shape=(BS, n_features)))
model.add(Dense(1))

# Compiling the model
model.compile(optimizer=Adam(), loss='mse')

# Training
EPOCHS = 25
earlyStop = EarlyStopping(monitor='val_loss', patience=4)
model.fit_generator(generator, epochs=EPOCHS, validation_data=valGenerator, callbacks=[earlyStop])
```

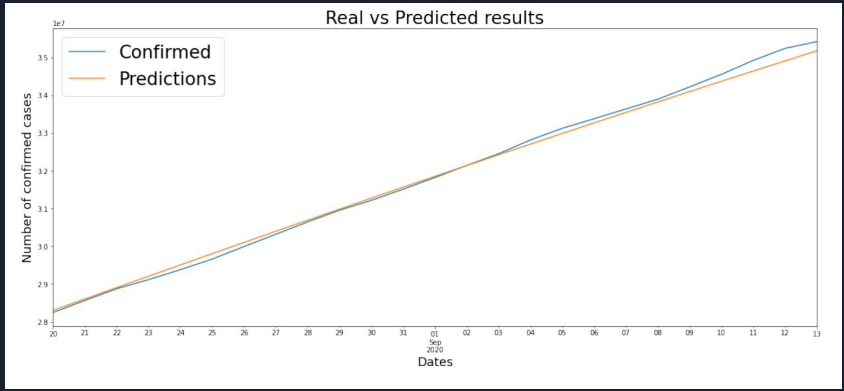
Code

Model Summary

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_6 (LSTM)                (None, 256)               264192
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 257
=================================================================
Total params: 264,449
Trainable params: 264,449
Non-trainable params: 0
```

LSTM on Global Data - Post training

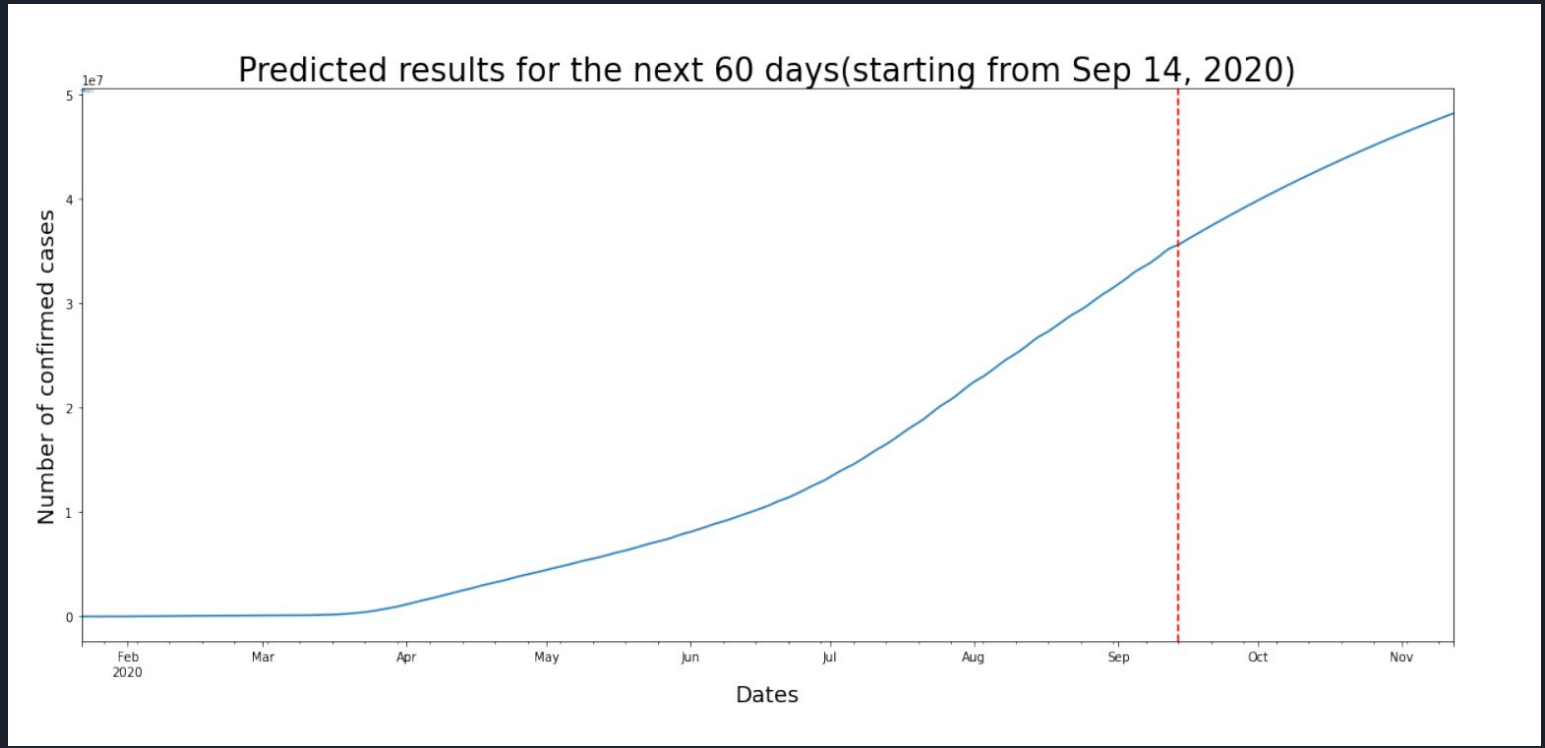





```
from sklearn.metrics import r2_score, mean_squared_error

mse = mean_squared_error(test.Confirmed, test.Predictions)
# length = len(test)
rmse = np.sqrt(mse)

r2s = r2_score(test.Confirmed, test.Predictions)

print(f'RMSE = {rmse}; r2 score = {r2s}')
```

```
RMSE = 131797.31945756712; r2 score = 0.9963465838706869
```

LSTM on Global Data - Forecasting for 60 days



Algorithm - ARIMA,Global covid 19 data

- ARIMA, short for 'Auto Regressive Integrated Moving Average' is actually a class of models that 'explains' a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values.
- SARIMAX which is an extended version of ARIMA that account to seasonality of data for a better performance in prediction is used in this forecasting.
- The model training methodology in our case study is given below:

```
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX

confirmed_to_fit = confirmed_m.groupby('Dates').sum()[confirmed_m.groupby('Dates').sum()['Count']>=0]

model = SARIMAX(confirmed_to_fit['Count'],order=(1,1,0), seasonal_order=(1,1,1,12))
results = model.fit()
print(results.summary())
```

SARIMAX model for the optimum values of p,q,d

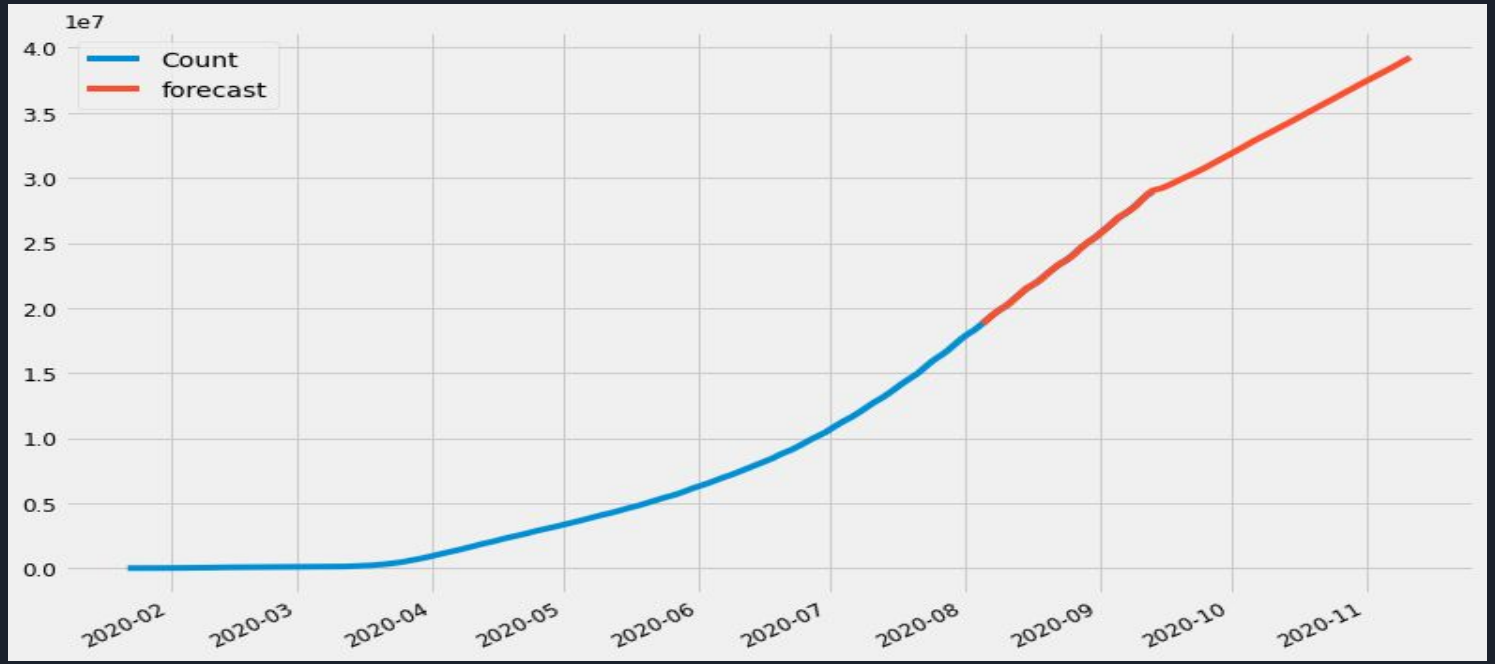
Statespace Model Results

Dep. Variable:	Count	No. Observations:	236
Model:	SARIMAX(1, 1, 0)x(1, 1, 1, 12)	Log Likelihood	-2537.654
Date:	Tue, 03 Nov 2020	AIC	5083.308
Time:	11:41:25	BIC	5096.937
Sample:	01-22-2020 - 09-13-2020	HQIC	5088.810
Covariance Type:	opg		

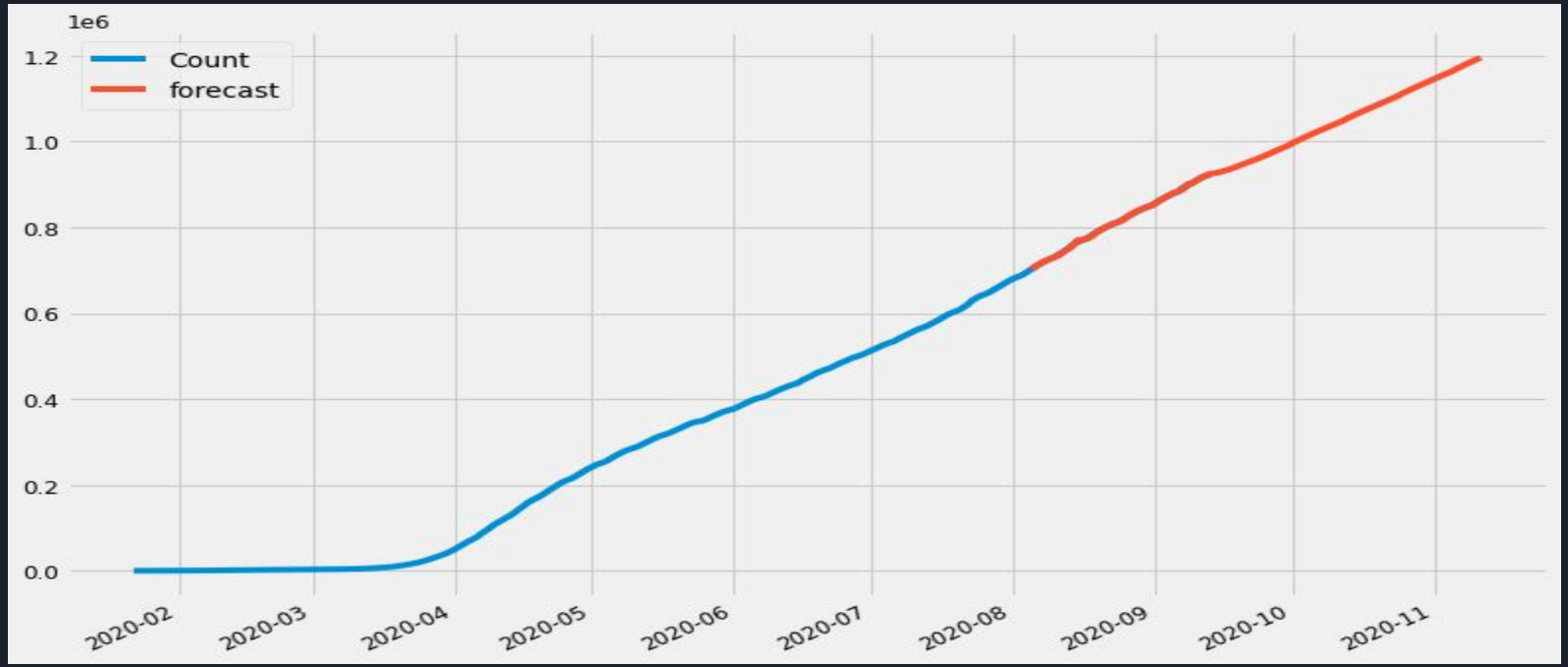
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9795	0.024	41.473	0.000	0.933	1.026
ar.S.L12	-0.2433	0.120	-2.032	0.042	-0.478	-0.009
ma.S.L12	-0.8403	0.121	-6.958	0.000	-1.077	-0.604
sigma2	6.321e+08	1.93e-11	3.28e+19	0.000	6.32e+08	6.32e+08

Ljung-Box (Q):	734.23	Jarque-Bera (JB):	846.82
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	21.87	Skew:	-1.12
Prob(H) (two-sided):	0.00	Kurtosis:	12.28

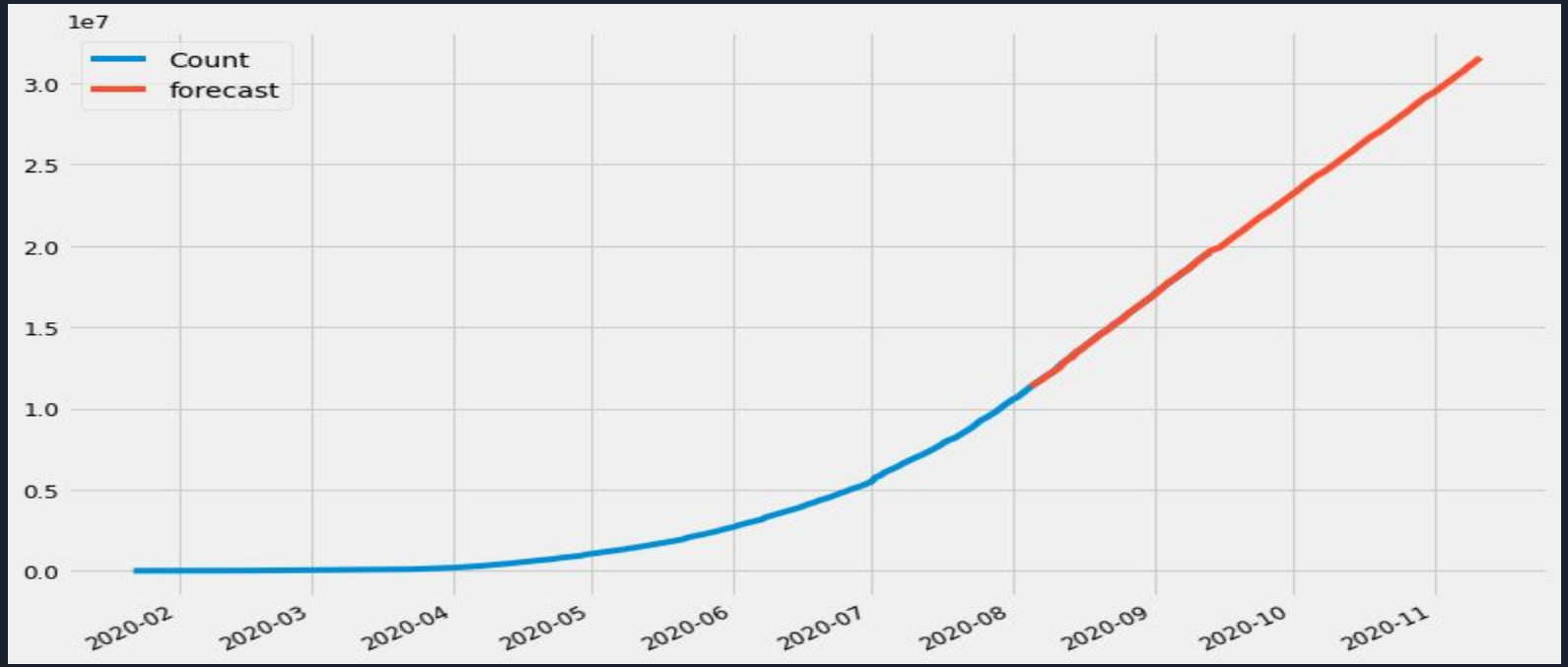
Prediction of confirmed covid 19 cases for the next 60 days



Prediction of deceased covid 19 cases for the next 60 days



Prediction of recovered covid 19 cases for the next 60 days



R2 and RMSE errors for the predictions

```
from sklearn.metrics import r2_score, mean_squared_error
test1 = confirmed_to_fit.iloc[196:, :]
mse = mean_squared_error(test1['Count'],test1['forecast'])
rmse = np.sqrt(mse)
r2s = r2_score(test1['Count'],test1['forecast'])
print(f'Confirmed cases RMSE = {rmse}; r2 score = {r2s}')
```

```
Confirmed cases RMSE = 35000.21094646491; r2 score = 0.999862857245256
```

```
test2 = deaths_to_fit.iloc[196:, :]
mse = mean_squared_error(test2['Count'],test2['forecast'])
rmse = np.sqrt(mse)
r2s = r2_score(test2['Count'],test2['forecast'])
print(f'Deceased cases RMSE = {rmse}; r2 score = {r2s}')
```

```
Deceased cases RMSE = 1600.2622356703353; r2 score = 0.9993966239121435
```

```
test3 = recovered_to_fit.iloc[196:, :]
mse = mean_squared_error(test3['Count'],test3['forecast'])
rmse = np.sqrt(mse)
r2s = r2_score(test3['Count'],test3['forecast'])
print(f'Recovered cases RMSE = {rmse}; r2 score = {r2s}')
```

```
Recovered cases RMSE = 47542.41949716056; r2 score = 0.9996259962066568
```

Mean Average Percentage Error

Confirmed cases

```
mape = np.mean(np.abs(confirmed_to_fit['forecast']-confirmed_to_fit['Count'])/np.abs(confirmed_to_fit['Count']))
mape*100
```

```
0.10637174121099298
```

The prediction has an error of .106% indicating that the model is very much accurate with an accuracy of 99.9%

Mean Average Percentage Error

Deceased cases

```
[63] mape = np.mean(np.abs(deaths_to_fit['forecast']-deaths_to_fit['Count'])/np.abs(deaths_to_fit['Count']))
     mape*100
```

```
0.13808435966875504
```

The prediction has an error of .13% indicating that the model is very much accurate with an accuracy of 99.86%

Recovered cases

```
mape = np.mean(np.abs(recovered_to_fit['forecast']-recovered_to_fit['Count'])/np.abs(recovered_to_fit['Count']))
mape*100
```

```
0.22739627662184567
```

The prediction has an error of .13% indicating that the model is very much accurate with an accuracy of 99.78%

Algorithm - LSTM, Data - Covid19 India

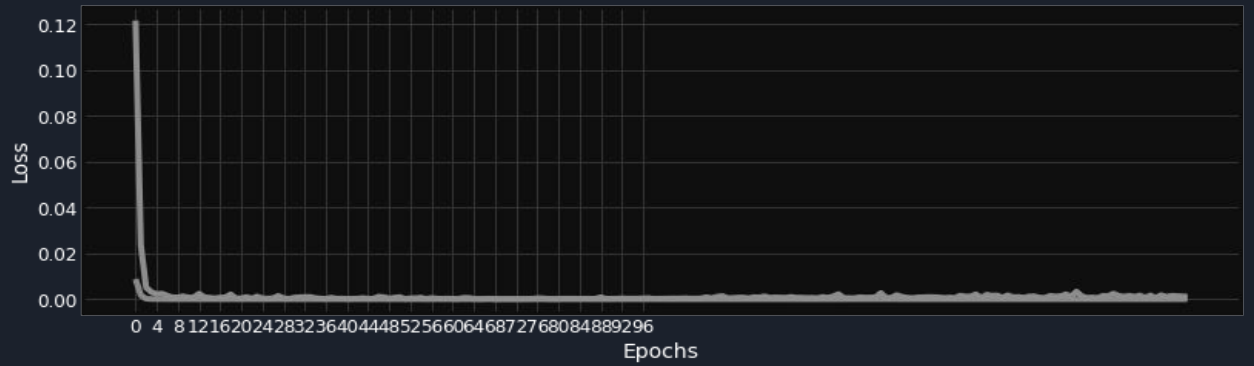
- LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.
- Model trained in our case study is given below

```
num_epochs=200
lstm_model = Sequential()
lstm_model.add(LSTM(58, activation='relu', input_shape=(n_input, n_features)))
lstm_model.add(Dense(1))
lstm_model.compile(optimizer='adam', loss='mse',metrics='mse')
lstm_model.summary()
```

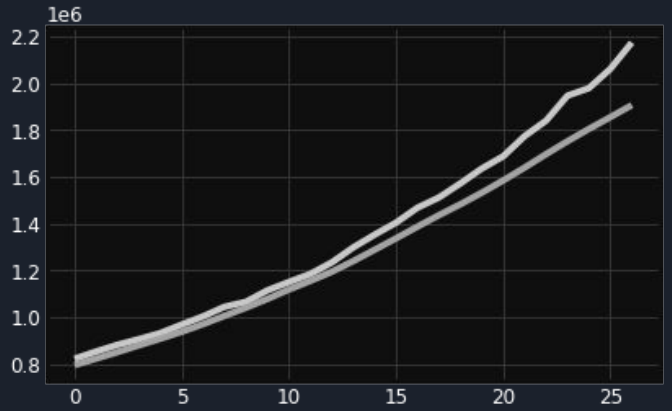
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 58)                15080
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 59
=================================================================
Total params: 15,139
Trainable params: 15,139
Non-trainable params: 0
_________________________________________________________________
```

Algorithm - LSTM, Data - Covid19 India

- The model in the previous slide provided the following loss graph.



- The comparison graph between actual and predicted values are provided below.



Algorithm - LSTM, Data - Covid19 India

- The final validation metric returned the below mentioned values.

```
rms = sqrt(mean_squared_error(inverseScaledActual[:,-1:],inverseScaledOutput[:,-1:]))
print('RMSE : ',rms)
```

```
RMSE :  103239.86560320178
```

```
mae = mean_absolute_error(inverseScaledActual[:,-1:],inverseScaledOutput[:,-1:])
print('MAE :' ,mae)
```

```
MAE : 80346.23067473483
```

```
r2score=r2_score(inverseScaledActual[:,-1:],inverseScaledOutput[:,-1:])
print('R2 Score: ',r2score )
```

```
R2 Score:  0.9079703241220058
```

Algorithm- ARIMA, Covid-19 India

- ARIMA, short for 'Auto Regressive Integrated Moving Average' is actually a class of models that 'explains' a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values.
- SARIMAX which is an extended version of ARIMA that account to seasonality of data for a better performance in prediction is used in this forecasting.
- The model training methodology in our case study is given below:

```
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

```
model = SARIMAX(total_confirmed_to_fit['Total Confirmed'],order=(1,1,0), seasonal_order=(1,1,1,12))
results = model.fit()
print(results.summary())
```

```
total_confirmed_to_fit['forecast'] = results.predict(start = 130, end= 190)
total_confirmed_to_fit[['Total Confirmed','forecast']].plot(figsize=(12,8));
```

Algorithm- ARIMA, Covid-19 India

The Details of the Sarimax model after fitting

SARIMAX Results

Dep. Variable:	Total Confirmed	No. Observations:	190
Model:	SARIMAX(1, 1, 0)x(1, 1, [1], 12)	Log Likelihood	-1520.482
Date:	Tue, 03 Nov 2020	AIC	3048.963
Time:	17:13:17	BIC	3061.668
Sample:	01-30-2020	HQIC	3054.116
	- 08-06-2020		
Covariance Type:	opg		

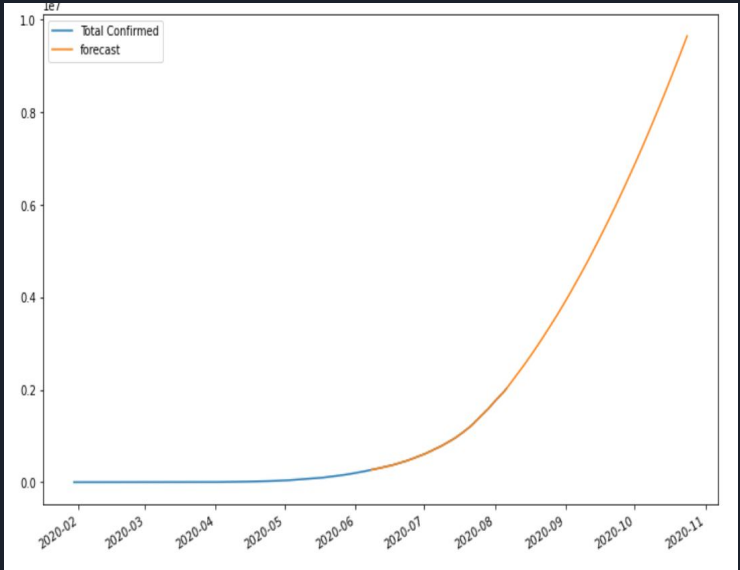
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	1.0000	0.004	224.304	0.000	0.991	1.009
ar.S.L12	-0.2234	0.118	-1.899	0.058	-0.454	0.007
ma.S.L12	-0.6986	0.118	-5.936	0.000	-0.929	-0.468
sigma2	1.637e+06	8.77e-08	1.87e+13	0.000	1.64e+06	1.64e+06

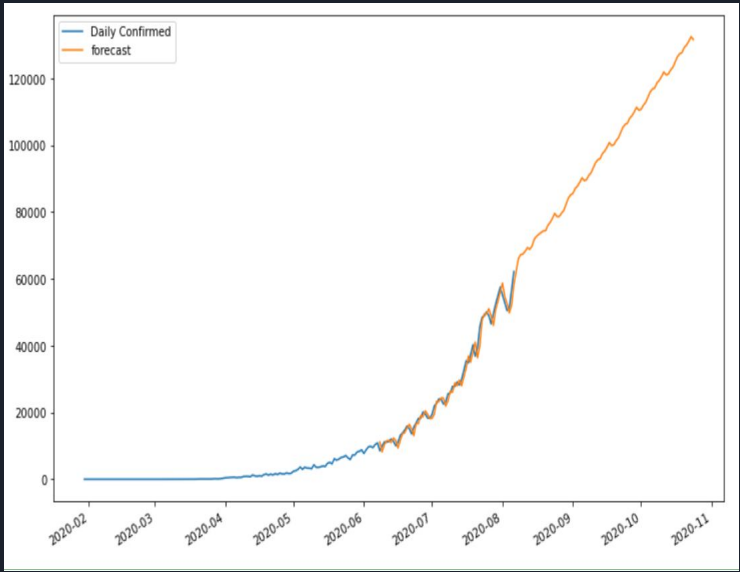
Ljung-Box (Q):	290.18	Jarque-Bera (JB):	237.00
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	1913.50	Skew:	0.90
Prob(H) (two-sided):	0.00	Kurtosis:	8.37

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Algorithm- ARIMA, Covid-19 India

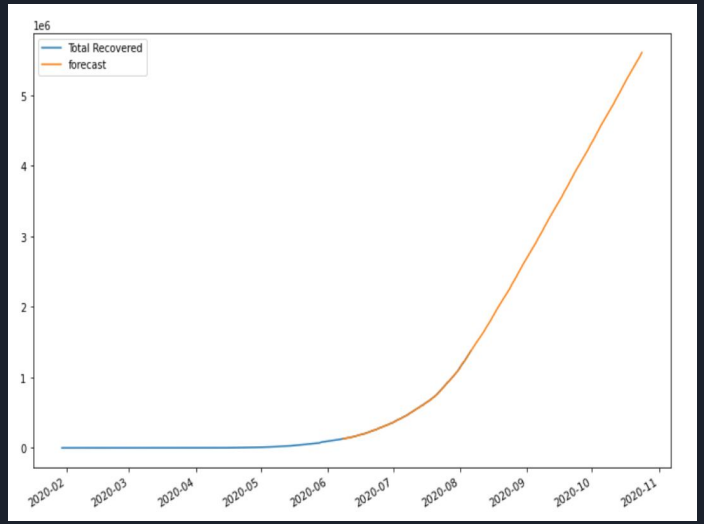
Predictions for total and daily confirmed cases:

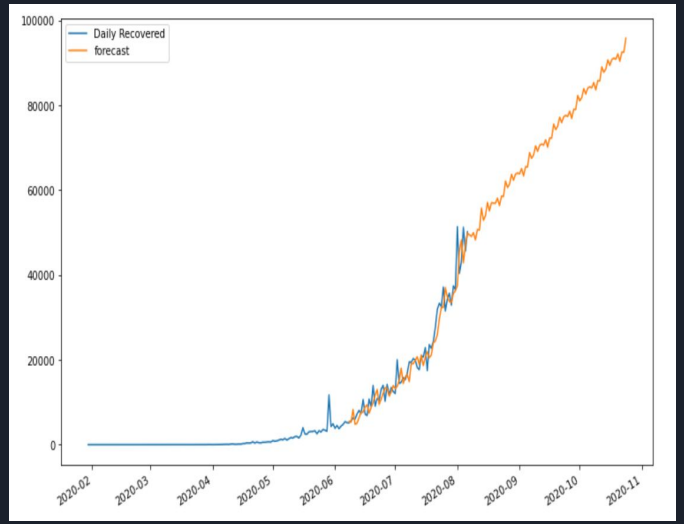




Algorithm- ARIMA, Covid-19 India

Predictions for total and daily recovery cases:





Algorithm- ARIMA, Covid-19 India

Model evaluation using Mean absolute percentage error (MAPE)

```
mape1 = np.mean(np.abs(test1['forecast']-test1['Total Confirmed'])/np.abs(test1['Total Confirmed']))
print('Accuracy of total confirmations forecast: ',100-(mape1*100))
```

```
Accuracy of total confirmations forecast:  99.7786465519674
```

```
mape2 = np.mean(np.abs(test2['forecast']-test2['Daily Confirmed'])/np.abs(test2['Daily Confirmed']))
print('Accuracy of daily confirmations forecast: ',100-(mape2*100))
```

```
Accuracy of daily confirmations forecast:  93.37483878422542
```

```
mape3 = np.mean(np.abs(test3['forecast']-test3['Total Recovered'])/np.abs(test3['Total Recovered']))
print('Accuracy of total recovery forecast: ',100-(mape3*100))
```

```
Accuracy of total recovery forecast:  99.27099011849319
```

```
mape4 = np.mean(np.abs(test4['forecast']-test4['Daily Recovered'])/np.abs(test4['Daily Recovered']))
print('Accuracy of daily recovery forecast: ',100-(mape4*100))
```

```
Accuracy of daily recovery forecast:  87.10281048121178
```

Algorithm- ARIMA, Covid-19 India

Model evaluation using Root Mean Square Error (RMSE)

```
realVals1 = test1['Total Confirmed']
predictedVals1 = test1['forecast']
mse1 = mean_squared_error(realVals1, predictedVals1)
rmse1 = sqrt(mse1)
print('Root mean square error of total confirmed cases forecast: ',rmse1)
```

```
Root mean square error of total confirmed cases forecast:  2125.1317359060663
```

```
realVals2 = test2['Daily Confirmed']
predictedVals2 = test2['forecast']
mse2 = mean_squared_error(realVals2, predictedVals2)
rmse2 = sqrt(mse2)
print('Root mean square error of daily confirmed cases forecast: ' ,rmse2)
```

```
Root mean square error of daily confirmed cases forecast:  2022.7478207589352
```

```
realVals3 = test3['Total Recovered']
predictedVals3 = test3['forecast']
mse3 = mean_squared_error(realVals3, predictedVals3)
rmse3 = sqrt(mse3)
print('Root mean square error of total recovery forecast: ',rmse3)
```

```
Root mean square error of total recovery forecast:  4053.037978541228
```

```
realVals4 = test4['Daily Recovered']
predictedVals4 = test4['forecast']
mse4 = mean_squared_error(realVals4, predictedVals4)
rmse4 = sqrt(mse4)
print('Root mean square error of daily recovery forecast: ', rmse4)
```

```
Root mean square error of daily recovery forecast:  3305.3554918793507
```

Algorithm- ARIMA, Covid-19 India

Model evaluation using R2-score

```
r2_score_1= r2_score(realVals1, predictedVals1)
print('R2 score of total confirmed cases forecast: ', r2_score_1)
```

```
R2 score of total confirmed cases forecast:  0.9999827456330113
```

```
r2_score_2= r2_score(realVals2, predictedVals2)
print('R2 score of daily confirmed cases forecast: ', r2_score_2)
```

```
R2 score of daily confirmed cases forecast:  0.9833497185909372
```

```
r2_score_3= r2_score(realVals3, predictedVals3)
print('R2 score of total recovery forecast: ', r2_score_3)
```

```
R2 score of total recovery forecast:  0.9998682037756753
```

```
r2_score_4= r2_score(realVals4, predictedVals4)
print('R2 score of daily recovery forecast: ', r2_score_4)
```

```
R2 score of daily recovery forecast:  0.9304260281962908
```

Comparison

R2-Score	India	Global
Arima	0.999	0.999
LSTM	0.927	0.996

Conclusion

- From our analysis, we have understood that LSTM and ARIMA provide really accurate results for time series analysis. For the Indian Data, the ARIMA model provided better results compared to the LSTM Model.
- For ARIMA in Indian data, the total prediction gives better accuracy than the daily case predictions. Daily Data has more significant variations than Total Cases for the concurrent dates.
- For the Global Data, the ARIMA model provided slightly better results compared to the LSTM Model.

References

1. Machine Learning Models for Government to Predict COVID-19 Outbreak
   By RAJAN GUPTA, GAURAV PANDEY, POONAM CHAUDHARY and SAIBAL K. PAL

2. Partial derivative Nonlinear Global Pandemic Machine Learning prediction of COVID 19
   By Durga PrasadKavadi,RizwanPatan,ManikandanRamachandran and Amir H.Gandomi