```
# importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.style as style
style.available
import seaborn as sns
%matplotlib inline
import folium
from sklearn.linear_model import LinearRegression, BayesianRidge
from sklearn.model_selection import RandomizedSearchCV,
train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, mean_absolute_error
import plotly.express as px
import plotly.graph_objs as go
from plotly.subplots import make_subplots
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
confirmed_df =
pd.read_csv('/content/drive/MyDrive/cricket/time_series_covid19_confir
med_global.csv')
deaths_df =
pd.read_csv('/content/drive/MyDrive/cricket/time_series_covid19_deaths
_global.csv')
recoveries_df =
pd.read_csv('/content/drive/MyDrive/cricket/time_series_covid19_recove
red_global.csv')
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
<ipython-input-1-fb8c44336eda> in <cell line: 1>()
----> 1 confirmed_df =
pd.read_csv('/content/drive/MyDrive/cricket/time_series_covid19_confir
med_global.csv')
      2 deaths_df =
pd.read_csv('/content/drive/MyDrive/cricket/time_series_covid19_deaths
_global.csv')
      3 recoveries_df =
pd.read_csv('/content/drive/MyDrive/cricket/time_series_covid19_recove
red_global.csv')

NameError: name 'pd' is not defined
```

```python
# shape of dataframe
print(confirmed_df.shape)
print(deaths_df.shape)
print(recoveries_df.shape)
```

```
(266, 178)
(266, 178)
(253, 178)
```

```python
#Information of Confirmed Cases Dataset
confirmed_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Columns: 178 entries, Province/State to 7/13/20
dtypes: float64(2), int64(174), object(2)
memory usage: 370.0+ KB
```

```python
#Statistical details of confirmed cases dataset
confirmed_df.describe()
```

```
{"type":"dataframe"}
```

```python
# first 5 rows of confirmed cases
confirmed_df.head()
```

```
{"type":"dataframe","variable_name":"confirmed_df"}
```

```python
# checking null values of confirmed cases
confirmed_df.isna().sum()
```

```
Province/State    185
Country/Region      0
Lat                 0
Long                0
1/22/20             0
                 ...
7/9/20              0
7/10/20             0
7/11/20             0
7/12/20             0
7/13/20             0
Length: 178, dtype: int64
```

```python
# checking all unique values of confirmed cases
confirmed_df.nunique()
```

```
Province/State     81
Country/Region    188
Lat               262
Long              263
1/22/20            11
```

```
                 ...
7/9/20            254
7/10/20           256
7/11/20           259
7/12/20           260
7/13/20           255
Length: 178, dtype: int64
```

```python
# value counts by country in confirmed cases
confirmed_df['Country/Region'].value_counts()
```

```
Country/Region
China             33
Canada            14
United Kingdom    11
France            11
Australia          8
                  ..
Honduras           1
Hungary            1
Iceland            1
India              1
Lesotho            1
Name: count, Length: 188, dtype: int64
```

```python
#Information of Death Cases Dataset
deaths_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Columns: 178 entries, Province/State to 7/13/20
dtypes: float64(2), int64(174), object(2)
memory usage: 370.0+ KB
```

```python
#Statistical details of deaths cases dataset
deaths_df.describe()
```

```
{"type":"dataframe"}
```

```python
# first 5 rows of death cases
deaths_df.head()
```

```
{"type":"dataframe","variable_name":"deaths_df"}
```

```python
# checking null values of death cases
deaths_df.isna().sum()
```

```
Province/State    185
Country/Region      0
Lat                 0
Long                0
```

```
1/22/20             0
                 ...
7/9/20              0
7/10/20             0
7/11/20             0
7/12/20             0
7/13/20             0
Length: 178, dtype: int64
```

```python
# checking all unique values of death cases
deaths_df.nunique()
```

```
Province/State     81
Country/Region    188
Lat               262
Long              263
1/22/20             2
                 ...
7/9/20            141
7/10/20           143
7/11/20           143
7/12/20           140
7/13/20           143
Length: 178, dtype: int64
```

```python
# value counts by country in death cases
deaths_df['Country/Region'].value_counts()
```

```
Country/Region
China             33
Canada            14
United Kingdom    11
France            11
Australia          8
                  ..
Honduras           1
Hungary            1
Iceland            1
India              1
Lesotho            1
Name: count, Length: 188, dtype: int64
```

```python
#Information of Recovery Cases Dataset
recoveries_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 253 entries, 0 to 252
Columns: 178 entries, Province/State to 7/13/20
dtypes: float64(2), int64(174), object(2)
memory usage: 352.0+ KB
```

```
#Statistical details of Recovery cases dataset
recoveries_df.describe()
```

{"type":"dataframe"}

```
# first 5 rows of recovery cases
recoveries_df.head()
```

{"type":"dataframe","variable_name":"recoveries_df"}

```
# checking null values of recovery cases
recoveries_df.isna().sum()
```

```
Province/State    186
Country/Region      0
Lat                 0
Long                0
1/22/20             0
                 ...
7/9/20              0
7/10/20             0
7/11/20             0
7/12/20             0
7/13/20             0
Length: 178, dtype: int64
```

```
# checking all unique values of recovery cases
recoveries_df.nunique()
```

```
Province/State     67
Country/Region    188
Lat               252
Long              252
1/22/20             2
                 ...
7/9/20            233
7/10/20           236
7/11/20           236
7/12/20           237
7/13/20           237
Length: 178, dtype: int64
```

```
# value counts by country in recovery cases
recoveries_df['Country/Region'].value_counts()
```

```
Country/Region
China             33
United Kingdom    11
France            11
Australia          8
Netherlands        5
```

```
                         ..
Guinea                    1
Guinea-Bissau             1
Guyana                    1
Haiti                     1
Lesotho                   1
Name: count, Length: 188, dtype: int64
```

Cleaning:

```
# Rename columns 'Province/State' & 'Country/Region' & change latest
date to 'Current'.

col=confirmed_df.columns[-1]

confirmed_df.rename(columns = {'Province/State' : 'Province',
'Country/Region' : 'Country', col : 'Current'},inplace = True)
deaths_df.rename(columns = {'Province/State' : 'Province',
'Country/Region' : 'Country', col : 'Current'},inplace = True)
recoveries_df.rename(columns = {'Province/State' : 'Province',
'Country/Region' : 'Country', col : 'Current'},inplace = True)

# confirmed cases
confirm = pd.DataFrame(confirmed_df.groupby('Country').sum())
confirm.reset_index(inplace = True)

# drop Lat & Long columns as they will not give accurate results
col = confirm['Country']
confirm.drop(['Lat','Long'],axis=1,inplace=True)
confirm.head()

{"type":"dataframe","variable_name":"confirm"}

# deaths
deaths= pd.DataFrame(deaths_df.groupby('Country').sum())
deaths.reset_index(inplace = True)

# drop Lat & Long columns as they do not give accurate results
deaths.drop(['Lat','Long'],axis=1,inplace=True)
deaths.head()

{"type":"dataframe","variable_name":"deaths"}

# recovery

recovery = pd.DataFrame(recoveries_df.groupby('Country').sum())
recovery.reset_index(inplace = True)
```

```
# drop Lat & Long columns as they do not give accurate results
recovery.drop(['Lat','Long'],axis=1,inplace=True)
recovery.head()

{"type":"dataframe","variable_name":"recovery"}
```

Create new dataframe for active cases:

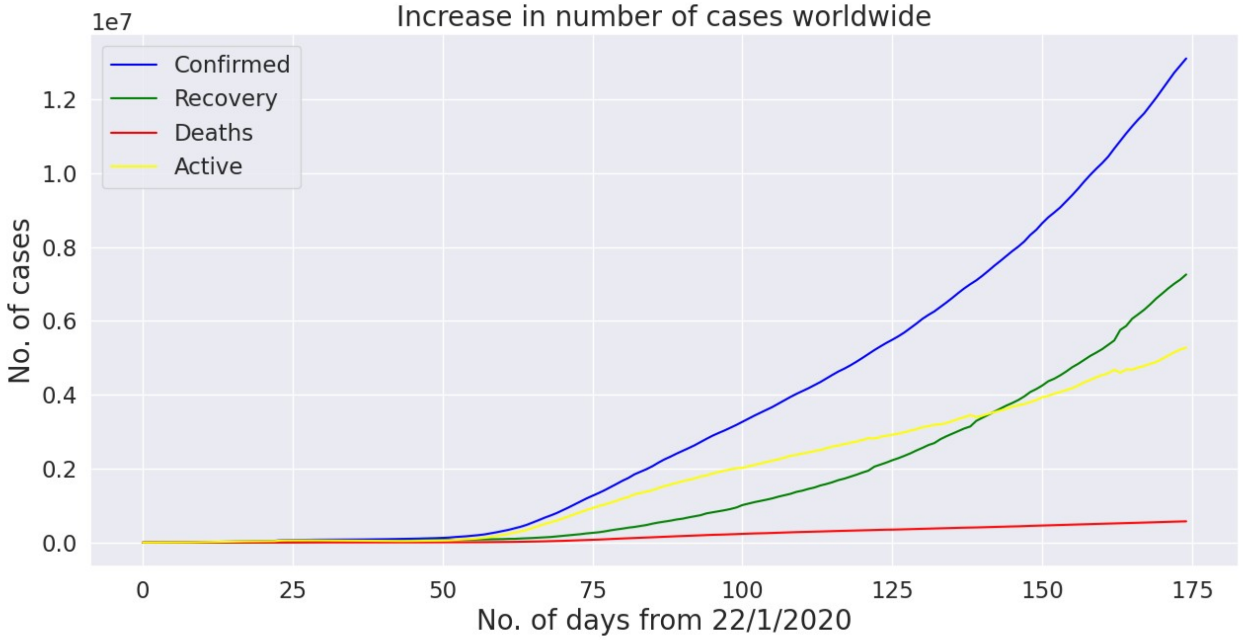
```
import pandas as pd

# Assuming 'confirm', 'recovery', and 'deaths' are your DataFrames
# Convert all columns except the first one to numeric
confirm.iloc[:, 1:] = confirm.iloc[:, 1:].apply(pd.to_numeric, 
errors='coerce')
recovery.iloc[:, 1:] = recovery.iloc[:, 1:].apply(pd.to_numeric, 
errors='coerce')
deaths.iloc[:, 1:] = deaths.iloc[:, 1:].apply(pd.to_numeric, 
errors='coerce')

# Create active cases DataFrame
active = confirm.copy()

# Calculate active cases
for i in active.columns[1:]:
    active[i] = active[i] - recovery[i] - deaths[i]

# Display the first few rows of the active DataFrame
active.head()

{"type":"dataframe","variable_name":"active"}
```

Now, that we have all the data of the recorded cases such as active cases, recovery cases, confirmed cases and death cases, I will calculate the total number of cases(confirmed, active, recovery, death) worldwide with the help of given dataset.

```
# Total number of confirmed cases, recovered cases, death cases and 
active cases till date as per the given dataset


print("Confirmed Cases :" , confirm.iloc[:,-1].sum())
print("Recovered Cases :" , recovery.iloc[:,-1].sum())
print("Death Cases :" , deaths.iloc[:,-1].sum())
print("Active Cases :", active.iloc[:,-1].sum())


Confirmed Cases : 13104391
Recovered Cases : 7257369
Death Cases : 573003
Active Cases : 5274019
```

Now, we know total number of cases worldwide, we will visualize it to understand it better.

```
# Plotting the number of confirmed cases, recovered cases, death cases
and active cases worldwide as per the given dataset.

confirm_date = confirm.iloc[:,1:].sum().values.tolist()
recovery_date = recovery.iloc[:,1:].sum().values.tolist()
deaths_date = deaths.iloc[:,1:].sum().values.tolist()
active_date = active.iloc[:,1:].sum().values.tolist()

plt.figure(figsize=(15,7))
plt.plot(confirm_date,color='Blue')
plt.plot(recovery_date,color='Green')
plt.plot(deaths_date,color='Red')
plt.plot(active_date,color='Yellow')

plt.xlabel('No. of days from 22/1/2020',size=20)
plt.ylabel('No. of cases',size=20)
plt.title('Increase in number of cases worldwide',size=20)
plt.legend(['Confirmed','Recovery','Deaths','Active'])
plt.show()
```



```
# Plotting the number of confirmed cases, recovered cases, death cases
and active cases worldwide
#as per the given dataset
#in scatterplots
Confirmed = confirm.sum()
```

```
Recovered = recovery.sum()
Death = deaths.sum()
Actives = active.sum()
fig = go.Figure()
fig.add_trace(go.Scatter(x=Confirmed.index, y=Confirmed.values, mode =
'lines+markers', name = 'Confirmed',
line = dict(color = "Blue", width = 2 )))
fig.add_trace(go.Scatter(x=Recovered.index, y=Recovered.values, mode =
'lines+markers', name = 'Recovered',
line = dict(color = "Green", width = 2)))
fig.add_trace(go.Scatter(x=Death.index, y=Death.values, mode =
'lines+markers', name = 'Deaths',
line = dict(color = "Red", width = 2)))
fig.add_trace(go.Scatter(x=Actives.index, y=Actives.values, mode =
'lines+markers', name = 'Active',
line = dict(color = "Orange", width = 2)))
fig.update_layout(title = 'Increase in number of cases worldwide',
xaxis_tickfont_size = 10,
                  yaxis = dict(title = 'Number of Cases'))

fig.show()
```

Observations:

A sharp rise in number of confirmed cases can be seen after 2 months of origin of coronavirus, the number of active cases are more compared to recovered cases, which is not good, but fortunately, the number of deaths are comparatively very less compared to the recovery case. This is a good sign.

```
# Finding countries which have more number of
cases(confirmed,deaths,recovery and active) currently as per the given
dataset

confirm_data =
confirm[['Country','Current']].sort_values('Current',ascending =
False)
deaths_data =
deaths[['Country','Current']].sort_values('Current',ascending = False)
recovery_data =
recovery[['Country','Current']].sort_values('Current',ascending =
False)
active_data =
active[['Country','Current']].sort_values('Current',ascending = False)

# Top 10 countries with more number of confirmed cases
confirm_data.head(10)
```

```
{"summary":"{\n  \"name\": \"confirm_data\",\n  \"rows\": 188,\n
\"fields\": [\n    {\n      \"column\": \"Country\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 188,\n        \"samples\": [\n          \"Saint
Kitts and Nevis\",\n          \"Mongolia\",\n          \"Colombia\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Current\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
296289,\n        \"min\": 9,\n        \"max\": 3364157,\n
\"num_unique_values\": 185,\n        \"samples\": [\n
109984,\n          40632,\n          317\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"confirm_data"}
```

```
# Top 10 countries with more number of death cases
deaths_data.head(10)
```

```
{"summary":"{\n  \"name\": \"deaths_data\",\n  \"rows\": 188,\n
\"fields\": [\n    {\n      \"column\": \"Country\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 188,\n        \"samples\": [\n
\"Eritrea\",\n          \"Brunei\",\n          \"Turkey\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Current\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
12617,\n        \"min\": 0,\n        \"max\": 135566,\n
\"num_unique_values\": 138,\n        \"samples\": [\n          84,\n
42,\n          9074\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"deaths_data"}
```

```
# Top 10 countries with more number of recovery cases
recovery_data.head(10)
```

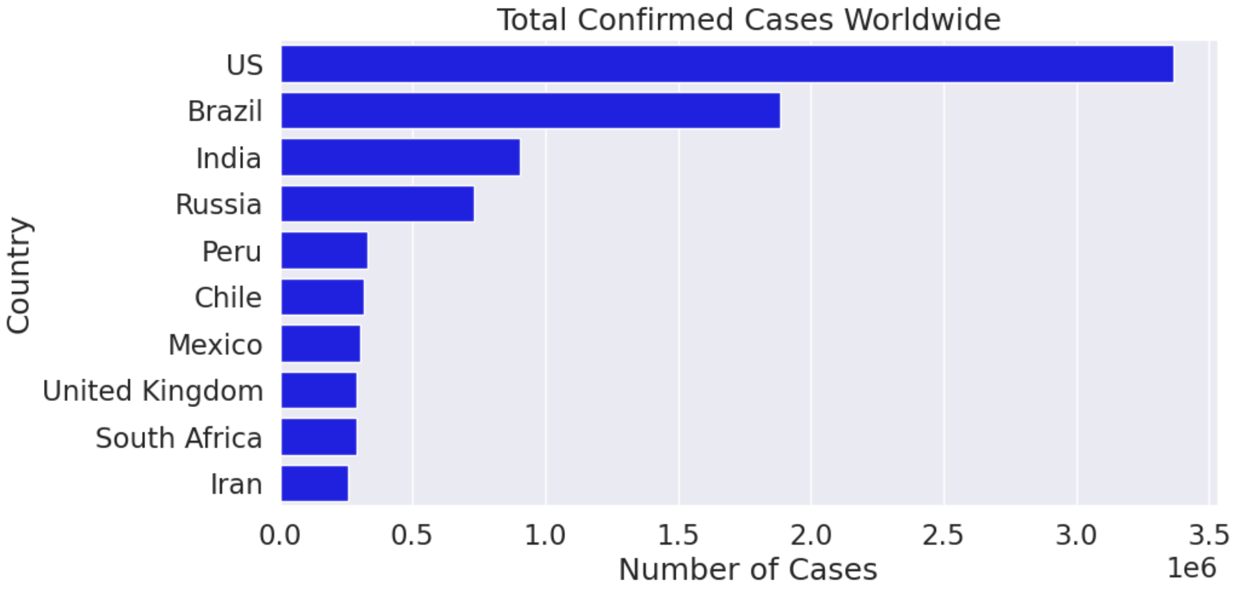
```
{"summary":"{\n  \"name\": \"recovery_data\",\n  \"rows\": 188,\n
\"fields\": [\n    {\n      \"column\": \"Country\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 188,\n        \"samples\": [\n
\"Seychelles\",\n          \"Eritrea\",\n          \"France\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Current\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
137838,\n        \"min\": 0,\n        \"max\": 1291251,\n
\"num_unique_values\": 183,\n        \"samples\": [\n          73381,\
n          21067,\n          155\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"recovery_data"}
```

```
# Top 10 countries with more number of active cases
active_data.head(10)
```

```
{"summary":"{\n  \"name\": \"active_data\",\n  \"rows\": 188,\n
\"fields\": [\n    {\n      \"column\": \"Country\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 188,\n        \"samples\": [\n
\"Timor-Leste\",\n          \"Comoros\",\n          \"Belgium\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Current\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
167616,\n        \"min\": 0,\n        \"max\": 2196652,\n
\"num_unique_values\": 170,\n        \"samples\": [\n          123,\n
24077,\n          404\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"active_data"}
```

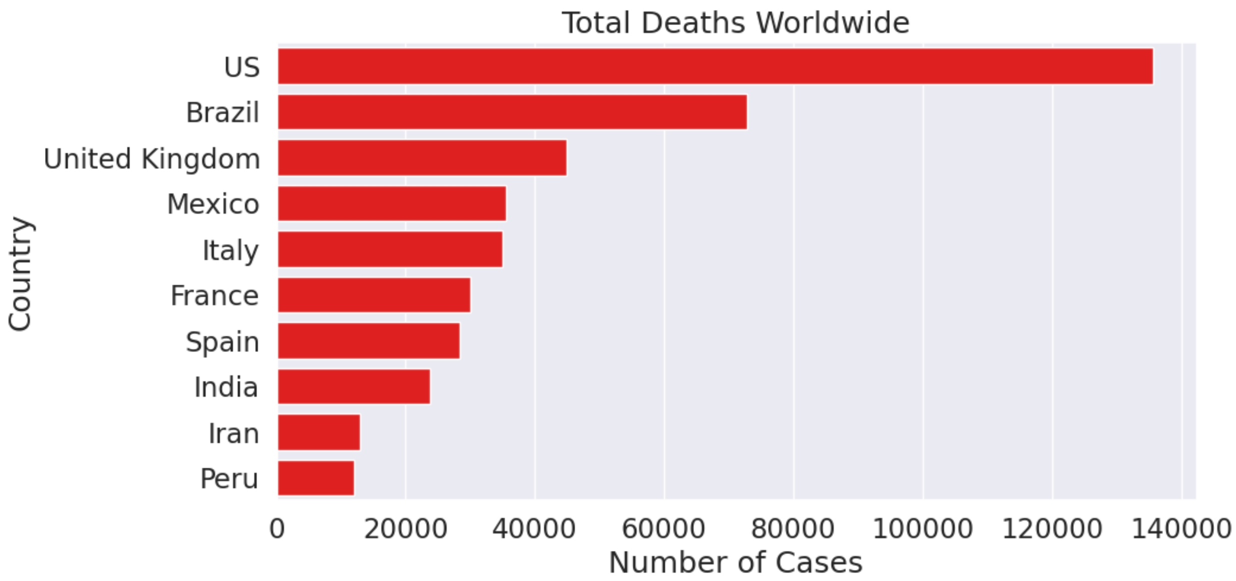
Now, we know the top 10 countries which are having more number of confirmed cases, death cases, recovery cases and active cases currently. We will visualize it to understand it clearly:

```
# Confirmed Cases
sns.set(font_scale=1.5)
plt.figure(figsize=(10,5))
fig= sns.barplot(x='Current', y='Country', data=confirm_data[:10],
orient='h',color='Blue')
plt.title('Total Confirmed Cases Worldwide')
fig.set(xlabel ='Number of Cases', ylabel ='Country')
plt.show()
```

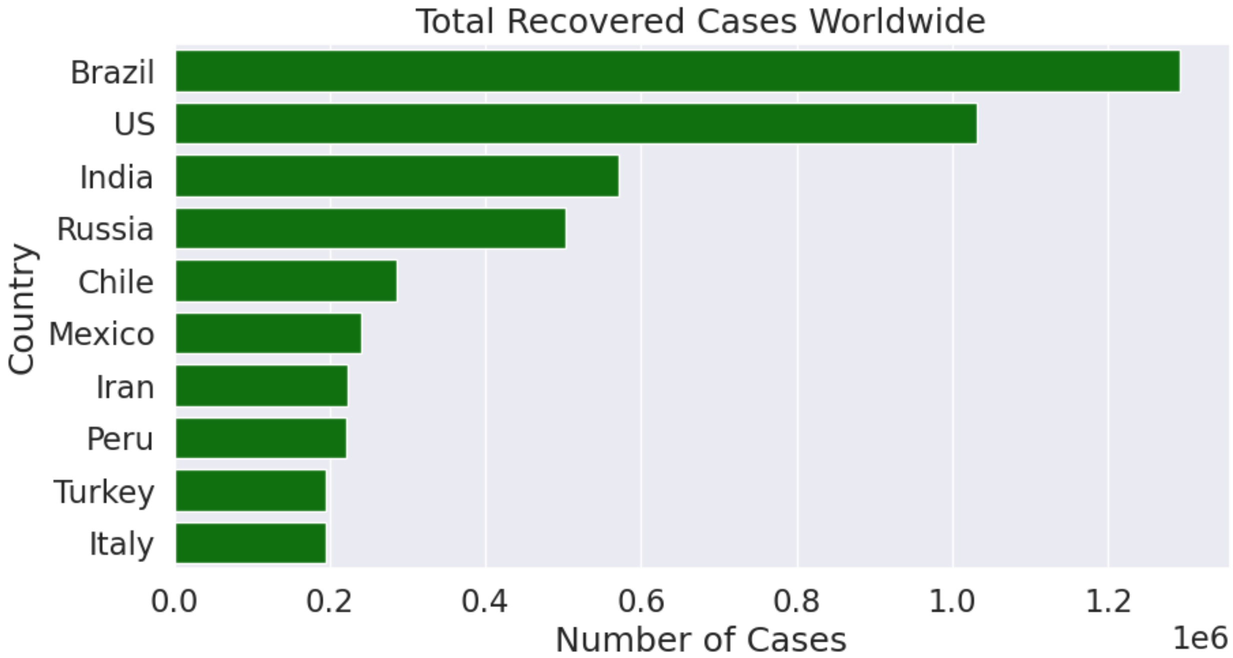


```
# Death Cases
plt.figure(figsize=(10,5))
fig= sns.barplot(x='Current', y='Country', data=deaths_data[:10],
orient='h',color='Red')
```

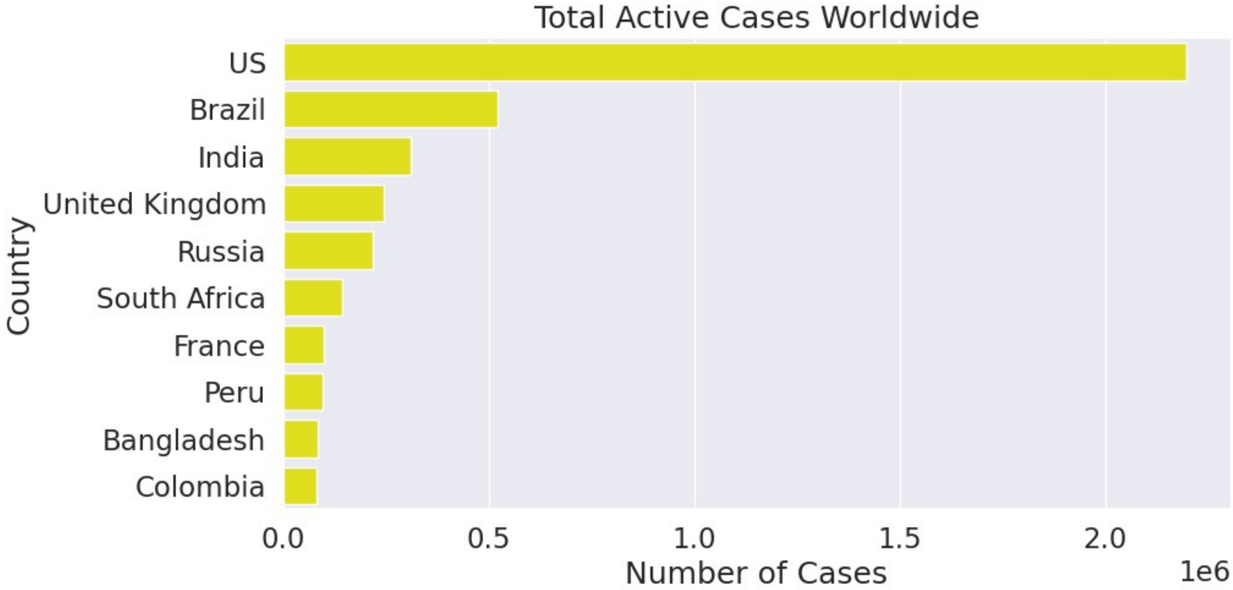
```
plt.title('Total Deaths Worldwide')
fig.set(xlabel ='Number of Cases', ylabel ='Country')
plt.show()
```



```
# Recovery Cases
plt.figure(figsize=(10,5))
fig= sns.barplot(x='Current', y='Country', data=recovery_data[:10],
orient='h',color='Green')
plt.title('Total Recovered Cases Worldwide')
fig.set(xlabel ='Number of Cases', ylabel ='Country')
plt.show()
```



```
# Active Cases
plt.figure(figsize=(10,5))
fig= sns.barplot(x='Current', y='Country', data=active_data[:10],
orient='h',color='Yellow')
plt.title('Total Active Cases Worldwide')
fig.set(xlabel ='Number of Cases', ylabel ='Country')
plt.show()
```



Observations:

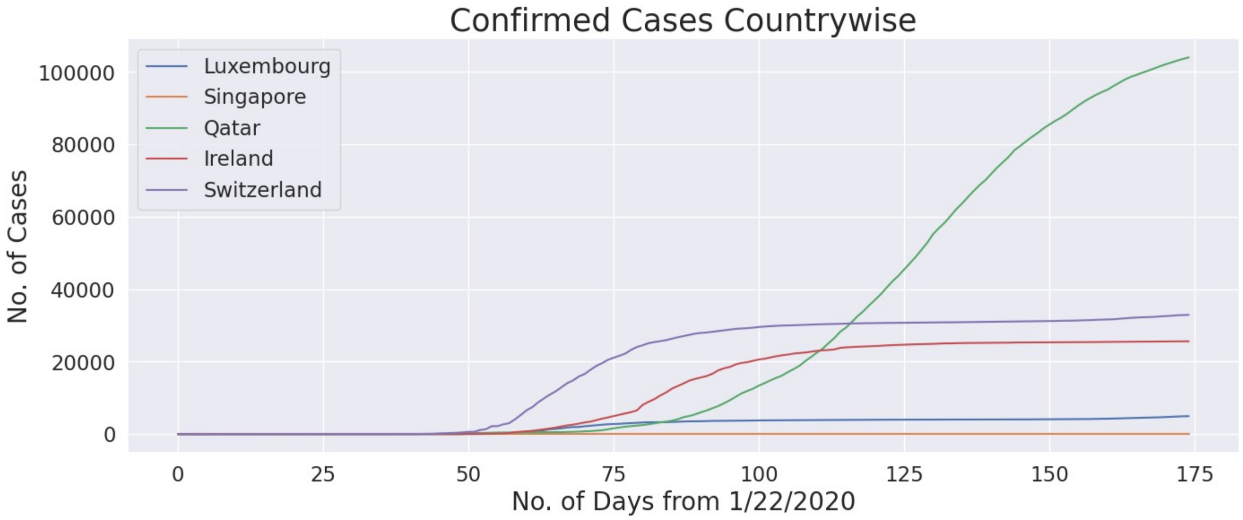
Countries like US, India, Brazil are having more number of confirm cases and death cases as per the given dataset. In active cases also, US, India are there in top 5 countries. This is a bad sign especially for US as well as India. But fortunately, in recovery cases, India and Brazil is on Top which is a good sign.

To find the impact of Covid-19 in wealthy countries:

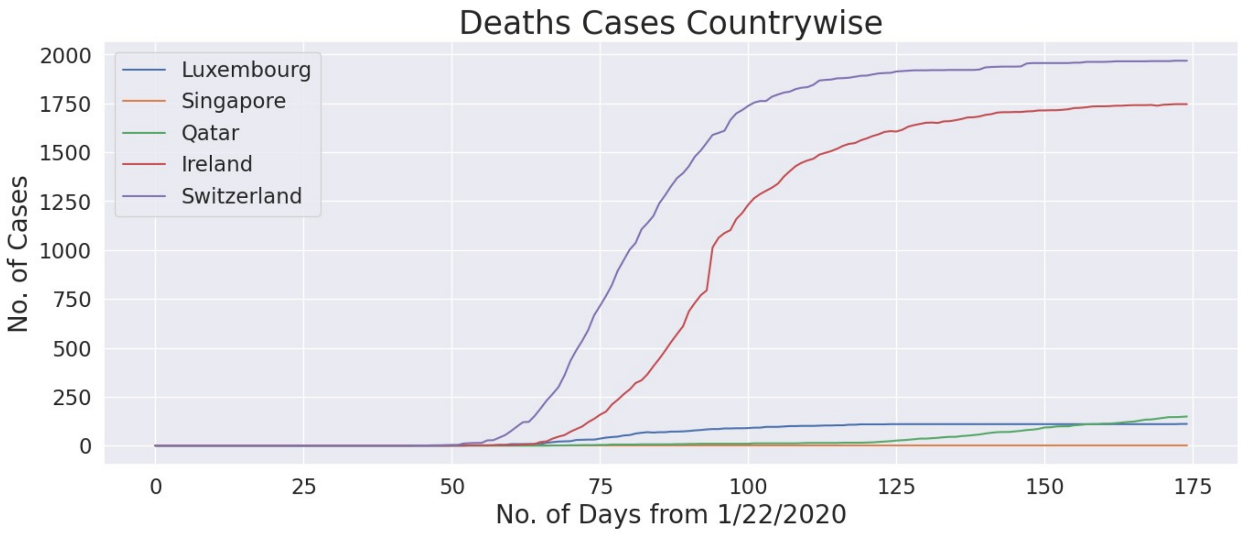
List of top 5 countries which are wealthy are: Luxembourg (GDP per capita :120,962.2), Singapore (GDP per capita :101,936.7), Qatar (GDP per capita :93,851.7), Ireland (GDP per capita :87,212) Switzerland (GDP per capita :70,726.6).

Now, I will find whether these 5 countries, which are wealthy are having more number of cases or not by using the given dataset.

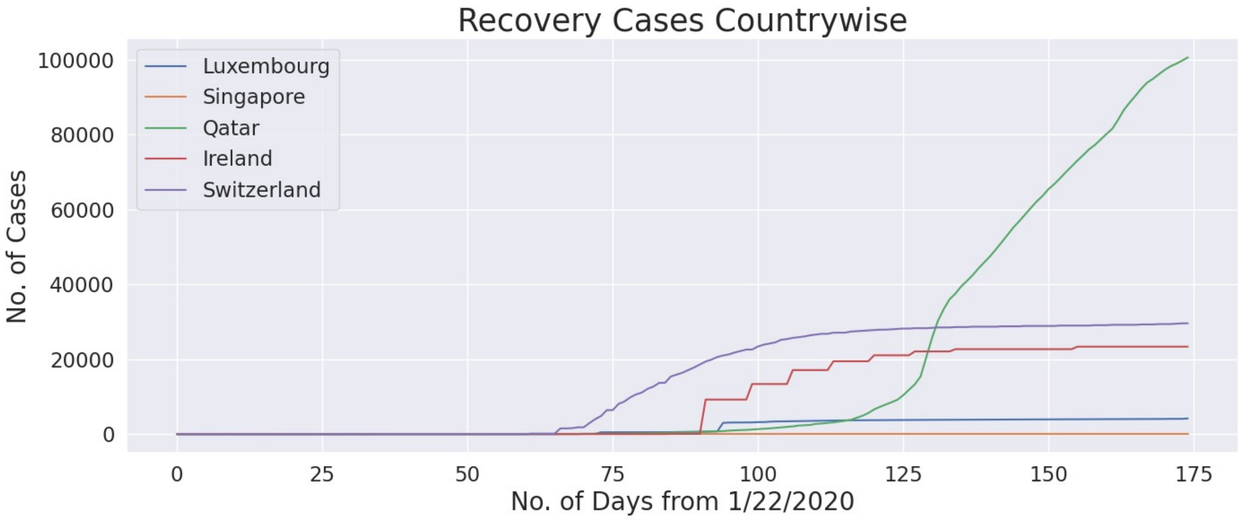
```
#Plotting the number of Confirmed cases countrywise

Luxembourg_confirm = confirm[confirm.Country ==
'Luxembourg'].iloc[:,1:].sum().values.tolist()
Singapore_confirm = confirm[confirm.Country == '
Singapore'].iloc[:,1:].sum().values.tolist()
Qatar_confirm = confirm[confirm.Country ==
'Qatar'].iloc[:,1:].sum().values.tolist()
Ireland_confirm = confirm[confirm.Country ==
'Ireland'].iloc[:,1:].sum().values.tolist()
Switzerland_confirm = confirm[confirm.Country ==
'Switzerland'].iloc[:,1:].sum().values.tolist()
plt.figure(figsize=(16,6))
plt.plot(Luxembourg_confirm)
plt.plot(Singapore_confirm)
plt.plot(Qatar_confirm)
plt.plot(Ireland_confirm)
plt.plot(Switzerland_confirm)
plt.title('Confirmed Cases Countrywise', size=25)
plt.xlabel('No. of Days from 1/22/2020', size=20)
plt.ylabel('No. of Cases', size=20)
plt.legend(['Luxembourg',
'Singapore','Qatar','Ireland','Switzerland'])
plt.show()
```



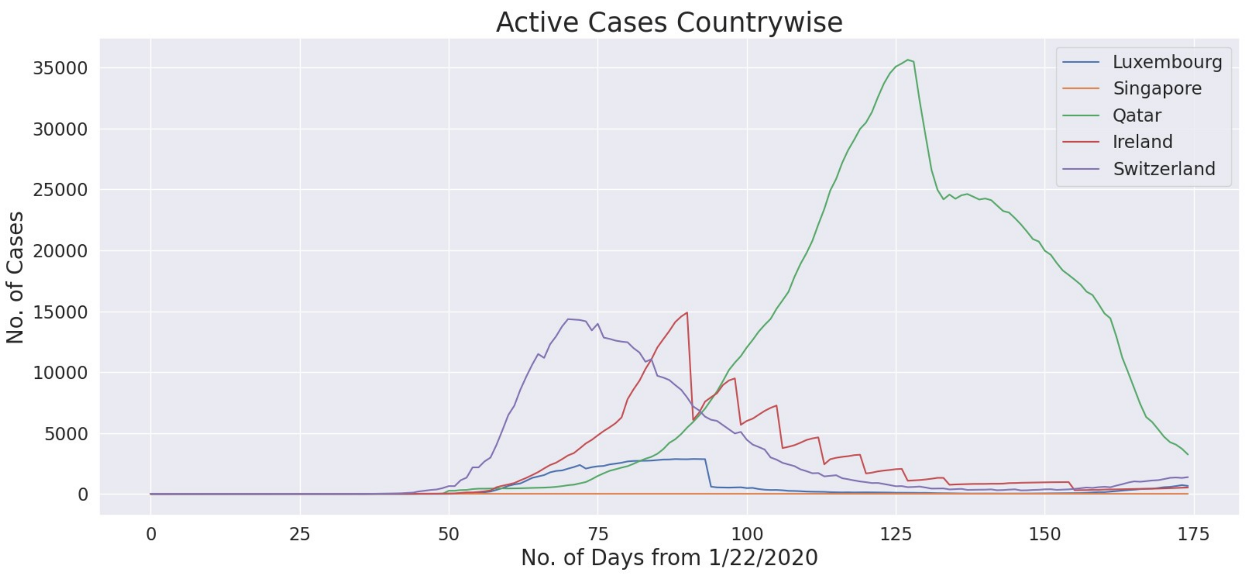
```
#Plotting the number of death cases countrywise
Luxembourg_deaths = deaths[deaths.Country ==
'Luxembourg'].iloc[:,1:].sum().values.tolist()
Singapore_deaths = deaths[deaths.Country == '
Singapore'].iloc[:,1:].sum().values.tolist()
Qatar_deaths = deaths[deaths.Country ==
'Qatar'].iloc[:,1:].sum().values.tolist()
Ireland_deaths = deaths[deaths.Country ==
'Ireland'].iloc[:,1:].sum().values.tolist()
Switzerland_deaths = deaths[deaths.Country ==
'Switzerland'].iloc[:,1:].sum().values.tolist()
plt.figure(figsize=(16,6))
plt.plot(Luxembourg_deaths)
plt.plot(Singapore_deaths)
plt.plot(Qatar_deaths)
plt.plot(Ireland_deaths)
plt.plot(Switzerland_deaths)
plt.title('Deaths Cases Countrywise', size=25)
plt.xlabel('No. of Days from 1/22/2020', size=20)
plt.ylabel('No. of Cases', size=20)
plt.legend(['Luxembourg',
'Singapore','Qatar','Ireland','Switzerland'])
plt.show()
```



```
#Plotting the number of recovery cases countrywise
Luxembourg_recovery = recovery[recovery.Country ==
'Luxembourg'].iloc[:,1:].sum().values.tolist()
Singapore_recovery = recovery[recovery.Country == '
Singapore'].iloc[:,1:].sum().values.tolist()
Qatar_recovery = recovery[recovery.Country ==
'Qatar'].iloc[:,1:].sum().values.tolist()
Ireland_recovery = recovery[recovery.Country ==
'Ireland'].iloc[:,1:].sum().values.tolist()
Switzerland_recovery = recovery[recovery.Country ==
'Switzerland'].iloc[:,1:].sum().values.tolist()
plt.figure(figsize=(16,6))
plt.plot(Luxembourg_recovery)
plt.plot(Singapore_recovery)
plt.plot(Qatar_recovery)
plt.plot(Ireland_recovery)
plt.plot(Switzerland_recovery)
plt.title('Recovery Cases Countrywise', size=25)
plt.xlabel('No. of Days from 1/22/2020', size=20)
plt.ylabel('No. of Cases', size=20)
plt.legend(['Luxembourg',
'Singapore','Qatar','Ireland','Switzerland'])
plt.show()
```



```
#Plotting the number of active cases countrywise
plt.figure(figsize=(19,8))
Luxembourg_active = active[active.Country ==
'Luxembourg'].iloc[:,1:].sum().values.tolist()
Singapore_active = active[active.Country == '
Singapore'].iloc[:,1:].sum().values.tolist()
Qatar_active = active[active.Country ==
'Qatar'].iloc[:,1:].sum().values.tolist()
Ireland_active = active[active.Country ==
'Ireland'].iloc[:,1:].sum().values.tolist()
Switzerland_active = active[active.Country ==
'Switzerland'].iloc[:,1:].sum().values.tolist()

plt.plot(Luxembourg_active)
plt.plot(Singapore_active)
plt.plot(Qatar_active)
plt.plot(Ireland_active)
plt.plot(Switzerland_active)
plt.title('Active Cases Countrywise', size=25)
plt.xlabel('No. of Days from 1/22/2020', size=20)
plt.ylabel('No. of Cases', size=20)
plt.legend(['Luxembourg',
'Singapore','Qatar','Ireland','Switzerland'])
plt.show()
```



Observations:

In Switzerland, number of confirmed cases and active cases are more but death cases are less and recovery cases are high. Singapore is having more impact of Covid-19 whereas Luxembourg is having less impact of Covid-19 as compared to other countries. Although in Qatar, there are more number of confirmed cases, recovery cases in Qatar are more, death cases and active cases are less in Qatar. In Ireland, number of confirmed cases and active cases are more!

Which countries are safe and which are not safe:

With the help of the given dataset, I have shown in worldmap, which will be easier to know which countries are having more impact of Covid-19 and which countries are having less impact of Covid-19.

```
# Showing Confirmed Cases in worldmap
fig = px.choropleth(confirm_data, locations="Country", 
locationmode='country names',
                  color=confirm_data['Current'], hover_name="Country", 
                  title='Countries with Confirmed 
Cases',hover_data=['Current'], color_continuous_scale="blues")
fig.show()

# Showing Death Cases in worldmap
fig = px.choropleth(deaths_data, locations="Country", 
locationmode='country names',
                  color=deaths_data['Current'], hover_name="Country", 
                  title='Countries with Death
Cases',hover_data=['Current'], color_continuous_scale="peach")
fig.show()

# Showing Recovered Cases in worldmap
fig = px.choropleth(recovery_data, locations="Country",
locationmode='country names',
                  color=recovery_data['Current'],
hover_name="Country",
                  title='Countries with Recovered
Cases',hover_data=['Current'], color_continuous_scale="emrld")
fig.show()

# Showing Active Cases in worldmap
fig = px.choropleth(active_data, locations="Country",
locationmode='country names',
                  color=active_data['Current'], hover_name="Country",
                  title='Countries with Active
Cases',hover_data=['Current'], color_continuous_scale="sunset")
fig.show()
```

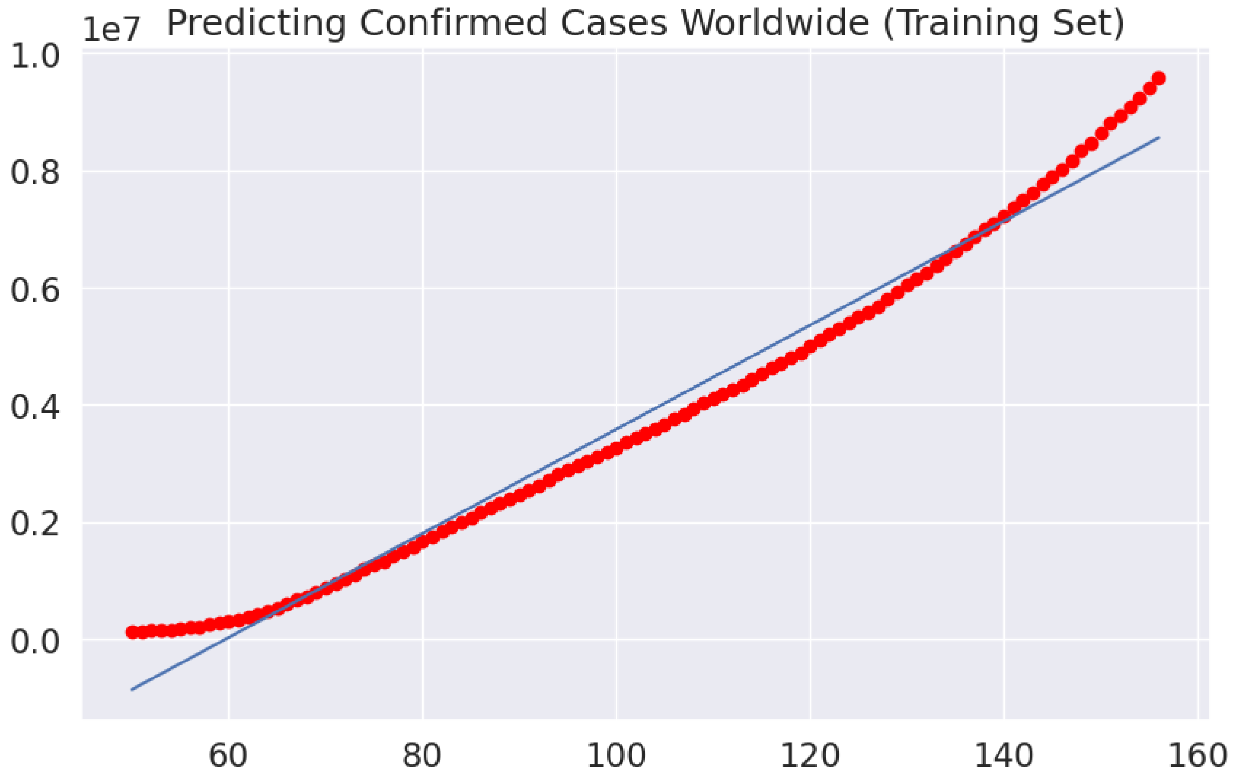
Observations:

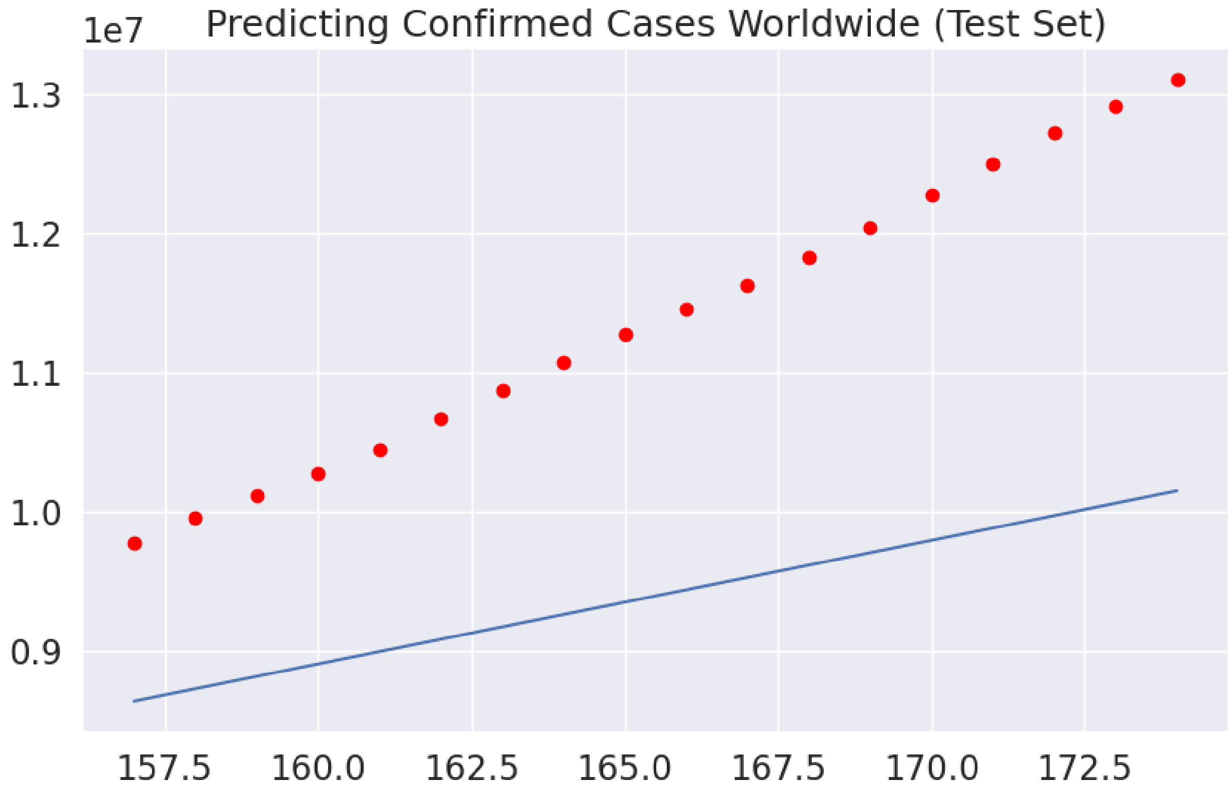
Countries like India, US, Brazil are having more number of confirmed cases as well as death cases, and in countries like China, Russia, Australia, number of confirmed cases and death cases are low. In US, number of recovery cases are too low but active cases are too high. In India and Brazil, it is opposite, there are less number of active cases and more number of recovery cases. In Russia also, number of recovery cases are more than active cases. While in countries like China, Australia, number of neither the number of active cases are high nor the recovery cases.

Prediction:

```
#Linear Regression

total_confirm = np.array(confirm_date).reshape(-1,1)
total_deaths = np.array(deaths_date).reshape(-1,1)
total_recovery = np.array(recovery_date).reshape(-1,1)
total_active = np.array(active_date).reshape(-1,1)
days = [ i for i in range(confirm.shape[1] - 1) ]
dates = np.array([i for i in range(len(days))]).reshape(-1, 1)

def linear_plot(x,y,reg,title):
    plt.figure(figsize=(10,6))
    plt.scatter(x,y,color='red')
    plt.plot(x,reg)
    plt.title(title)
```

```
X_train_confirmed, X_test_confirmed, y_train_confirmed,
y_test_confirmed = train_test_split(dates[50:],
total_confirm[50:], test_size=0.14, shuffle=False)

reg = LinearRegression()
reg.fit(X_train_confirmed, y_train_confirmed);

# Plot training set
linear_plot(X_train_confirmed,y_train_confirmed,reg.predict(X_train_co
nfirmed),
'Predicting Confirmed Cases Worldwide (Training Set)')

# Plot test set
linear_plot(X_test_confirmed,y_test_confirmed,reg.predict(X_test_confi
rmed),
'Predicting Confirmed Cases Worldwide (Test Set)')
```





Observations:

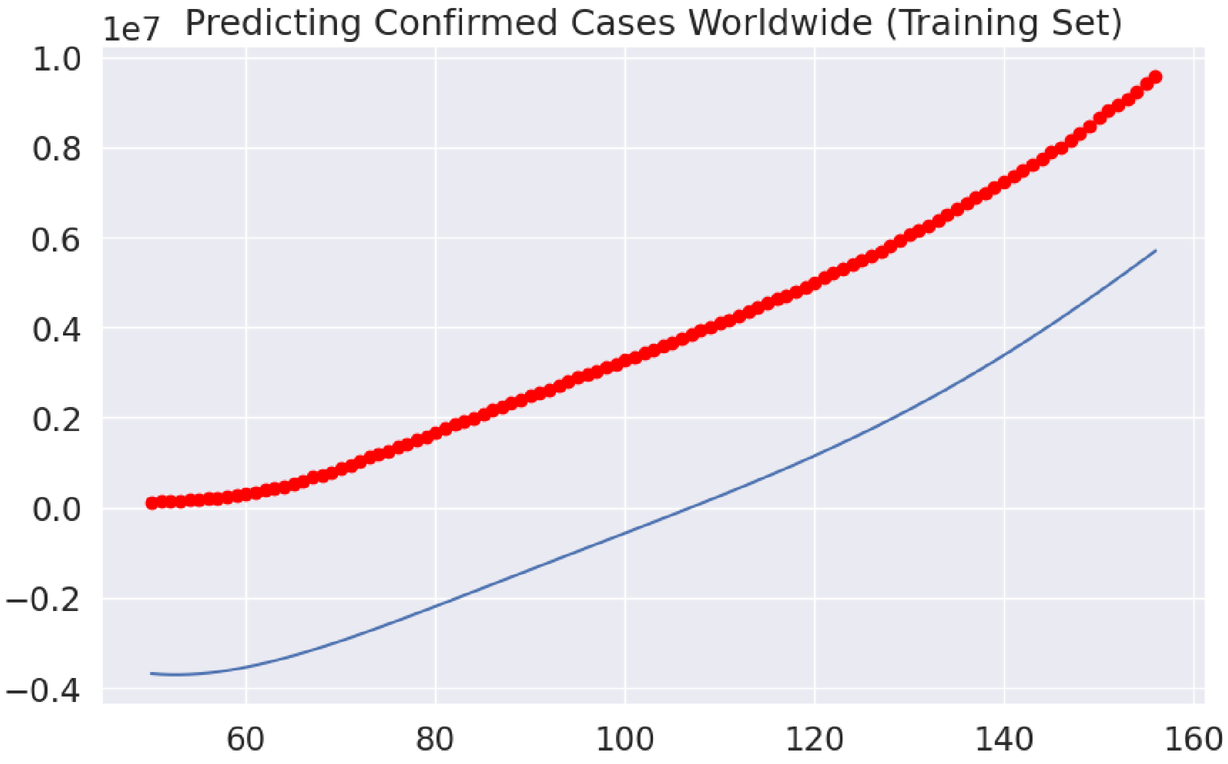
The test set predictions are not very accurate as training set predictions. As the total confirmed cases has a parabolic curve, trying polynomial linear regression.

Polynomial Linear Regression:

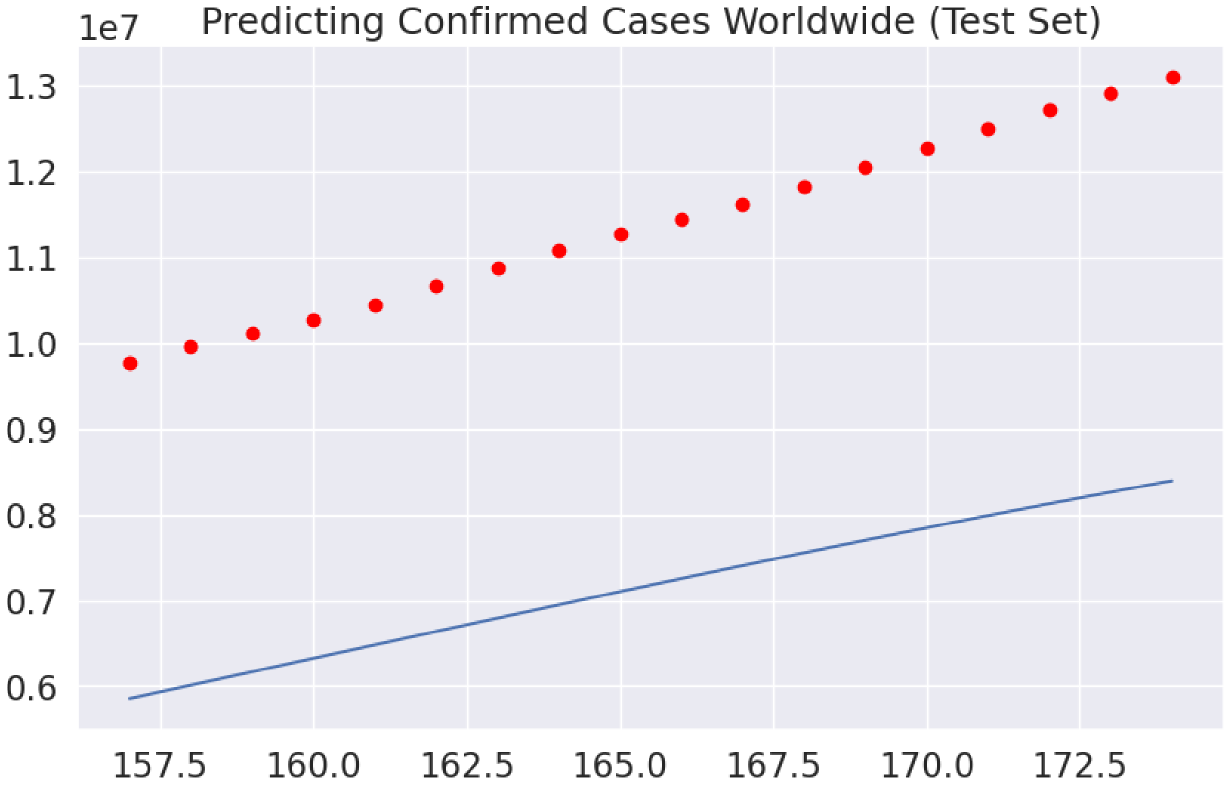
Confirmed cases;

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Assuming X_train_confirmed, X_test_confirmed, y_train_confirmed are
your datasets

# Step 1: Transform the data for polynomial regression
poly = PolynomialFeatures(degree=5)
poly_X_train_confirmed = poly.fit_transform(X_train_confirmed)
poly_X_test_confirmed = poly.transform(X_test_confirmed)
```

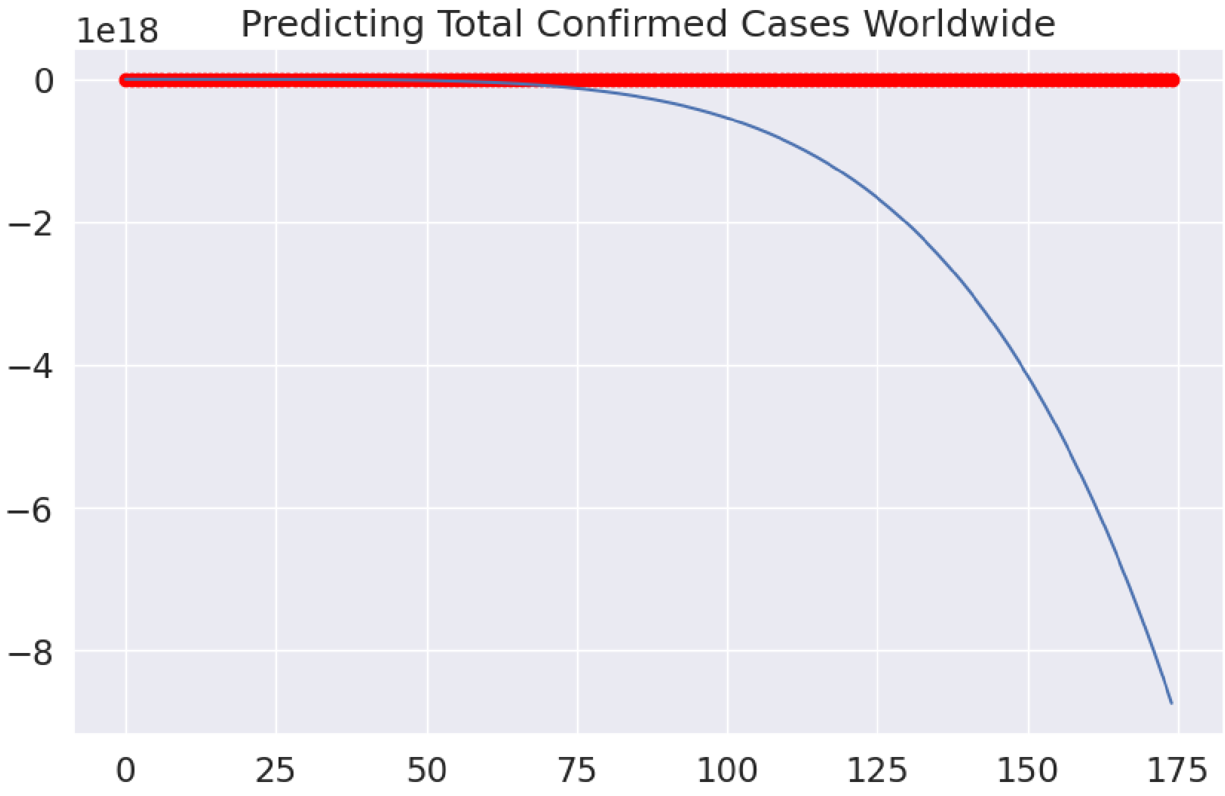
```
# Step 2: Normalize the data if necessary (optional but recommended)
scaler = StandardScaler()
poly_X_train_confirmed = scaler.fit_transform(poly_X_train_confirmed)
poly_X_test_confirmed = scaler.transform(poly_X_test_confirmed)

# Step 3: Perform polynomial regression
poly_reg = LinearRegression(fit_intercept=False)
poly_reg.fit(poly_X_train_confirmed, y_train_confirmed)

# Predicting the results
y_pred_confirmed = poly_reg.predict(poly_X_test_confirmed)

# Displaying results (optional)
print("Coefficients:", poly_reg.coef_)
print("Intercept:", poly_reg.intercept_)

Coefficients: [[ 0.00000000e+00 -4.48852817e+07  1.89051551e+08 -
2.95357202e+08
   2.09953040e+08 -5.60128416e+07]]
Intercept: 0.0

# Plot training set
linear_plot(X_train_confirmed,y_train_confirmed,poly_reg.predict(poly_
X_train_confirmed),
'Predicting Confirmed Cases Worldwide (Training Set)')

# Plot test set
linear_plot(X_test_confirmed,y_test_confirmed,poly_reg.predict(poly_X_
test_confirmed),
'Predicting Confirmed Cases Worldwide (Test Set)')

# Plot total cases
linear_plot(dates,total_confirm,poly_reg.predict(poly.fit_transform(da
tes)),
'Predicting Total Confirmed Cases Worldwide')
```



Predicting Confirmed Cases Worldwide (Training Set)



Predicting Confirmed Cases Worldwide (Test Set)



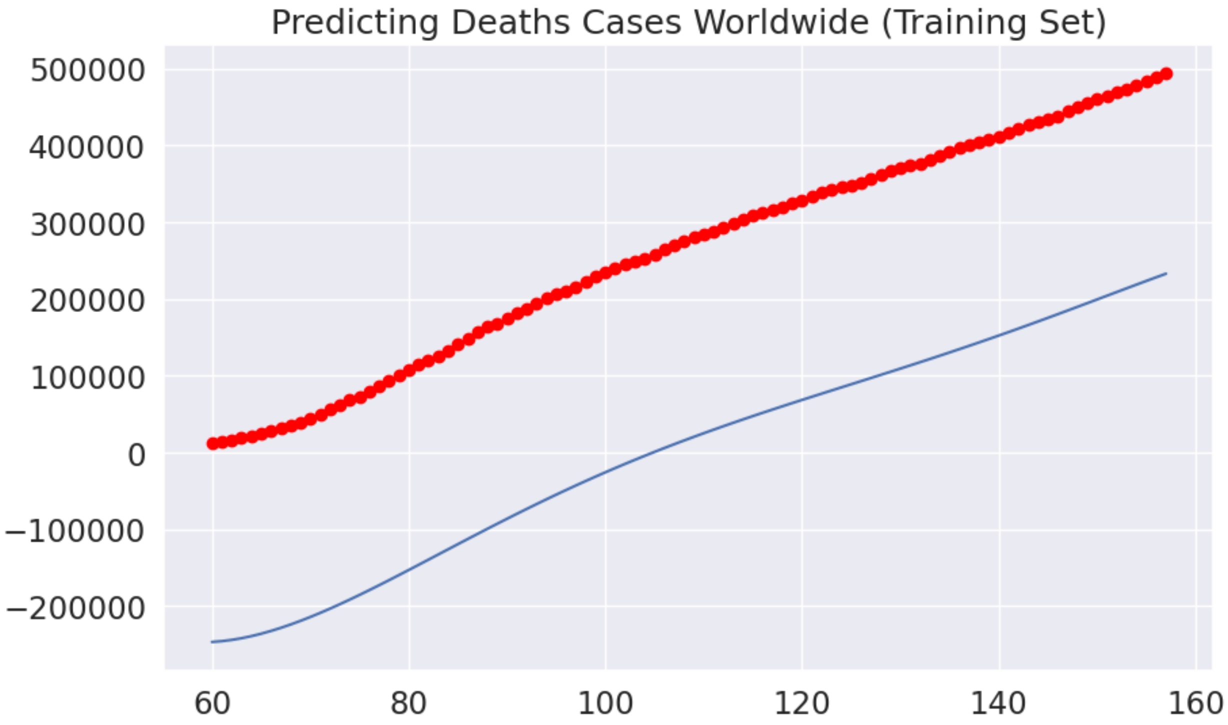
Death Cases:

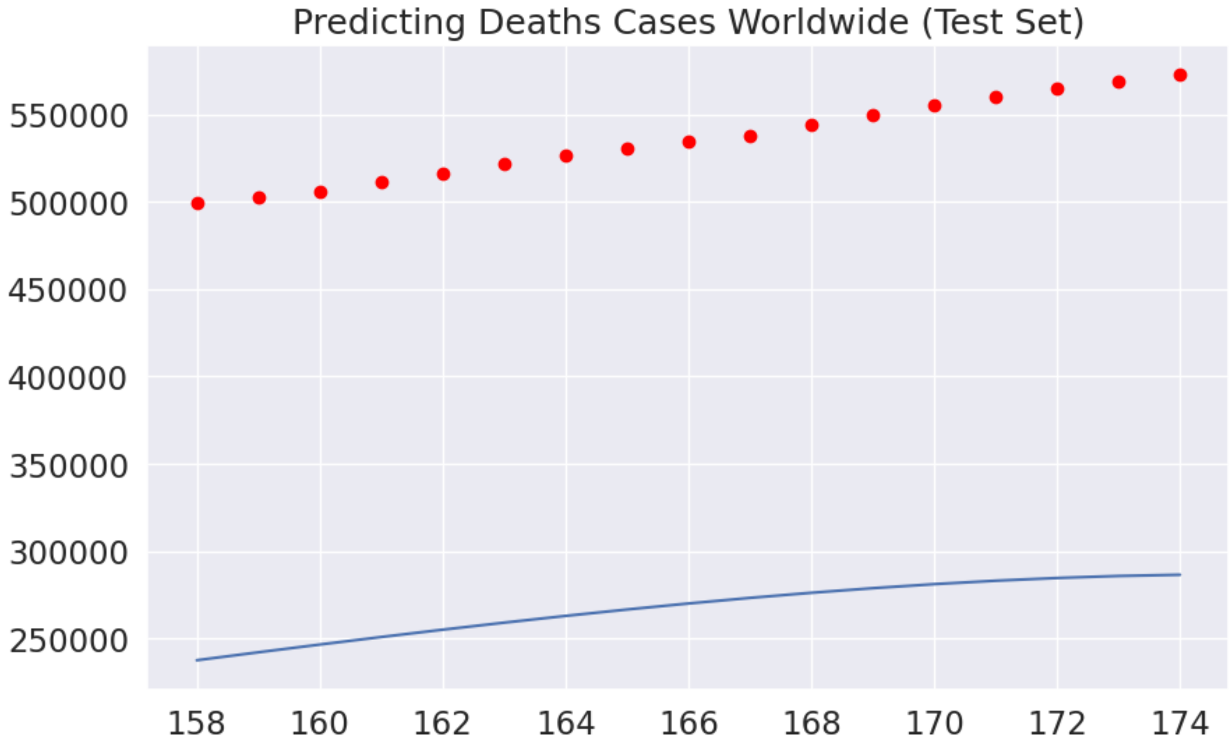
```
#For Death Cases
X_train_deaths, X_test_deaths, y_train_deaths, y_test_deaths =
train_test_split(dates[60:], total_deaths[60:],
test_size=0.14, shuffle=False)

import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Assuming X_train_deaths, X_test_deaths, y_train_deaths are your
datasets

# Step 1: Transform the data for polynomial regression
poly = PolynomialFeatures(degree=5)
poly_X_train_deaths = poly.fit_transform(X_train_deaths)
poly_X_test_deaths = poly.transform(X_test_deaths)

# Step 2: Normalize the data (optional but recommended)
scaler = StandardScaler()
poly_X_train_deaths = scaler.fit_transform(poly_X_train_deaths)
poly_X_test_deaths = scaler.transform(poly_X_test_deaths)
```

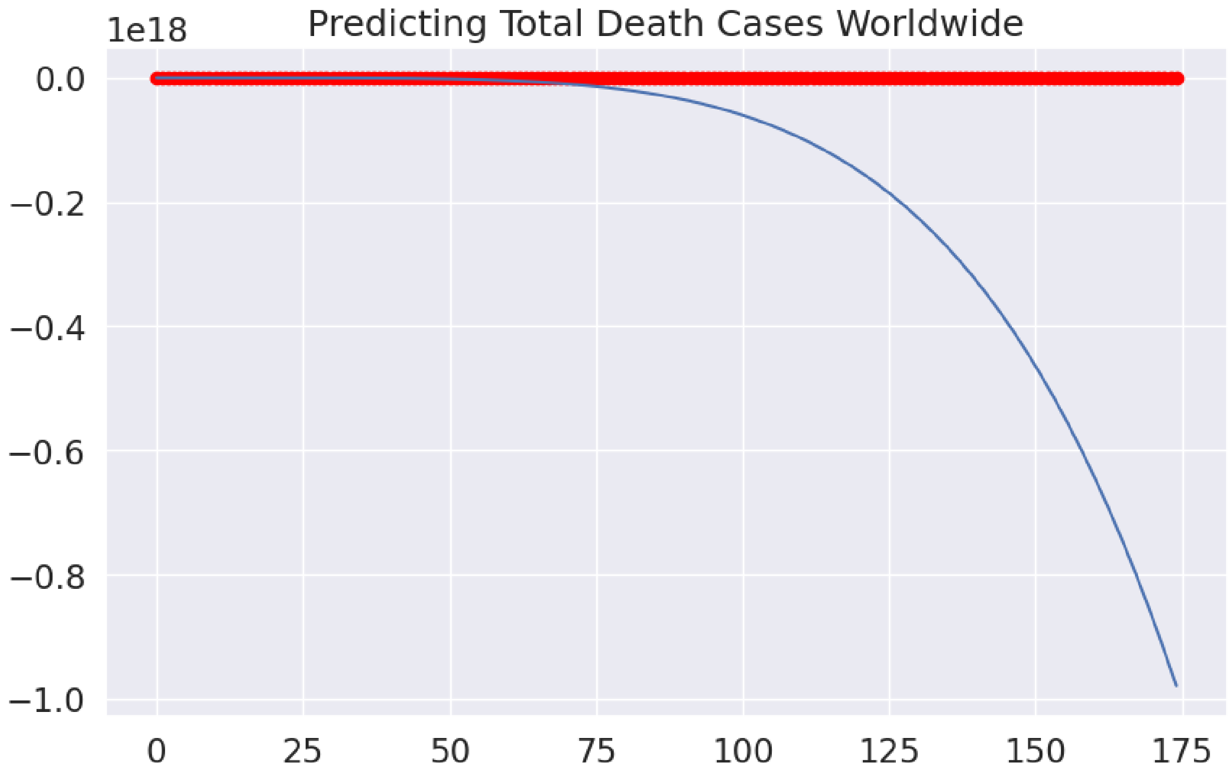
```
# Step 3: Perform polynomial regression
poly_reg = LinearRegression(fit_intercept=False)
poly_reg.fit(poly_X_train_deaths, y_train_deaths)

# Predicting the results
y_pred_deaths = poly_reg.predict(poly_X_test_deaths)

# Displaying results (optional)
print("Coefficients:", poly_reg.coef_)
print("Intercept:", poly_reg.intercept_)
```

```
Coefficients: [[        0.          -5520171.24700862
22902486.57492261
  -35114027.13823323  24146028.86250534  -6273656.88120934]]
Intercept: 0.0
```

```
# Plot training set
linear_plot(X_train_deaths,y_train_deaths,poly_reg.predict(poly_X_trai
n_deaths),'Predicting Deaths Cases Worldwide (Training Set)')

# Plot test set
linear_plot(X_test_deaths,y_test_deaths,poly_reg.predict(poly_X_test_d
eaths),'Predicting Deaths Cases Worldwide (Test Set)')

# Plot total cases
linear_plot(dates,total_deaths,poly_reg.predict(poly.fit_transform(dat
es)),'Predicting Total Death Cases Worldwide')
```




Predicting Deaths Cases Worldwide (Test Set)
550000
500000
450000
400000
350000
300000
250000
158
160
162
164
166
168
170
172
174


Predicting Total Death Cases Worldwide
1e18
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
25
50
75
100
125
150
175