Motivation

Predicting the dynamics of SARS-Cov-2 infections is essential for quick and effective diagnosis of Covid19,public health planning and mitigating burden on healthcare systems.

Dataset Description

The John Hopkins University has a dedicated Github repository for Covid19 where it has been publishing time series data for confirmed, recovered and death cases every day for each country.

Project Overview

Two specific datasets - i.e. time based and country based are created from the JHU data. Exploratoy data analysis is performed on this data followed by modeling using bidirectional convolutional LSTM for forecasting.

References

1. Importing Libraries

```
import pandas as pd
import numpy as np
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
pd.set_option('precision',0)
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv1D, Dense, Flatten, Dropout, BatchNormaliza
from tensorflow.keras.models import Model,Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau,EarlyStopping
import warnings
warnings.filterwarnings('ignore')


```

2. Creating dataset and preprocessing the dataset

We will create a time series data and country data of Covid19 cases with the data exracted from the JHU repository

Extracting data from the JHU Github repository

```
url = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data
df_confirmed = pd.read_csv(url)
url = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data
df_deaths = pd.read_csv(url)
url = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data
df_recovered = pd.read_csv(url)

```

```
df_confirmed.head(5)
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/2
0	NaN	Afghanistan	34	68	0	0	0	0	
1	NaN	Albania	41	20	0	0	0	0	
2	NaN	Algeria	28	2	0	0	0	0	
3	NaN	Andorra	43	2	0	0	0	0	
4	NaN	Angola	-11	18	0	0	0	0	

5 rows × 636 columns

```
df_deaths.head(5)
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/2
0	NaN	Afghanistan	34	68	0	0	0	0	
1	NaN	Albania	41	20	0	0	0	0	
2	NaN	Algeria	28	2	0	0	0	0	
3	NaN	Andorra	43	2	0	0	0	0	
4	NaN	Angola	-11	18	0	0	0	0	

5 rows × 636 columns

```
df_recovered.head(5)
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/2
0	NaN	Afghanistan	34	68	0	0	0	0	
1	NaN	Albania	41	20	0	0	0	0	
2	NaN	Algeria	28	2	0	0	0	0	
3	NaN	Andorra	43	2	0	0	0	0	
4	NaN	Angola	-11	18	0	0	0	0	

5 rows × 636 columns

```
df_list = [df_confirmed,df_deaths,df_recovered]
cases = ['Confirmed', 'Deaths', 'Recovered', 'Active']
case_color = ['orange','red','green','blue']
case_dict = {cases[i]:case_color[i] for i in range(len(cases))}
```

Creating time series data from the extracted data

```
## creating time series data

time_series_data = pd.DataFrame()
for i in range(len(cases)-1):
    df =  pd.DataFrame(df_list[i][df_list[i].columns[4:]].sum(),columns=[cases[i]])
    time_series_data = pd.concat([time_series_data,df],axis = 1)
time_series_data.index = pd.to_datetime(time_series_data.index,format='%m/%d/%y')
time_series_data['Active'] = time_series_data['Confirmed'] - time_series_data['Deaths'] -
time_series_data= time_series_data.rename_axis('ObservationDate').reset_index()
```

```
time_series_data.head(10).style.background_gradient(cmap='PuBu')
```

	ObservationDate	Confirmed	Deaths	Recovered	Active
0	2020-01-22 00:00:00	557	17	30	510
1	2020-01-23 00:00:00	655	18	32	605
2	2020-01-24 00:00:00	941	26	39	876
3	2020-01-25 00:00:00	1434	42	42	1350
4	2020-01-26 00:00:00	2118	56	56	2006
5	2020-01-27 00:00:00	2927	82	65	2780
6	2020-01-28 00:00:00	5578	131	108	5339
7	2020-01-29 00:00:00	6167	133	127	5907
8	2020-01-30 00:00:00	8235	171	145	7919
9	2020-01-31 00:00:00	9927	213	225	9489

```
time_series_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 632 entries, 0 to 631
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ObservationDate  632 non-null    datetime64[ns]
 1   Confirmed        632 non-null    int64
 2   Deaths           632 non-null    int64
 3   Recovered        632 non-null    int64
 4   Active           632 non-null    int64
dtypes: datetime64[ns](1), int64(4)
memory usage: 24.8 KB
```

Creating country wise data from the extracted data

[ ] ↳ *3 cells hidden*

3. Exploratory Data Analysis

```
country_wise_data = country_wise_data.sort_values(by='Confirmed',ascending=False).reset_in
country_wise_data.head(10).style.background_gradient(cmap='Oranges',subset=["Confirmed"])\
.background_gradient(cmap='Reds',subset=['Deaths'])\
.background_gradient(cmap='Greens',subset=["Recovered"])\
.background_gradient(cmap='Blues',subset=["Active"])
```

	Country/Region	Confirmed	Deaths	Recovered	Active
0	US	11205587115	211002971	496971828	10497612316
1	India	7608281596	102130320	4859387857	2646763419
2	Brazil	5379691803	148754887	3412350387	1818586529
3	Russia	1799296853	40777360	1128064202	630455291
4	France	1738215841	38375644	96506536	1603333661
5	United Kingdom	1627917566	47085772	3447801	1577383993
6	Turkey	1456098884	14368097	919259007	522471780
7	Spain	1258730734	30503286	71183831	1157043617
8	Italy	1247523840	43752047	759237934	444533859
9	Argentina	1171933965	26403973	711610324	433919668

Comment: It seems as if the recovery data for US is missing. Based on the above table it can be said that US is the most affected country with respect to number of confirmed cases and fatalities.

```
#Displaying dates where confirmed case counts were highest
time_series_data = time_series_data.sort_values('ObservationDate', ascending=False).reset_
time_series_data.head(10).style.background_gradient(cmap='Oranges',subset=["Confirmed"])\
.background_gradient(cmap='Reds',subset=['Deaths'])\
```

```
.background_gradient(cmap='Greens',subset=["Recovered"])\
.background_gradient(cmap='Blues',subset=["Active"])
```

	ObservationDate	Confirmed	Deaths	Recovered	Active
0	2021-10-14 00:00:00	239608139	4882066	0	234726073
1	2021-10-13 00:00:00	239167859	4874258	0	234293601
2	2021-10-12 00:00:00	238705193	4865619	0	233839574
3	2021-10-11 00:00:00	238272643	4857420	0	233415223
4	2021-10-10 00:00:00	237879268	4851586	0	233027682
5	2021-10-09 00:00:00	237578599	4847106	0	232731493
6	2021-10-08 00:00:00	237249330	4842432	0	232406898
7	2021-10-07 00:00:00	236770669	4834866	0	231935803
8	2021-10-06 00:00:00	236338773	4826388	0	231512385
9	2021-10-05 00:00:00	235825585	4816933	0	231008652

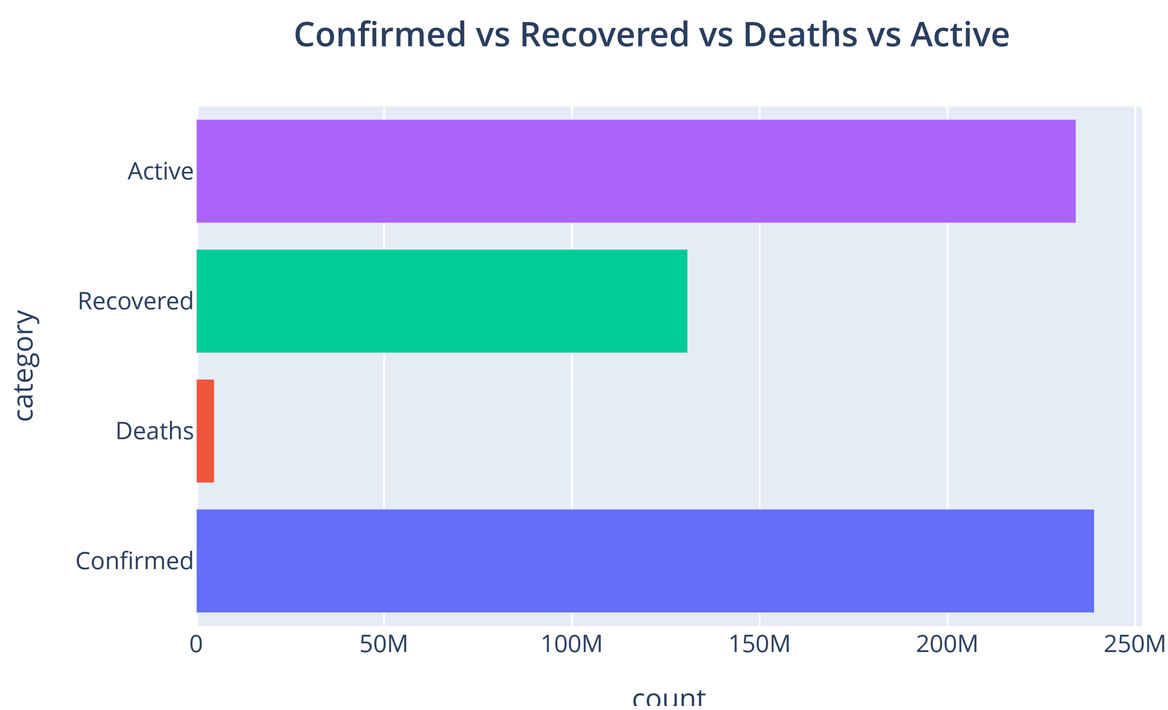
Comment:

Some recovery cases are zero so we will fill with previous non-zero values

```
time_series_data['Recovered'] = time_series_data['Recovered'].replace(to_replace=0, method
time_series_data.head(10).style.background_gradient(cmap='Oranges',subset=["Confirmed"])\
.background_gradient(cmap='Reds',subset=['Deaths'])\
.background_gradient(cmap='Greens',subset=["Recovered"])\
.background_gradient(cmap='Blues',subset=["Active"])
```

	ObservationDate	Confirmed	Deaths	Recovered	Active
0	2021-10-14 00:00:00	239608139	4882066	130899061	234726073
1	2021-10-13 00:00:00	239167859	4874258	130899061	234293601
2	2021-10-12 00:00:00	238705193	4865619	130899061	233839574
3	2021-10-11 00:00:00	238272643	4857420	130899061	233415223
4	2021-10-10 00:00:00	237879268	4851586	130899061	233027682
5	2021-10-09 00:00:00	237578599	4847106	130899061	232731493
6	2021-10-08 00:00:00	237249330	4842432	130899061	232406898
7	2021-10-07 00:00:00	236770669	4834866	130899061	231935803
8	2021-10-06 00:00:00	236338773	4826388	130899061	231512385
9	2021-10-05 00:00:00	235825585	4816933	130899061	231008652

```
count_df = time_series_data.iloc[1,1:]
count_df = pd.DataFrame(count_df).reset_index(level = 0).rename(columns = {'index':'catego
fig = px.bar(count_df, x='count', y='category',
             hover_data=['count'], color='count',
             labels={}, orientation='h',height=400, width = 650)
fig.update_layout(title_text='<b>Confirmed vs Recovered vs Deaths vs Active</b>',title_x=0
fig.show()
```

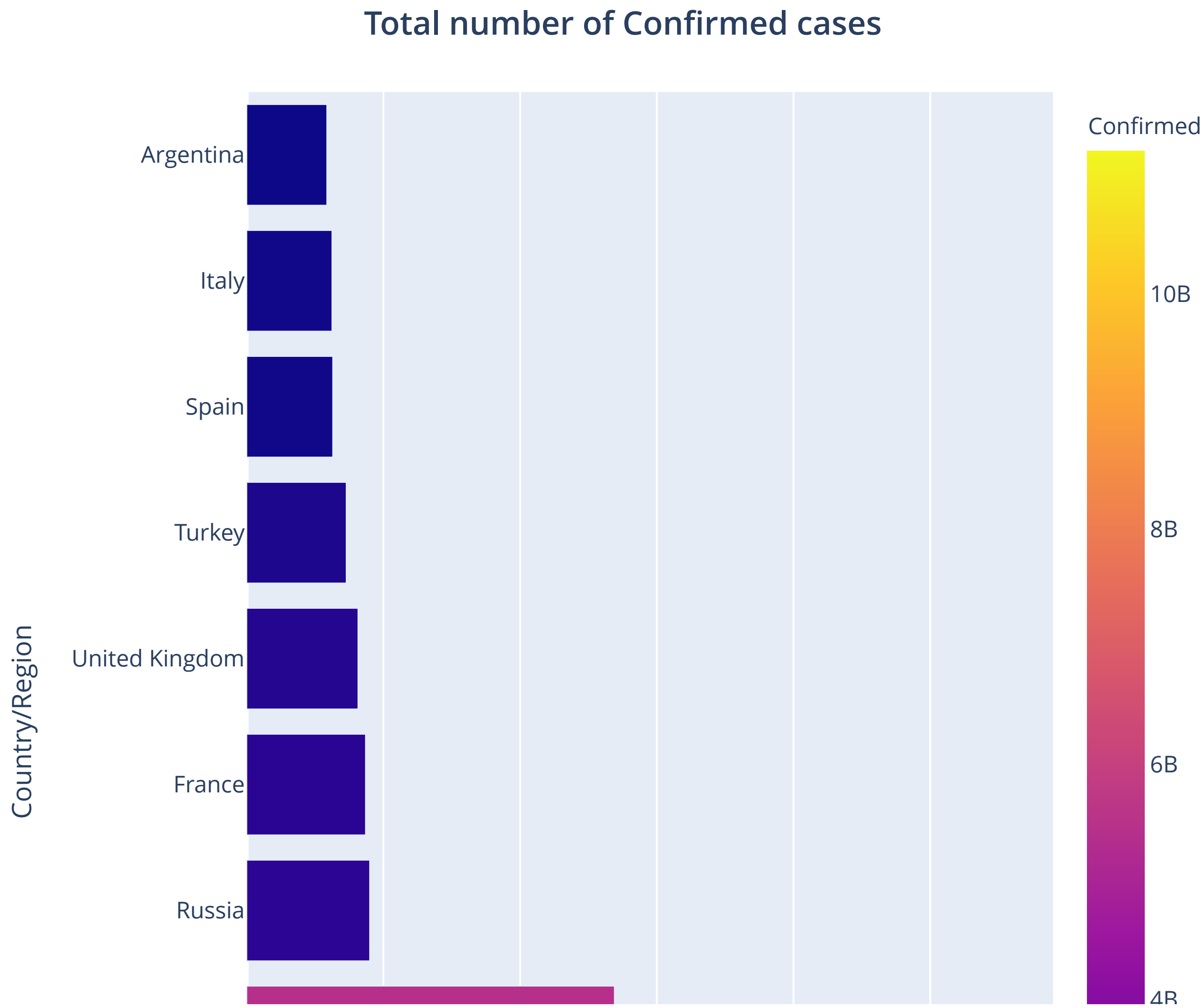


Comment:

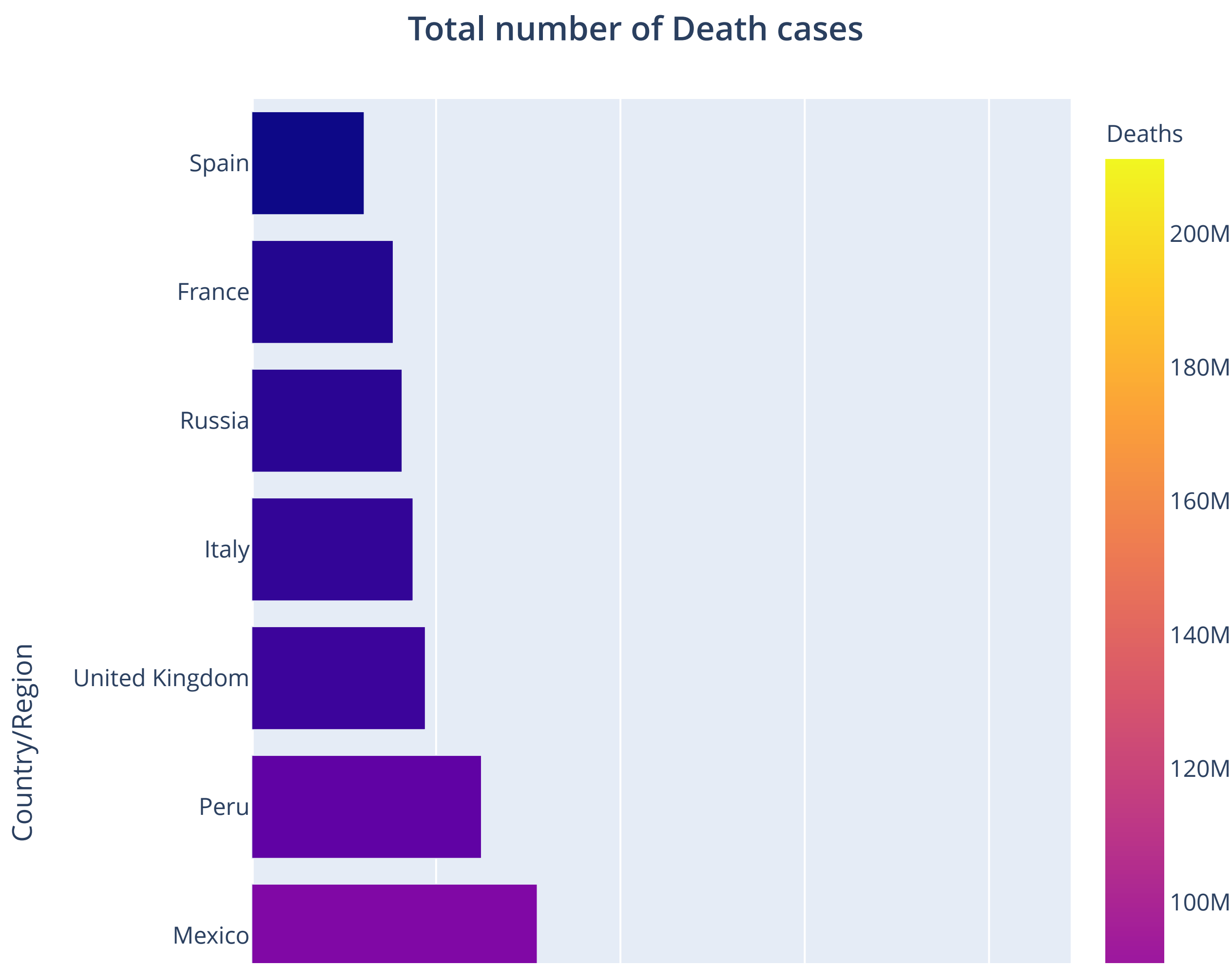
The numbers of deaths are pretty low and the recovery numbers are lesser than the confirmed numbers.

```
df_confirmed = country_wise_data.loc[:,['Country/Region','Confirmed']].sort_values(by = 'C
df_deaths =    country_wise_data.loc[:,['Country/Region','Deaths']].sort_values(by = 'Deat
df_active =    country_wise_data.loc[:,['Country/Region','Active']].sort_values(by = 'Acti
df_recovered =    country_wise_data.loc[:,['Country/Region','Recovered']].sort_values(by =
```

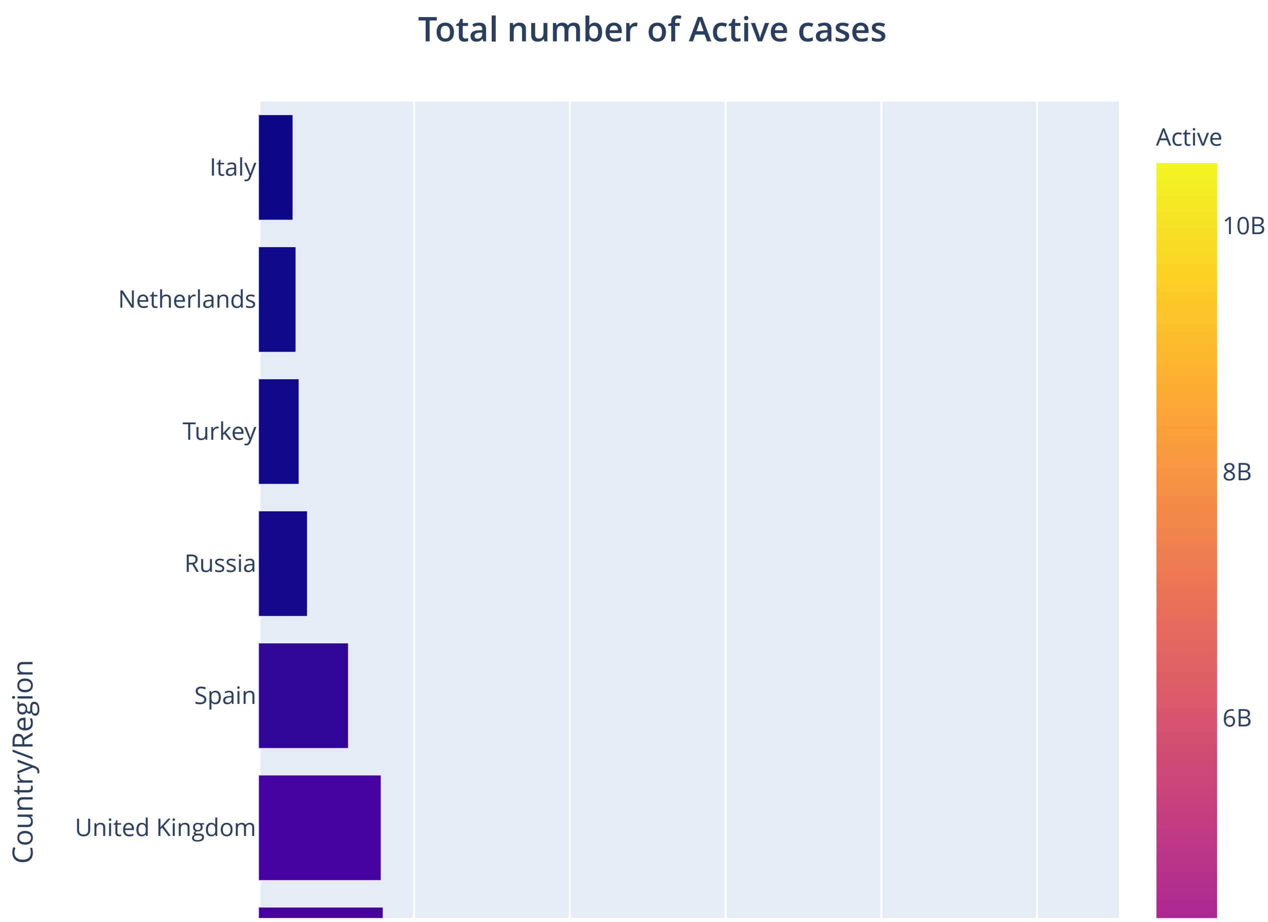
```
fig = px.bar(df_confirmed, x='Confirmed', y='Country/Region',
             hover_data=['Confirmed'], color='Confirmed',
             labels={},orientation='h', height=800, width=650)
fig.update_layout(title_text='<b>Total number of Confirmed cases</b>',title_x=0.5)
fig.show()
```



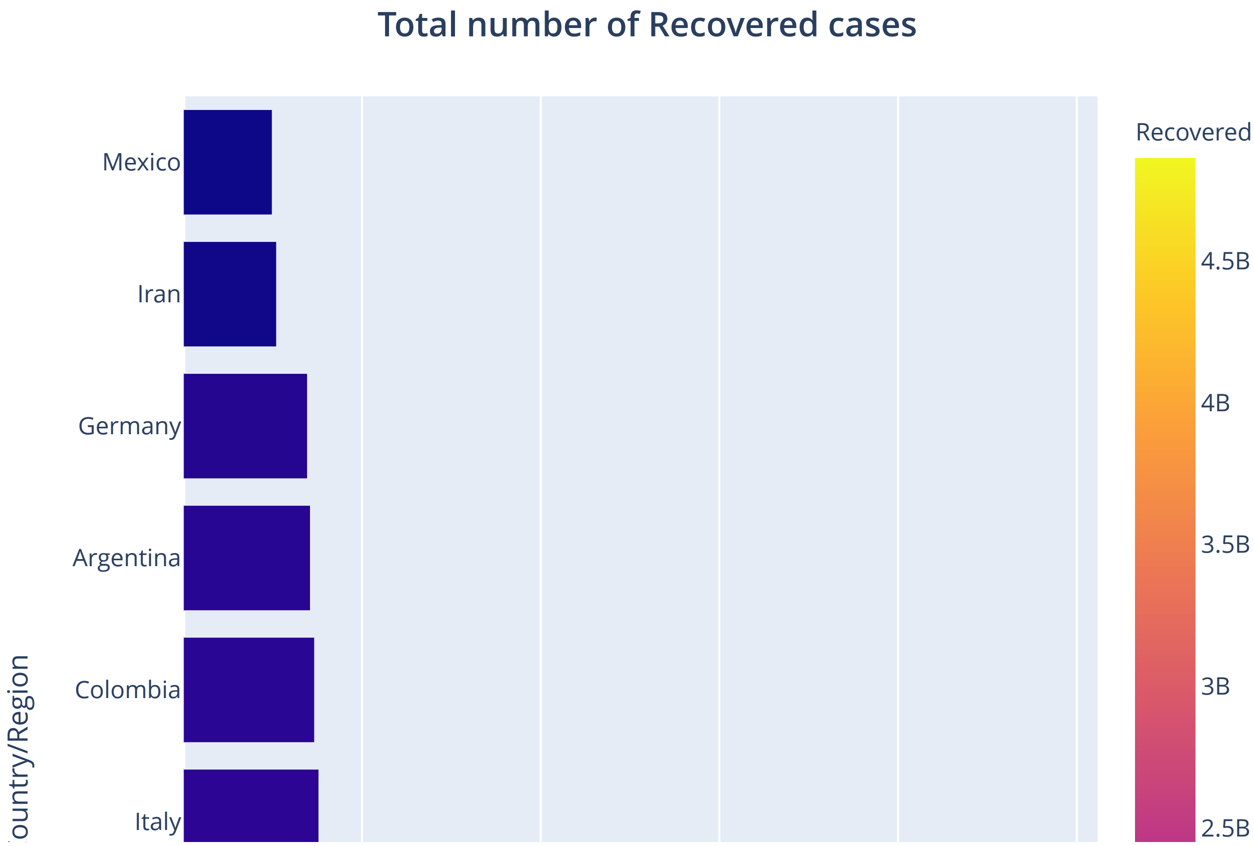

```
fig = px.bar(df_deaths, x='Deaths', y='Country/Region',
             hover_data=['Deaths'], color='Deaths',
             labels={},orientation='h', height=800, width=650)
fig.update_layout(title_text='<b>Total number of Death cases</b>',title_x=0.5)
fig.show()
```




```
fig = px.bar(df_active, x='Active', y='Country/Region',
             hover_data=['Active'], color='Active',
             labels={},orientation='h', height=800, width=650)
fig.update_layout(title_text='<b>Total number of Active cases</b>',title_x=0.5)
fig.show()
```




```
fig = px.bar(df_recovered, x='Recovered', y='Country/Region',
             hover_data=['Recovered'], color='Recovered',
             labels={},orientation='h', height=800, width=650)
fig.update_layout(title_text='<b>Total number of Recovered cases</b>',title_x=0.5)
fig.show()
```




```
# Cases over time
time_series_data = time_series_data.sort_values('ObservationDate').reset_index(drop = True
time_series_data.iloc[:,1:] = time_series_data.iloc[:,1:].astype('int64')
time_series_data.head(10)
```

	ObservationDate	Confirmed	Deaths	Recovered	Active
0	2020-01-22	557	17	30	510
1	2020-01-23	655	18	32	605
2	2020-01-24	941	26	39	876
3	2020-01-25	1434	42	42	1350
4	2020-01-26	2118	56	56	2006
5	2020-01-27	2927	82	65	2780
6	2020-01-28	5578	131	108	5339
7	2020-01-29	6167	133	127	5907
8	2020-01-30	8235	171	145	7919
9	2020-01-31	9927	213	225	9489

```
#cases_over_time moving average
time_series_data_avg = time_series_data.copy()
time_series_data_avg.iloc[:,1:] = time_series_data_avg.iloc[:,1:].rolling(window = 7, min_
```

```
time_series_data_avg.head(20)
```

	ObservationDate	Confirmed	Deaths	Recovered	Active
0	2020-01-22	557	17	30	510
1	2020-01-23	606	18	31	558
2	2020-01-24	718	20	34	664
3	2020-01-25	897	26	36	835
4	2020-01-26	1141	32	40	1069
5	2020-01-27	1439	40	44	1354
6	2020-01-28	2030	53	53	1924
7	2020-01-29	2831	70	67	2695
8	2020-01-30	3914	92	83	3740
9	2020-01-31	5198	118	110	4970
10	2020-02-01	6713	149	145	6419
11	2020-02-02	8808	193	205	8411
12	2020-02-03	11231	242	285	10704
13	2020-02-04	13848	294	392	13163
14	2020-02-05	16916	355	535	16026
15	2020-02-06	20141	421	728	18991
16	2020-02-07	23637	494	984	22159
17	2020-02-08	27221	572	1317	25333
18	2020-02-09	30560	650	1712	28198
19	2020-02-10	33829	733	2187	30909

```
tplot = go.Figure()
for case in cases:
    tplot.add_trace(
        go.Scatter(
            x = time_series_data['ObservationDate'],
            y = time_series_data[case],
            name = case,
            line = dict(color=case_dict[case]),
            hovertemplate ='<br><b>Date</b>: %{x}'+'<br><b>Count</b>: %{y}',
        )
    )
for case in cases:
    tplot.add_trace(
```

```
        go.Scatter(
            x = time_series_data_avg['ObservationDate'],
            y = time_series_data_avg[case],
            name = case + " 7-day moving average",
            line = dict(dash = 'dash',color=case_dict[case]),
            hovertemplate ='<br><b>Date</b>: %{x}'+'<br><b>Moving Average Count</b>: %{y}'
            showlegend = False
        )
    )

tplot.update_layout(
    updatemenus=[
        dict(
        buttons=list(
            [dict(label = 'All Cases',
                  method = 'update',
                  args = [{'visible': [True, True, True, True, True, True, True, True]},
                          {'title': 'All Cases',
                           'showlegend':True}]),
             dict(label = 'Confirmed',
                  method = 'update',
                  args = [{'visible': [True, False, False, False, True, False, False, Fals
                          {'title': 'Confirmed',
                           'showlegend':True}]),
             dict(label = 'Active',
                  method = 'update',
                  args = [{'visible': [False, False, False, True, False, False, False, Tru
                          {'title': 'Active',
                           'showlegend':True}]),
             dict(label = 'Recovered',
                  method = 'update',
                  args = [{'visible': [False, False, True, False, False, False, True, Fals
                          {'title': 'Recovered',
                           'showlegend':True}]),
             dict(label = 'Deaths',
                  method = 'update',
                  args = [{'visible': [False, True, False, False, False, True, False, Fals
                          {'title': 'Deaths',
                           'showlegend':True}]),
            ]),type = 'buttons',
             direction="right",
             showactive = True,
             x=-0.25,
             xanchor="left",
             y=1.25,
             yanchor="top"
        ),
        dict(
        buttons=list(
            [dict(label = 'Linear Scale',
                  method = 'relayout',
                  args = [{'yaxis': {'type': 'linear'}},
```

```
                          {'title': 'All Cases',
                           'showlegend':True}]),
             dict(label = 'Log Scale',
                  method = 'relayout',
                  args = [{'yaxis': {'type': 'log'}},
                          {'title': 'All Cases',
                           'showlegend':True}]),
            ]),
             direction="right",
             x=-0.25,
             xanchor="left",
             y=1.39,
             yanchor="top"
        )
    ])

tplot.update_layout(
    height=600, width=1100,
    title_text="<b>Global cases over time</b>", title_x=0.5, title_font_size=20,
                            legend=dict(orientation='h',yanchor='top',y=1.15,xanchor='righ
                            xaxis_title="Observation Date", yaxis_title="Number of Cases")
tplot.show()

```

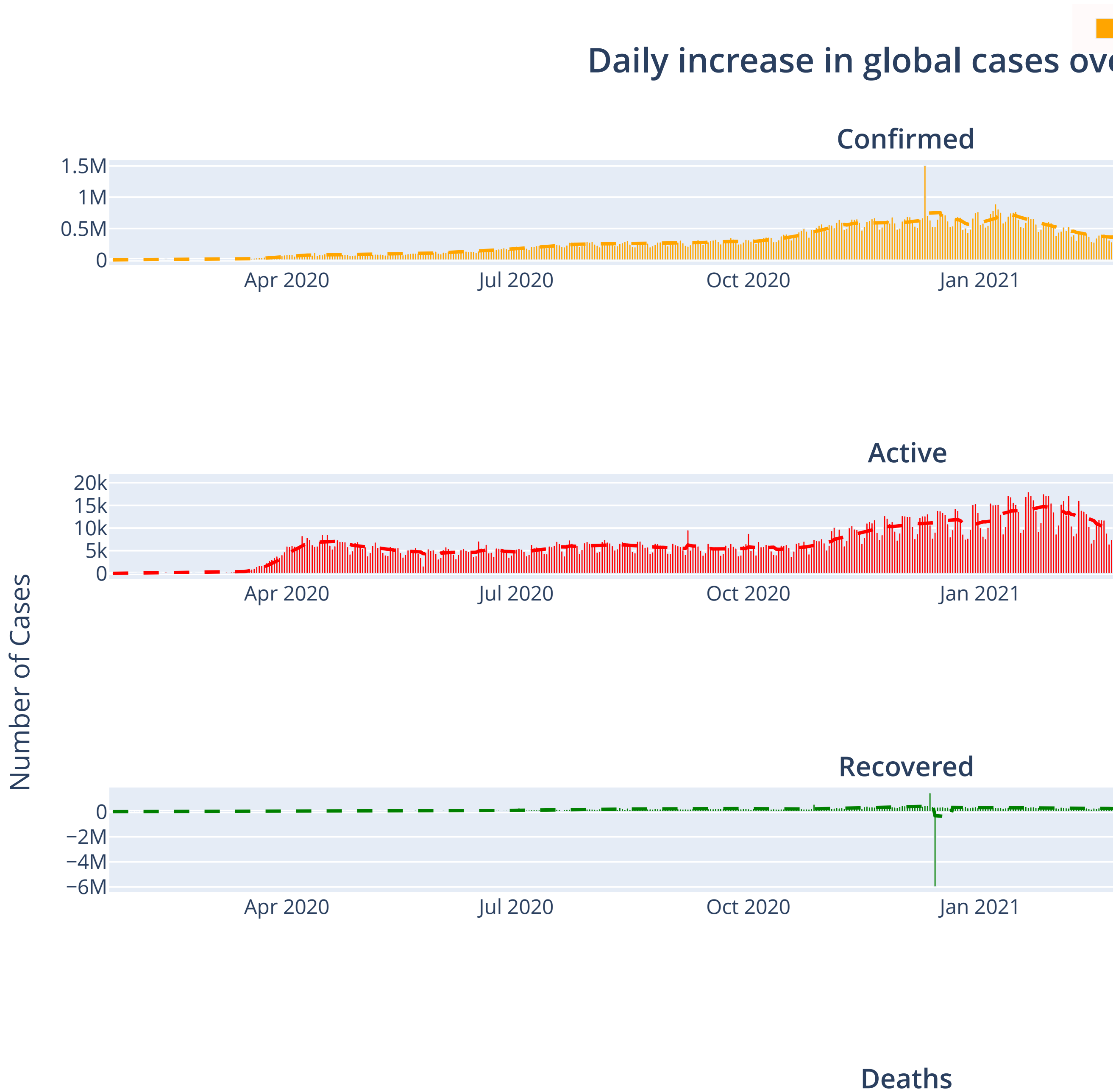
Linear Scale ▶

All Cases | Confirmed | Active | Recovered | Deaths

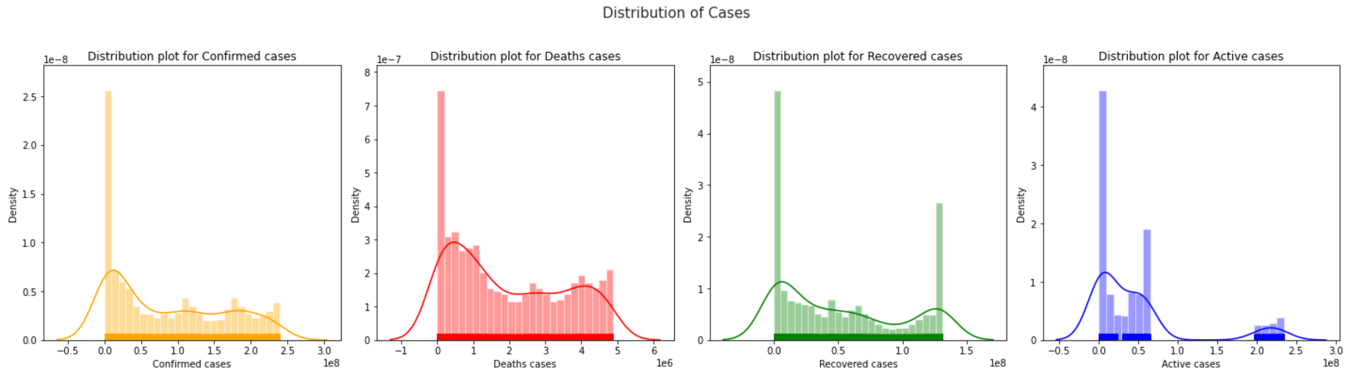
Global cases over time

```
# Cases over time
time_series_data_increase = time_series_data.copy()
time_series_data_increase.iloc[:,1:] = time_series_data_increase.iloc[:,1:].diff(1)
time_series_data_increase_avg  = time_series_data_increase.copy()
time_series_data_increase_avg.iloc[:,1:] = time_series_data_increase_avg.iloc[:,1:].rollin
```

```
fig = make_subplots(rows=len(cases), cols=1, vertical_spacing=0.2, horizontal_spacing=0.04
                           subplot_titles=('<b>Confirmed</b>','<b>Active</b>','<b>Recovere
                            x_title="Observation Date", y_title="Number of Cases")


for i in range(len(cases)):

    fig.add_trace( go.Bar(
                x = time_series_data_increase['ObservationDate'],
                y = time_series_data_increase[cases[i]],
                name = cases[i],
                hovertemplate ='<br><b>Date</b>: %{x}'+'<br><b>Count</b>: %{y}',
                marker = dict(color = case_dict[cases[i]])
            ),row = i+1, col = 1)

    fig.add_trace( go.Scatter(
                x = time_series_data_increase_avg['ObservationDate'],
                y = time_series_data_increase_avg[cases[i]],
                name = cases[i],
                hovertemplate ='<br><b>Date</b>: %{x}'+'<br><b>7-day average</b>: %{y}',
                showlegend=False,
                line=dict(dash="dash", color=case_color[i])
            ),row = 1+i, col = 1)
fig.update_layout(
    height=800, width=1100,
    title_text="<b>Daily increase in global cases over time</b>", title_x=0.5, title_font_
                            legend=dict(orientation='h',yanchor='top',y=1.15,xanchor='righ
fig.show()
```




```
fig,ax = plt.subplots(1,4,figsize = (20,5))
for i in range(len(cases)):
    sns.set_style('whitegrid')
    sns.distplot(time_series_data[cases[i]], kde = True, rug = True, bins = 25,ax =ax[i],c
    ax[i].set_xlabel( cases[i]+ ' cases')
    ax[i].set_ylabel('Density')
    ax[i].set_title('Distribution plot for ' + cases[i]+ ' cases')
fig.tight_layout()
plt.suptitle('Distribution of Cases',fontsize = 15,y = 1.1)
plt.show()

```



Comment:
The distribution plots are skewed. So appropriate normalization technqiues need to be applied before modelling.

```

# m = np.mean(time_series_data.iloc[:,1:], axis=0) # array([16.25, 26.25])
# std = np.std(time_series_data.iloc[:,1:], axis=0) # array([17.45530005, 22.18529919])
# md = np.median(time_series_data.iloc[:,1:],axis = 0)
# p75 = np.percentile(time_series_data.iloc[:,1:],75,axis = 0)
# p25 = np.percentile(time_series_data.iloc[:,1:],25,axis = 0)
```

```
# time_series_normalized = time_series_data.copy()
# time_series_normalized.iloc[:,1:] = np.tanh((time_series_normalized.iloc[:,1:] - md) / (
```

```
# fig,ax = plt.subplots(1,4,figsize = (20,5))
# for i in range(len(cases)):
#     sns.set_style('whitegrid')
#     sns.distplot(time_series_normalized[cases[i]], kde = True, rug = True, bins = 25,ax
#     ax[i].set_xlabel( cases[i]+ ' cases')
#     ax[i].set_ylabel('Density')
#     ax[i].set_title('Distribution plot for ' + cases[i]+ ' cases')
# fig.tight_layout()
# plt.suptitle('Distribution of Normalized time series data',fontsize = 15,y = 1.1)
# plt.show()
```

4.Predictive modelling using CNN + Bi-Directional LSTM

Creating dataset for predictive modelling

```
dataset = time_series_data.iloc[:,1].values #using only confirmed cases
dataset.shape
```

```
(632,)
```

```
#Feature scaling
split = round(0.8*len(dataset))
dataset = dataset.reshape(-1,1)
scaler = MinMaxScaler(feature_range=(0,1))
scaler.fit(dataset[:split]) #fit only on the training data which is the first 80%
dataset_n = scaler.transform(dataset).flatten()
dataset_n.shape

```

```
(632,)
```

```
def create_dataset(df,previous,split_ratio):
    X, Y = [], []
    for i in range(len(df)-previous):
        a = df[i:(i+previous)]
        X.append(a)
        y = df[i+previous]
        Y.append(y)
    X = np.array(X).reshape(-1,T,1)
    Y = np.array(Y)
    N = len(X)
    split = round(split_ratio*len(df))
    X_train = X[:split]
    X_test = X[split:]
    Y_train = Y[:split]
    Y_test = Y[split:]
    print("X.shape", X.shape, "Y.shape", Y.shape)
    print("X_train.shape", X_train.shape, "Y_train.shape", Y_train.shape)
    print("X_test.shape", X_test.shape, "Y_test.shape", Y_test.shape)
    return X,X_train,X_test,Y,Y_train,Y_test
```

```
T = 30  #number of past days used to predict the value for the current day
X,X_train,X_test,Y,Y_train,Y_test = create_dataset(dataset_n,T,0.8) #80% of data for train
```

```
X.shape (602, 30, 1) Y.shape (602,)
X_train.shape (506, 30, 1) Y_train.shape (506,)
X_test.shape (96, 30, 1) Y_test.shape (96,)
```

Building the model architecture

```
def plotLearningCurve(history,epochs,text):
```

```
epochRange = range(1,epochs+1)
plt.figure(figsize = (5,5))
plt.plot(epochRange,history.history['loss'])
plt.plot(epochRange,history.history['val_loss'])
plt.title('Model Loss for ' + text)
plt.xlabel('Epoch', fontsize = 20)
plt.ylabel('Loss', fontsize = 20)
plt.legend(['Training set','Validation set'])
plt.show()
```

```
def lstm_model(previous):
  i = Input(shape=(previous,1)) #input shape is n-timesteps x n-features
  x = Conv1D(filters=32, kernel_size= 3, strides=2, padding="causal", activation="relu")
  x = LSTM(64, return_sequences=True)(x)
  x = Dropout(0.4)(x)
  x = LSTM(64,return_sequences=True)(x)
  x = Dropout(0.4)(x)
  x = LSTM(64)(x)
  x = Dropout(0.2)(x)
  x = Dense(1)(x)
  model = Model(i, x)
  model.summary()
  return model
```

```
model = lstm_model(T)
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 30, 1)]           0
_________________________________________________________________
conv1d (Conv1D)              (None, 15, 32)            128
_________________________________________________________________
lstm (LSTM)                  (None, 15, 64)            24832
_________________________________________________________________
dropout (Dropout)            (None, 15, 64)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 15, 64)            33024
_________________________________________________________________
dropout_1 (Dropout)          (None, 15, 64)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 64)                33024
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 91,073
Trainable params: 91,073
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss = 'mse',
             optimizer = 'adam')
```

```
batchsize = 64
epochs = 100
learning_rate_reduction = ReduceLROnPlateau(monitor='val_loss',
                                            patience=3,
                                            verbose=1,
                                            factor=0.8,
                                            min_lr=1e-10)
early_stop = EarlyStopping(monitor='val_loss',patience=15,restore_best_weights=True)
r = model.fit(X_train,
                    Y_train,
                    batch_size=batchsize,
                    epochs=epochs,
                    validation_split=0.2,
                    shuffle=False, #time_series
                    callbacks=[learning_rate_reduction,early_stop])
```

```
Epoch 1/100
7/7 [==============================] - 7s 221ms/step - loss: 0.0454 - val_loss: 0.1453
Epoch 2/100
7/7 [==============================] - 0s 45ms/step - loss: 0.0259 - val_loss: 0.0105
Epoch 3/100
7/7 [==============================] - 0s 46ms/step - loss: 0.0169 - val_loss: 0.0241
Epoch 4/100
7/7 [==============================] - 0s 45ms/step - loss: 0.0077 - val_loss: 0.0056
Epoch 5/100
7/7 [==============================] - 0s 45ms/step - loss: 0.0149 - val_loss: 0.0427
Epoch 6/100
7/7 [==============================] - 0s 44ms/step - loss: 0.0018 - val_loss: 0.0062
Epoch 7/100
7/7 [==============================] - 0s 46ms/step - loss: 0.0046 - val_loss: 1.1991e-0
Epoch 8/100
7/7 [==============================] - 0s 45ms/step - loss: 0.0063 - val_loss: 0.0261
Epoch 9/100
7/7 [==============================] - 0s 45ms/step - loss: 0.0019 - val_loss: 2.2199e-0
Epoch 10/100
7/7 [==============================] - 0s 44ms/step - loss: 0.0023 - val_loss: 2.3604e-0

Epoch 00010: ReduceLROnPlateau reducing learning rate to 0.000800000037997961.
Epoch 11/100
7/7 [==============================] - 0s 47ms/step - loss: 0.0023 - val_loss: 0.0092
Epoch 12/100
7/7 [==============================] - 0s 45ms/step - loss: 0.0014 - val_loss: 5.0023e-0
Epoch 13/100
7/7 [==============================] - 0s 44ms/step - loss: 0.0012 - val_loss: 0.0013

Epoch 00013: ReduceLROnPlateau reducing learning rate to 0.0006400000303983689.
Epoch 14/100
7/7 [==============================] - 0s 47ms/step - loss: 0.0015 - val_loss: 8.0936e-0
Epoch 15/100
7/7 [==============================] - 0s 45ms/step - loss: 0.0013 - val_loss: 6.3139e-0
Epoch 16/100
```

```
7/7 [==============================] - 0s 44ms/step - loss: 0.0015 - val_loss: 0.0030

Epoch 00016: ReduceLROnPlateau reducing learning rate to 0.0005120000336319208.
Epoch 17/100
7/7 [==============================] - 0s 44ms/step - loss: 0.0013 - val_loss: 0.0010
Epoch 18/100
7/7 [==============================] - 0s 43ms/step - loss: 0.0013 - val_loss: 0.0019
Epoch 19/100
7/7 [==============================] - 0s 43ms/step - loss: 0.0012 - val_loss: 0.0038

Epoch 00019: ReduceLROnPlateau reducing learning rate to 0.00040960004553198815.
Epoch 20/100
7/7 [==============================] - 0s 44ms/step - loss: 9.1699e-04 - val_loss: 0.002
Epoch 21/100
7/7 [==============================] - 0s 43ms/step - loss: 0.0012 - val_loss: 0.0017
Epoch 22/100
7/7 [==============================] - 0s 44ms/step - loss: 0.0013 - val_loss: 0.0018

Epoch 00022: ReduceLROnPlateau reducing learning rate to 0.00032768002711236477.
```

```
n_epochs = len(r.history['loss'])
```

```
print("Train score:", model.evaluate(X_train,Y_train))
print("Test score:", model.evaluate(X_test,Y_test))

```

```
16/16 [==============================] - 0s 8ms/step - loss: 5.1284e-04
Train score: 0.000512842379976064
3/3 [==============================] - 1s 10ms/step - loss: 0.0019
Test score: 0.0018639853224158287
```

```
plotLearningCurve(r,n_epochs,text = 'CNN+LSTM model')
```

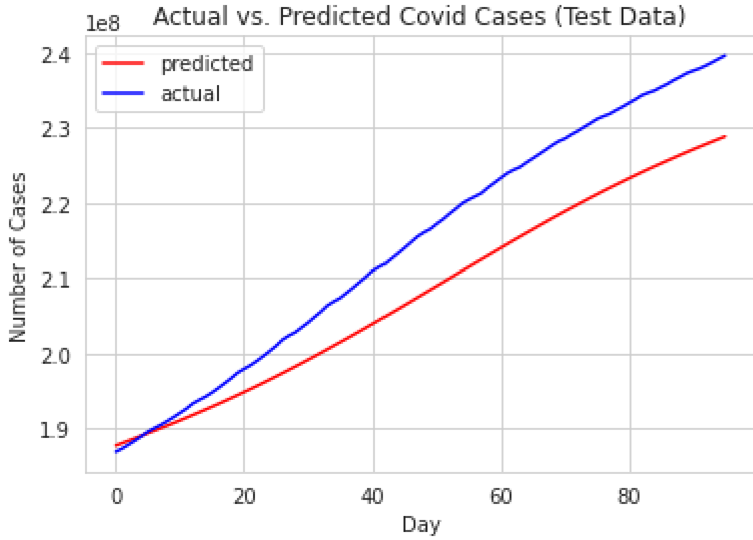

```
r.history['loss']
```

```
[0.045422572642564774,
 0.025905705988407135,
 0.016938544809818268,
 0.007742736488580704,
 0.014860113151371479,
 0.0018392590573057532,
 0.004579311702400446,
 0.006305883638560772,
 0.0018983760382980108,
 0.002292555756866932,
 0.0023065414279699326,
 0.0014067406300455332,
 0.0012126350775361061,
 0.0014814937021583319,
 0.0013433381682261825,
 0.0015455055981874466,
 0.0013058619806542993,
 0.0013411532854661345,
 0.0012355889193713665,
 0.0009169915574602783,
 0.0012460516300052404,
 0.00127340666949749]
```

Predicting test data

```
Y_pred = model.predict(X_test)
Y_pred = scaler.inverse_transform(Y_pred)
Y_test = scaler.inverse_transform(Y_test.reshape(-1,1))
plt.plot(Y_pred, color='red')
plt.plot(Y_test, color='blue')
plt.title('Actual vs. Predicted Covid Cases (Test Data)')
plt.ylabel('Number of Cases')
plt.xlabel('Day')
plt.legend(['predicted', 'actual'])
```

```
<matplotlib.legend.Legend at 0x7f0b2c584150>
```



✓ 1s completed at 2:25 PM