In [1]:
```
#import packages
from __future__ import print_function
import numpy as np
import pandas as pd
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
```

In [2]:
```
death_df = pd.read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv")
confirmed_df = pd.read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv")
recovered_df = pd.read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_recovered_global.csv")
country_df = pd.read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/web-data/data/cases_country.csv")
```

Data Cleansing

In [3]:
```
death_df.head()
```

Out[3]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0

5 rows × 139 columns

In [4]:
```
confirmed_df.head()
```

Out[4]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0

5 rows × 139 columns

In [5]:
```
recovered_df.head()
```

Out[5]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0

5 rows × 139 columns

In [6]:
```
country_df.head()
```

Out[6]:

	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	Inci
0	Australia	2020-06-06 01:32:55	-25.0000	133.0000	7251	102	6688.0	461	
1	Austria	2020-06-06 01:32:55	47.5162	14.5501	16843	672	15742.0	429	1
2	Canada	2020-06-06 01:32:55	60.0010	-95.0010	95947	7778	53068.0	35101	2
3	China	2020-06-06 01:32:55	30.5928	114.3055	84177	4638	79420.0	119	
4	Denmark	2020-06-06 01:32:55	56.2639	9.5018	12075	586	10853.0	636	2

Data Munging

In [7]:
```
#Rename columns
death_df.columns = map(str.lower, death_df.columns)
confirmed_df.columns = map(str.lower, confirmed_df.columns)
recovered_df.columns = map(str.lower, recovered_df.columns)
country_df.columns = map(str.lower, country_df.columns)
```

In [8]:
```
death_df = death_df.rename(columns = {'province/state' : 'state', 'count
ry/region' : 'country'})
confirmed_df = confirmed_df.rename(columns = {'province/state' : 'state'
, 'country/region' : 'country'})
recovered_df = recovered_df.rename(columns = {'province/state' : 'state'
, 'country/region' : 'country'})
country_df = country_df.rename(columns = {'country_cegion' : 'country'})
```

In [9]:
```
sorted_country_df = country_df.sort_values('confirmed', ascending=False)
.head(10)
sorted_country_df = sorted_country_df.rename(columns = {'country_region'
: 'country'})
```

In [10]:
```
sorted_country_df
```

Out[10]:

	country	last_update	lat	long_	confirmed	deaths	recovered	active	incid
17	US	2020-06-06 01:32:55	40.000000	-100.000000	1897239	109127	491706.0	1344028	575
21	Brazil	2020-06-06 01:32:55	-14.235000	-51.925300	614941	34021	NaN	580920	289
13	Russia	2020-06-06 01:32:55	61.524000	105.318800	449256	5520	212237.0	231499	307
16	United Kingdom	2020-06-06 01:32:55	55.000000	-3.000000	284734	40344	1228.0	243162	419
18	Spain	2020-06-06 01:32:55	40.463667	-3.749220	240978	27134	150376.0	63468	515
93	India	2020-06-06 01:32:55	20.593684	78.962880	236184	6649	113233.0	116302	17
10	Italy	2020-06-06 01:32:55	41.871900	12.567400	234531	33774	163781.0	36976	387
6	France	2020-06-06 01:32:55	46.227600	2.213700	190180	29114	70622.0	90444	291
22	Peru	2020-06-06 01:32:55	-9.190000	-75.015200	187400	5162	79214.0	103024	568
7	Germany	2020-06-06 01:32:55	51.165691	10.451526	184924	8658	168480.0	7786	220

In [11]:
```
def highlight_col(x):
    r = 'background-color : red'
    p = 'background-color : purple'
    y = 'background-color : yellow'
    temp_df = pd.DataFrame('', index=x.index, columns = x.columns)
    temp_df.iloc[:,4] = p
    temp_df.iloc[:,5] = r
    temp_df.iloc[:,6] = y
    return temp_df
sorted_country_df.head(5).style.apply(highlight_col, axis=None)
```

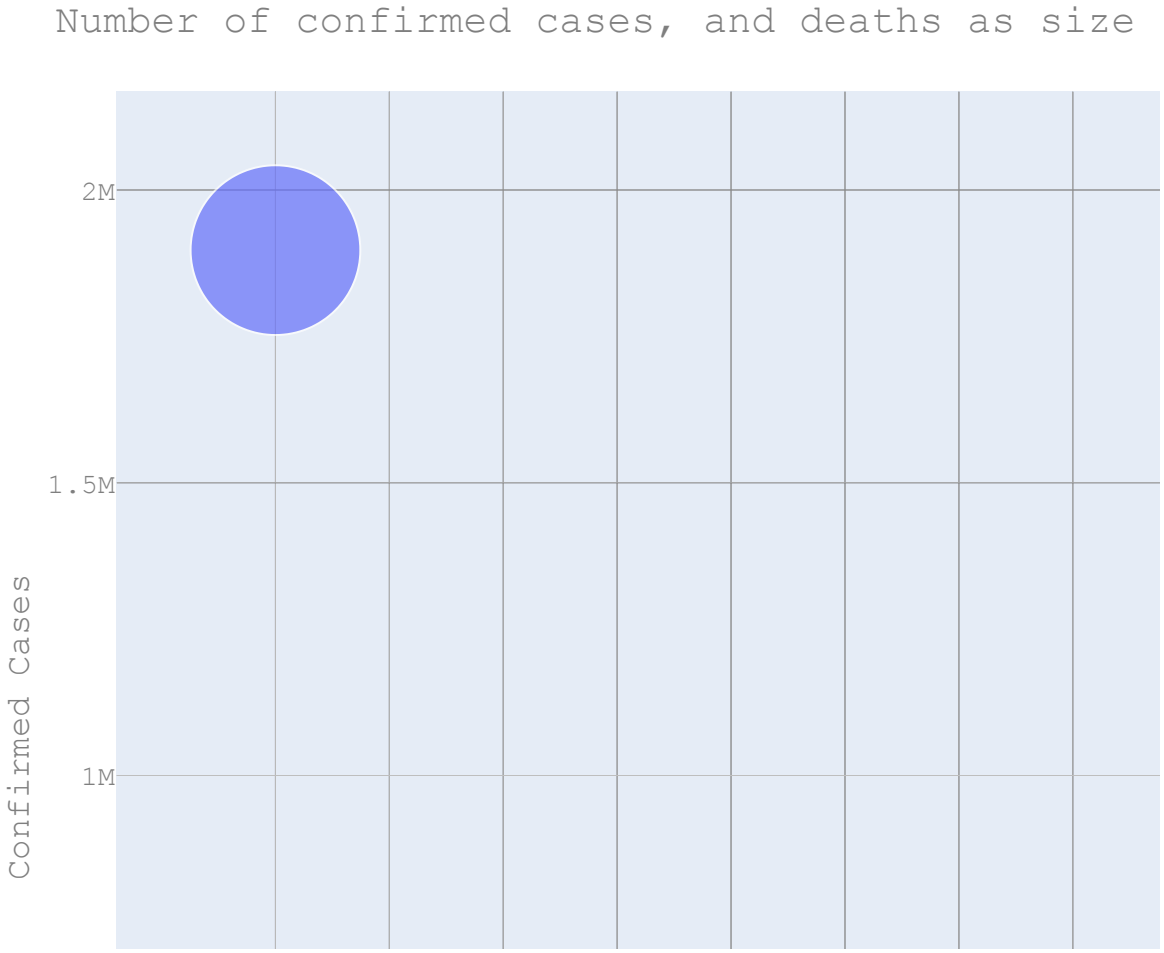
Out[11]:

	country	last_update	lat	long_	confirmed	deaths	recovered	active	incident_ra
17	US	2020-06-06 01:32:55	40	-100	1897239	109127	491706	1344028	575.85
21	Brazil	2020-06-06 01:32:55	-14.235	-51.9253	614941	34021	nan	580920	289.30
13	Russia	2020-06-06 01:32:55	61.524	105.319	449256	5520	212237	231499	307.84
16	United Kingdom	2020-06-06 01:32:55	55	-3	284734	40344	1228	243162	419.4
18	Spain	2020-06-06 01:32:55	40.4637	-3.74922	240978	27134	150376	63468	515.40

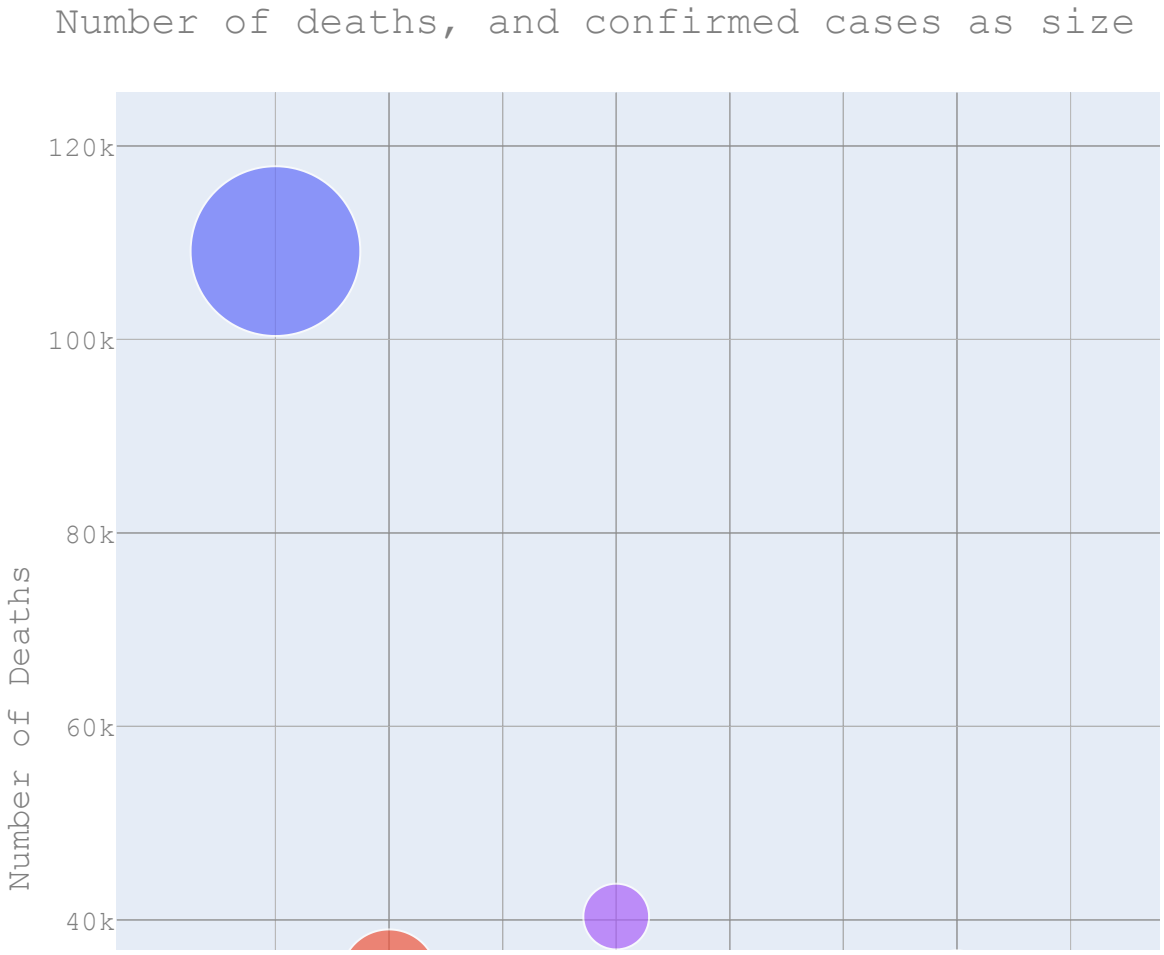
Data Visualizations

In [12]:
```
import plotly.express as px
```

In [13]:
```
fig = px.scatter(sorted_country_df.head(10), x='country', y='confirmed',
size='deaths', color='country',
                 hover_name='country', size_max=60)
fig.update_layout(
    height=800,
    title_text='Number of confirmed cases, and deaths as size of plot wi
thin each country',
    xaxis_title='Country',
    yaxis_title='Confirmed Cases',
    font=dict(
        family="Courier New, monospace",
        size=14,
        color="#7f7f7f")
)
fig.show()
```


Number of confirmed cases, and deaths as size
Confirmed Cases
2M
1.5M
1M

In [14]:
```
fig = px.scatter(sorted_country_df.head(10), x='country', y='deaths', si
ze='confirmed', color='country',
                 hover_name='country', size_max=60)
fig.update_layout(
    height=800,
    title_text='Number of deaths, and confirmed cases as size of plot wi
thin each country',
    xaxis_title='Country',
    yaxis_title='Number of Deaths',
    font=dict(
        family="Courier New, monospace",
        size=14,
        color="#7f7f7f")
)
fig.show()
```


Number of deaths, and confirmed cases as size
Number of Deaths
120k
100k
80k
60k
40k

In [15]:
```
import plotly.graph_objects as go

def plot_cases_for_country(country):
    labels = ['confirmed', 'deaths']
    colors = ['blue', 'red']
    mode_size = [6, 8]
    line_size = [4, 5]
    df_list = [confirmed_df, death_df]

    fig = go.Figure()

    for i, df in enumerate(df_list):
        if country == 'World' or country == 'world':
            x_data = np.array(list(df.iloc[:, 5:].columns))
            y_data = np.sum(np.asarray(df.iloc[:, 5:]), axis=0)
        else:
            x_data = np.array(list(df.iloc[:, 5:].columns))
            y_data = np.sum(np.asarray(df[df['country']==country].iloc
[:, 5:]), axis=0)
        fig.add_trace(go.Scatter(x=x_data, y=y_data, mode='lines+marker
s',
                                 name=labels[i],
                                 line=dict(color=colors[i], width=line_si
ze[i]),
                                 connectgaps=True,
                                 text='Total' + str(labels[i]) + ":" + st
r(y_data[-1])))
    fig.update_layout(yaxis_type="log")
    fig.show()

interact(plot_cases_for_country, country='World')
```

Out[15]:
```
<function __main__.plot_cases_for_country(country)>
```

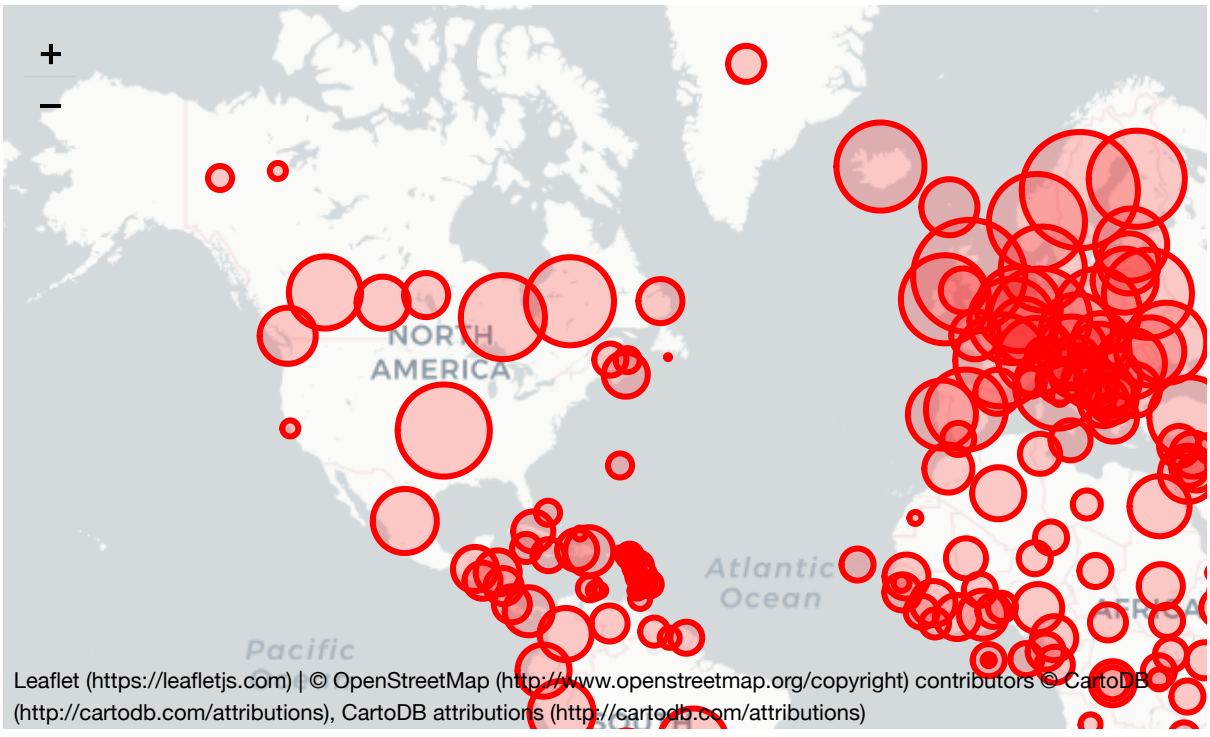
In [16]:
```
#leaflet maps
import folium
```

In [69]:
```
world_map = folium.Map(location=[11,0], tiles='cartodbpositron', zoom_st
art=2, max_zoom=6,
                       )

for i in range(len(confirmed_df)):
    folium.Circle(
    location=[confirmed_df.iloc[i]['lat'], confirmed_df.iloc[i]['long'
]],
    fill=True,
    tooltip = "<div style = 'margin:0; background-color:black; color:whi
te;'>" +
                "<h4 style = 'text-align:center; font-weight:bold;'>" +
confirmed_df.iloc[i]['country'] + "</h4>"
                "<hr style = 'argin:10px; color:white;'>" +
                "<ul style = 'color:white;;ist-style-type:circle; align-
item:left; padding-left:20px; padding-right:20px;'>" +
                    "<li>Confirmed: "+ str(confirmed_df.iloc[i,-1]) + "
</li>" +
                    "<li>Deaths: "+ str(death_df.iloc[i,-1]) + "</li>"
+
                    "<li>Death Rate: "+str(np.round(death_df.iloc[i,-1]
/(confirmed_df.iloc[i,-1]+1.00001)*100,2)) + "</li>" +
                "</ul></div>",
    radius=(int(np.log(confirmed_df.iloc[i, -1]+1.00001)) + 0.2)* 50000,
    #fill_color='blue',
    color='red').add_to(world_map)

world_map
```

Out[69]:



Machine Learning and Forecasting

In [18]:
```
covid=pd.read_csv("https://raw.githubusercontent.com/IsaiahAim/Covid19-Visualization-and-Time-Series/master/covid_19_data.csv")
```

In [19]:
```
covid.head()
```

Out[19]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recove
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	

In [20]:
```
print("Size/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)
```

```
Size/Shape of the dataset:  (29426, 8)
Checking for null values:
 SNo                    0
ObservationDate         0
Province/State      14899
Country/Region          0
Last Update             0
Confirmed               0
Deaths                  0
Recovered               0
dtype: int64
Checking Data-type of each column:
 SNo                   int64
ObservationDate       object
Province/State        object
Country/Region        object
Last Update           object
Confirmed            float64
Deaths               float64
Recovered            float64
dtype: object
```

In [21]:
```
covid.drop(["SNo"],1,inplace=True)
```

In [22]:
```
#Converting "Observation Date" into Datetime format
covid["ObservationDate"]=pd.to_datetime(covid["ObservationDate"])
```

In [23]:
```
#Grouping different types of cases as per the date
datewise=covid.groupby(["ObservationDate"]).agg({"Confirmed":'sum',"Reco
vered":'sum',"Deaths":'sum'})
datewise["Days Since"]=datewise.index-datewise.index.min()
```

In [24]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
!pip install plotly
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
import six
from datetime import timedelta
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
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,ExponentialSmoot
hing
from sklearn.preprocessing import PolynomialFeatures
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
std=StandardScaler()
```

```
Requirement already satisfied: plotly in /anaconda3/lib/python3.6/site-
packages (4.2.1)
Requirement already satisfied: six in /anaconda3/lib/python3.6/site-pac
kages (from plotly) (1.12.0)
Requirement already satisfied: retrying>=1.3.3 in /anaconda3/lib/python
3.6/site-packages (from plotly) (1.3.3)
WARNING: You are using pip version 20.0.2; however, version 20.1.1 is a
vailable.
You should consider upgrading via the '/anaconda3/bin/python -m pip ins
tall --upgrade pip' command.
```

Linear Regression for Confirmed cases Prediction

In [25]:
```
datewise["Days Since"]=datewise.index-datewise.index[0]
datewise["Days Since"]=datewise["Days Since"].dt.days
```

In [26]:
```
train_ml=datewise.iloc[:int(datewise.shape[0]*0.95)]
valid_ml=datewise.iloc[int(datewise.shape[0]*0.95):]
model_scores=[]
```

In [27]:
```
lin_reg=LinearRegression(normalize=True)
```

In [28]:
```
lin_reg.fit(np.array(train_ml["Days Since"]).reshape(-1,1),np.array(trai
n_ml["Confirmed"]).reshape(-1,1))
```

Out[28]:
```
LinearRegression(normalize=True)
```

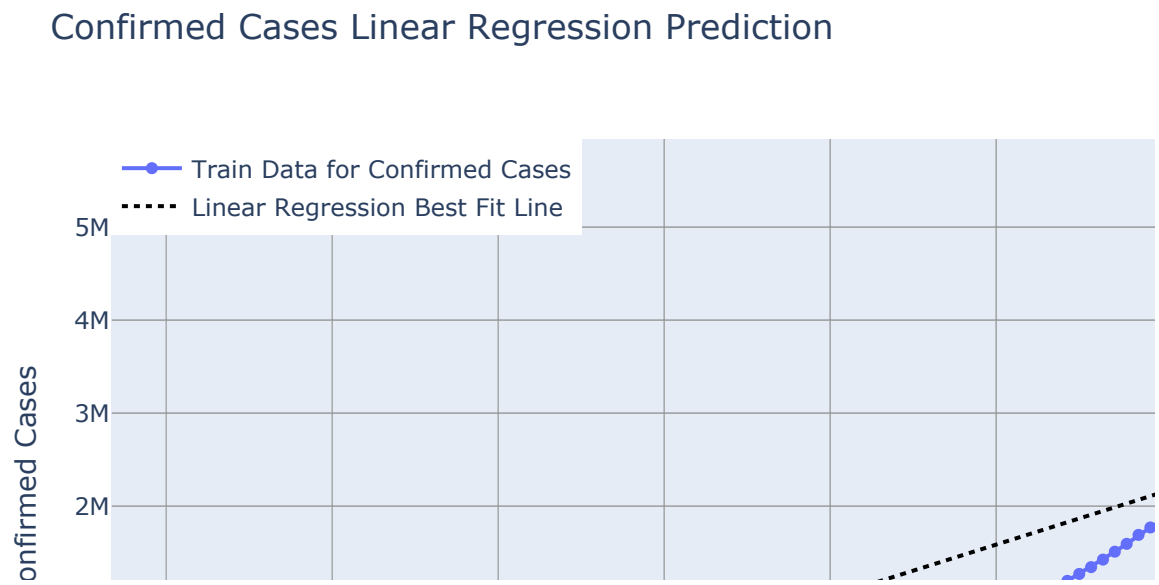
In [29]:
```
prediction_valid_linreg=lin_reg.predict(np.array(valid_ml["Days Since"])
.reshape(-1,1))
```

In [30]:
```
model_scores.append(np.sqrt(mean_squared_error(valid_ml["Confirmed"],pre
diction_valid_linreg)))
print("Root Mean Square Error for Linear Regression: ",np.sqrt(mean_squa
red_error(valid_ml["Confirmed"],prediction_valid_linreg)))
```

```
Root Mean Square Error for Linear Regression:  1455469.9995208008
```

In [31]:
```
plt.figure(figsize=(11,6))
prediction_linreg=lin_reg.predict(np.array(datewise["Days Since"]).resha
pe(-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()
```




```
<Figure size 792x432 with 0 Axes>
```

The Linear Regression Model is absolutely falling aprat. As it is clearly visible that the trend of Confirmed Cases in absolutely not Linear.

Polynomial Regression for Prediction of Confirmed Cases

In [32]:
```
train_ml=datewise.iloc[:int(datewise.shape[0]*0.95)]
valid_ml=datewise.iloc[int(datewise.shape[0]*0.95):]
```

In [33]:
```
poly = PolynomialFeatures(degree = 10)
```

In [34]:
```
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"]
```

In [35]:
```
linreg=LinearRegression(normalize=True)
linreg.fit(train_poly,y)
```

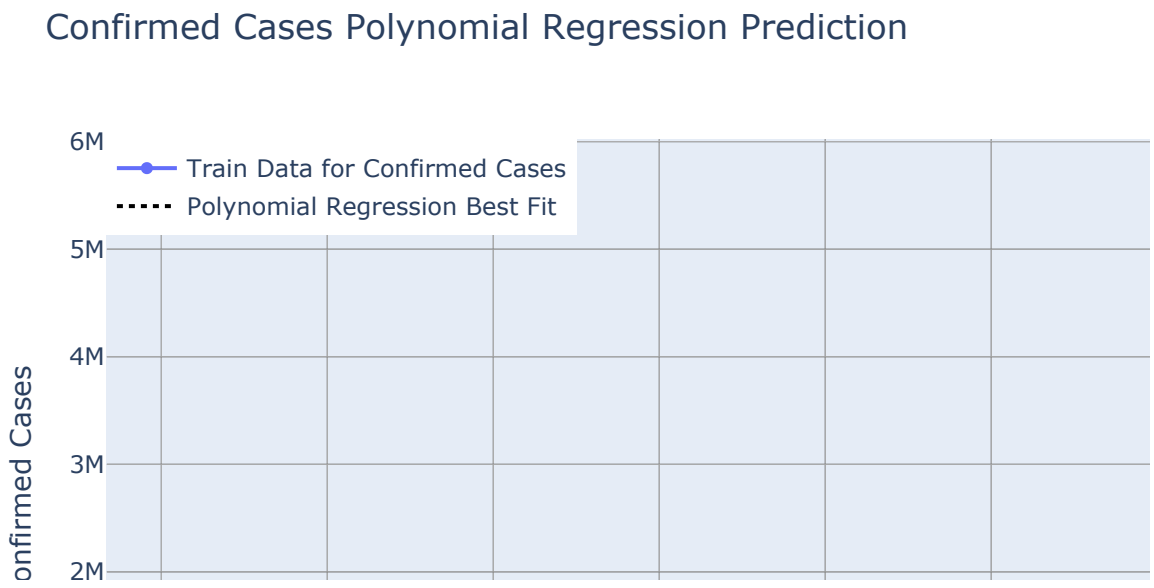
Out[35]:
```
LinearRegression(normalize=True)
```

In [36]:
```
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)
```

```
Root Mean Squared Error for Polynomial Regression:  92458.33270337264
```

In [37]:
```
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 Predictio
n",
                  xaxis_title="Date",yaxis_title="Confirmed Cases",
                  legend=dict(x=0,y=1,traceorder="normal"))
fig.show()
```




```
<Figure size 792x432 with 0 Axes>
```

In [38]:
```
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])
```

Support Vector Machine ModelRegressor for Prediction of Confirmed Case

In [39]:
```
train_ml=datewise.iloc[:int(datewise.shape[0]*0.95)]
valid_ml=datewise.iloc[int(datewise.shape[0]*0.95):]
```

In [40]:
```
#Intializing SVR Model
svm=SVR(C=1,degree=5,kernel='poly',epsilon=0.01)
```

In [41]:
```
#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))
```

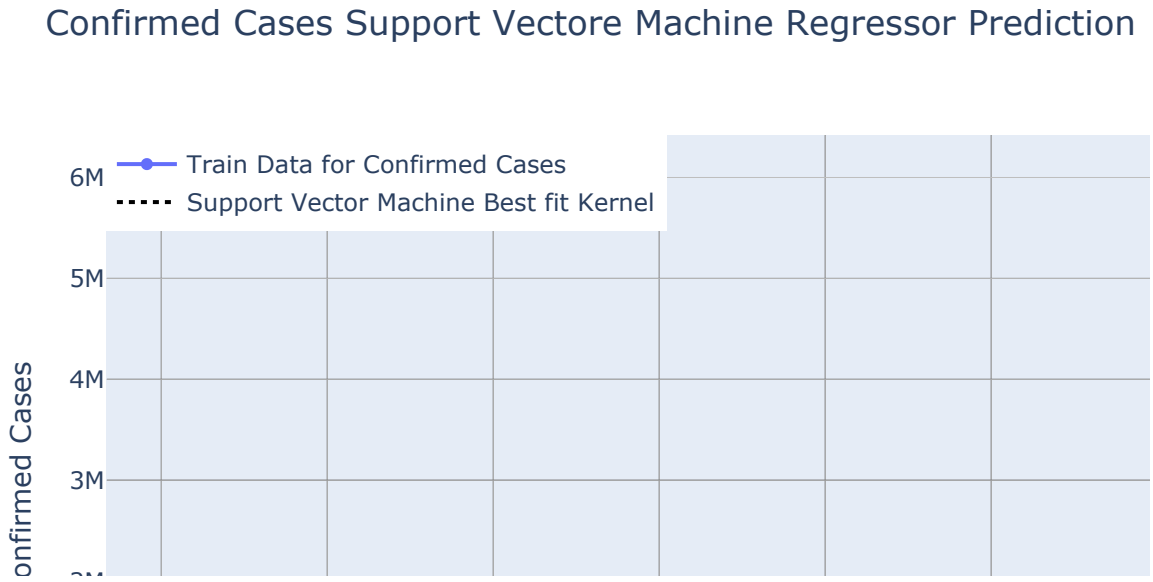
Out[41]:
```
SVR(C=1, degree=5, epsilon=0.01, kernel='poly')
```

In [42]:
```
prediction_valid_svm=svm.predict(np.array(valid_ml["Days Since"]).reshap
e(-1,1))
```

In [43]:
```
model_scores.append(np.sqrt(mean_squared_error(valid_ml["Confirmed"],pre
diction_valid_svm)))
print("Root Mean Square Error for Support Vectore Machine: ",np.sqrt(mea
n_squared_error(valid_ml["Confirmed"],prediction_valid_svm)))
```

```
Root Mean Square Error for Support Vectore Machine:  311298.7092689125
```

In [44]:
```
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 K
ernel",
                    line=dict(color='black', dash='dot')))
fig.update_layout(title="Confirmed Cases Support Vectore Machine Regress
or Prediction",
                  xaxis_title="Date",yaxis_title="Confirmed Cases",legend
=dict(x=0,y=1,traceorder="normal"))
fig.show()
```



<Figure size 792x432 with 0 Axes>

In [45]:
```
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])
```

In [46]:
```
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()
```

Out[46]:

	Dates	Linear Regression Prediction	Polynonmial Regression Prediction	SVM Prediction
0	2020-05-26	3912878.857095	5917606.035900	6321303.883234
1	2020-05-27	3952970.985187	6175507.673398	6572149.584052
2	2020-05-28	3993063.113279	6487548.973352	6831086.304902
3	2020-05-29	4033155.241371	6867024.057100	7098308.217984
4	2020-05-30	4073247.369463	7329703.679300	7374012.577501

Linear predictions are no where close to the actual values

Time Series Forecating

Holt's Linear Model

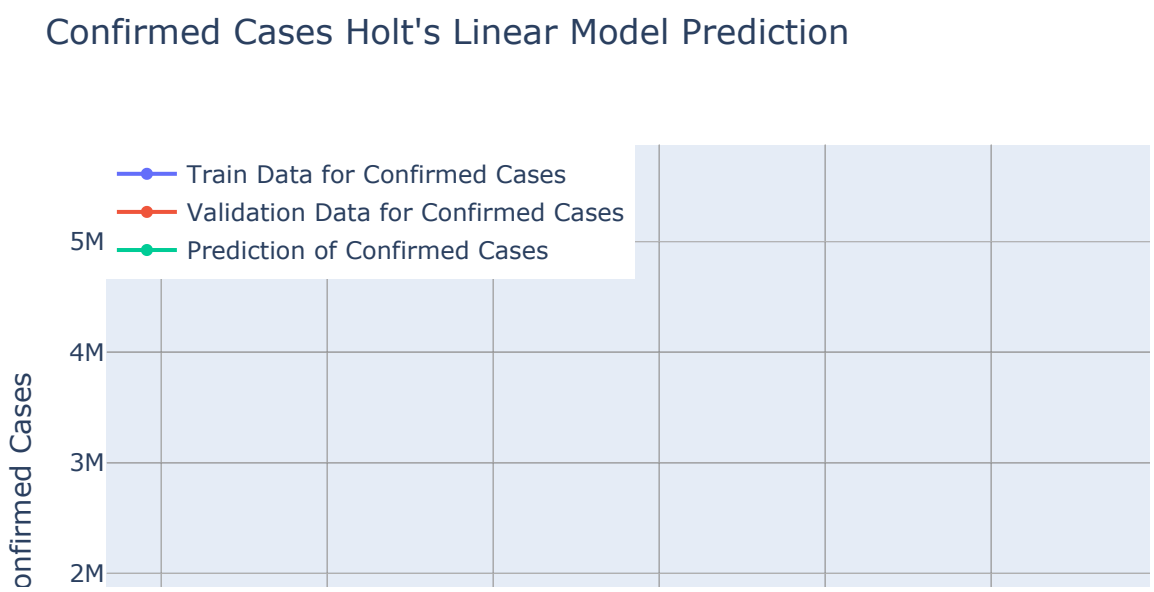
In [47]:
```
model_train=datewise.iloc[:int(datewise.shape[0]*0.95)]
valid=datewise.iloc[int(datewise.shape[0]*0.95):]
y_pred=valid.copy()
```

In [48]:
```
holt=Holt(np.asarray(model_train["Confirmed"])).fit(smoothing_level=0.3,smoothing_slope=1.0,optimized=False)
```

In [49]:
```
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"])))
```

```
Root Mean Square Error Holt's Linear Model:  24450.45935022441
```

In [50]:
```
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 Confi
rmed Cases",))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["Holt"],
                    mode='lines+markers',name="Prediction of Confirmed C
ases",))
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()
```



In [51]:
```
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()
```

Out[51]:

	Dates	Linear Regression Prediction	Polynonmial Regression Prediction	SVM Prediction	Holt's Linear Model Prediction
0	2020-05-26	3912878.857095	5917606.035900	6321303.883234	5558721.687501
1	2020-05-27	3952970.985187	6175507.673398	6572149.584052	5652410.760206
2	2020-05-28	3993063.113279	6487548.973352	6831086.304902	5746099.832911
3	2020-05-29	4033155.241371	6867024.057100	7098308.217984	5839788.905616
4	2020-05-30	4073247.369463	7329703.679300	7374012.577501	5933477.978321

Holt's winter model for daily time series

In [52]:
```
model_train=datewise.iloc[:int(datewise.shape[0]*0.95)]
valid=datewise.iloc[int(datewise.shape[0]*0.95):]
y_pred=valid.copy()
```

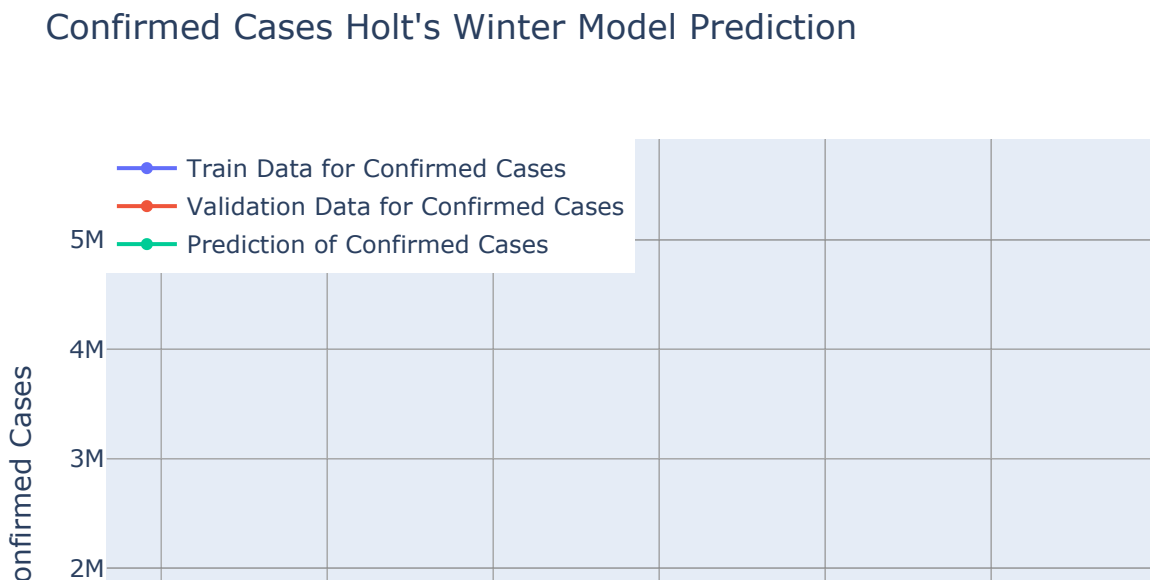
In [53]:
```
es=ExponentialSmoothing(np.asarray(model_train['Confirmed']),seasonal_periods=7,trend='mul', seasonal='mul').fit()
```

In [54]:
```
y_pred["Holt's Winter Model"]=es.forecast(len(valid))
```

In [55]:
```
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"])))
```

```
Root Mean Square Error for Holt's Winter Model:  19744.217511524937
```

In [56]:
```
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 Confi
rmed Cases",))
fig.add_trace(go.Scatter(x=valid.index, y=y_pred["Holt\'s Winter Model"
],
                    mode='lines+markers',name="Prediction of Confirmed C
ases",))
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()
```



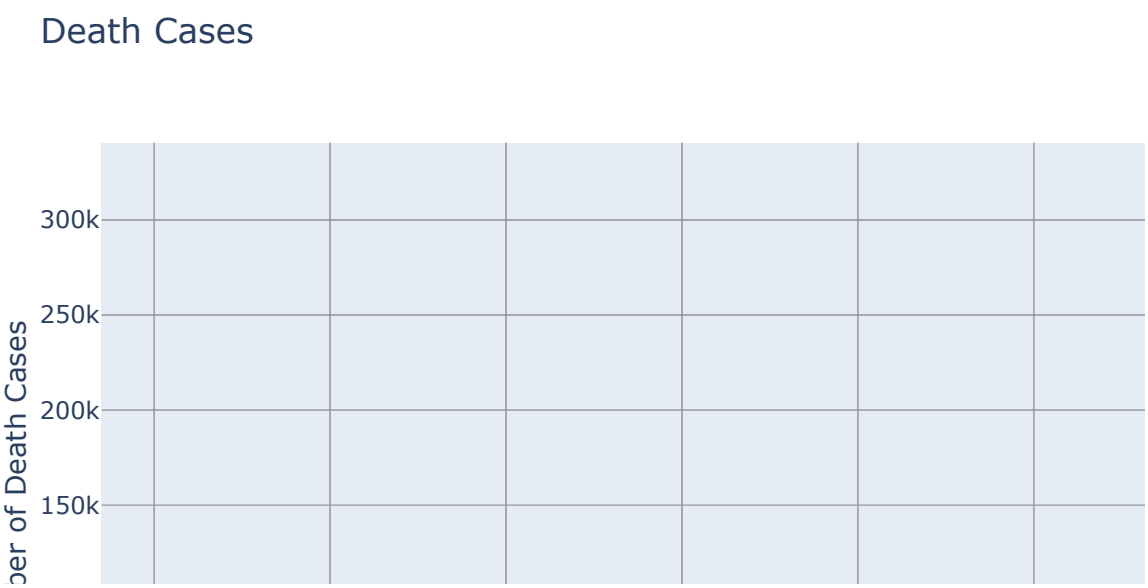
In [57]:
```
holt_winter_new_prediction=[]
for i in range(1,18):
    holt_winter_new_prediction.append(es.forecast((len(valid)+i))[-1])
model_predictions["Holt's Winter Model Prediction"]=holt_winter_new_pred
iction
model_predictions.head()
```

Out[57]:

	Dates	Linear Regression Prediction	Polynonmial Regression Prediction	SVM Prediction	Holt's Linear Model Prediction	Holt's Winter Model Prediction
0	2020-05-26	3912878.857095	5917606.035900	6321303.883234	5558721.687501	5646950.665657
1	2020-05-27	3952970.985187	6175507.673398	6572149.584052	5652410.760206	5761752.727572
2	2020-05-28	3993063.113279	6487548.973352	6831086.304902	5746099.832911	5881495.761054
3	2020-05-29	4033155.241371	6867024.057100	7098308.217984	5839788.905616	6010594.077581
4	2020-05-30	4073247.369463	7329703.679300	7374012.577501	5933477.978321	6131901.479046

In [58]:
```
model_train=datewise.iloc[:int(datewise.shape[0]*0.95)]
valid=datewise.iloc[int(datewise.shape[0]*0.95):]
y_pred=valid.copy()
```

In [59]:
```
fig=go.Figure()
fig.add_trace(go.Scatter(x=model_train.index, y=model_train["Deaths"],
                         mode='lines+markers',name="Death Cases"))
fig.update_layout(title="Death Cases",
                  xaxis_title="Date",yaxis_title="Number of Death Cases",
legend=dict(x=0,y=1,traceorder="normal"))
fig.show()
```



Prophet Model for Forecasting

In [60]:
```
from fbprophet import Prophet
```

In [61]:
```
prophet_c=Prophet(interval_width=0.95,weekly_seasonality=True,)
prophet_confirmed=pd.DataFrame(zip(list(datewise.index),list(datewise["C
onfirmed"])),columns=['ds','y'])
```

In [62]:
```
prophet_c.fit(prophet_confirmed)
```

```
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_se
asonality=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seas
onality=True to override this.
```

Out[62]:
```
<fbprophet.forecaster.Prophet at 0x11e126a20>
```

In [63]:
```
forecast_c=prophet_c.make_future_dataframe(periods=17)
forecast_confirmed=forecast_c.copy()
```

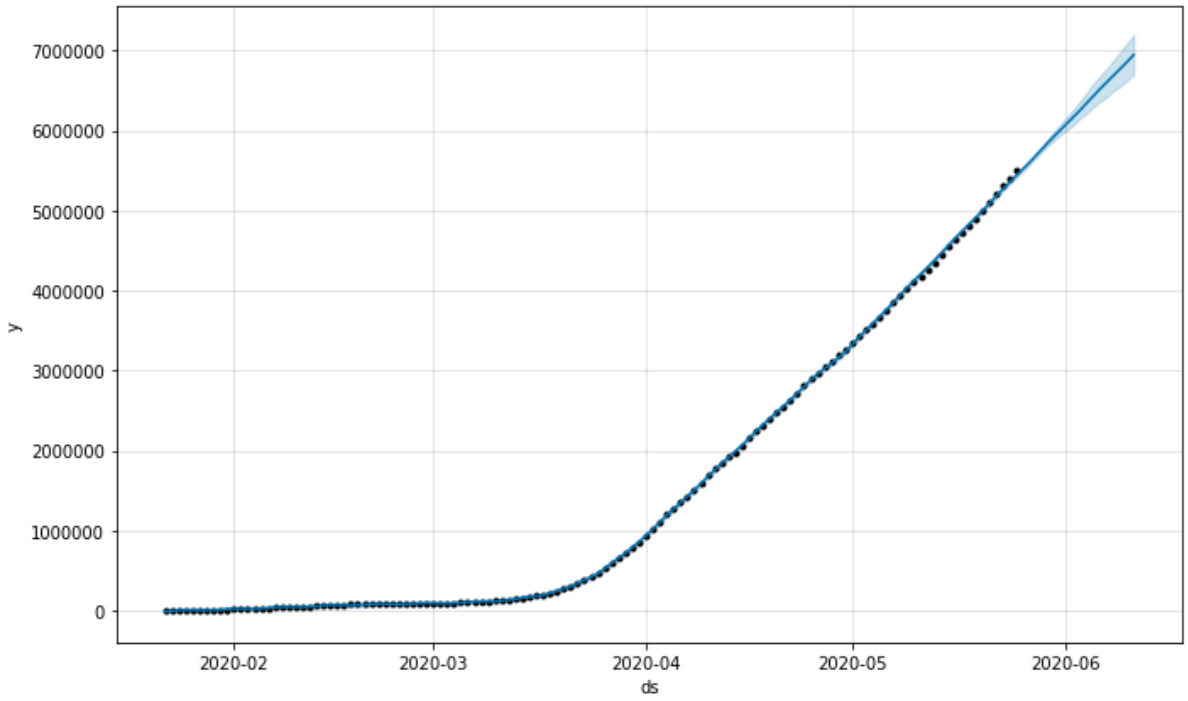
In [64]:
```
confirmed_forecast=prophet_c.predict(forecast_c)
```

In [65]:
```
model_scores.append(np.sqrt(mean_squared_error(datewise["Confirmed"],con
firmed_forecast['yhat'].head(datewise.shape[0]))))
print("Root Mean Squared Error for Prophet Model: ",np.sqrt(mean_squared
_error(datewise["Confirmed"],confirmed_forecast['yhat'].head(datewise.sh
ape[0]))))
```

```
Root Mean Squared Error for Prophet Model:  12535.658623928177
```

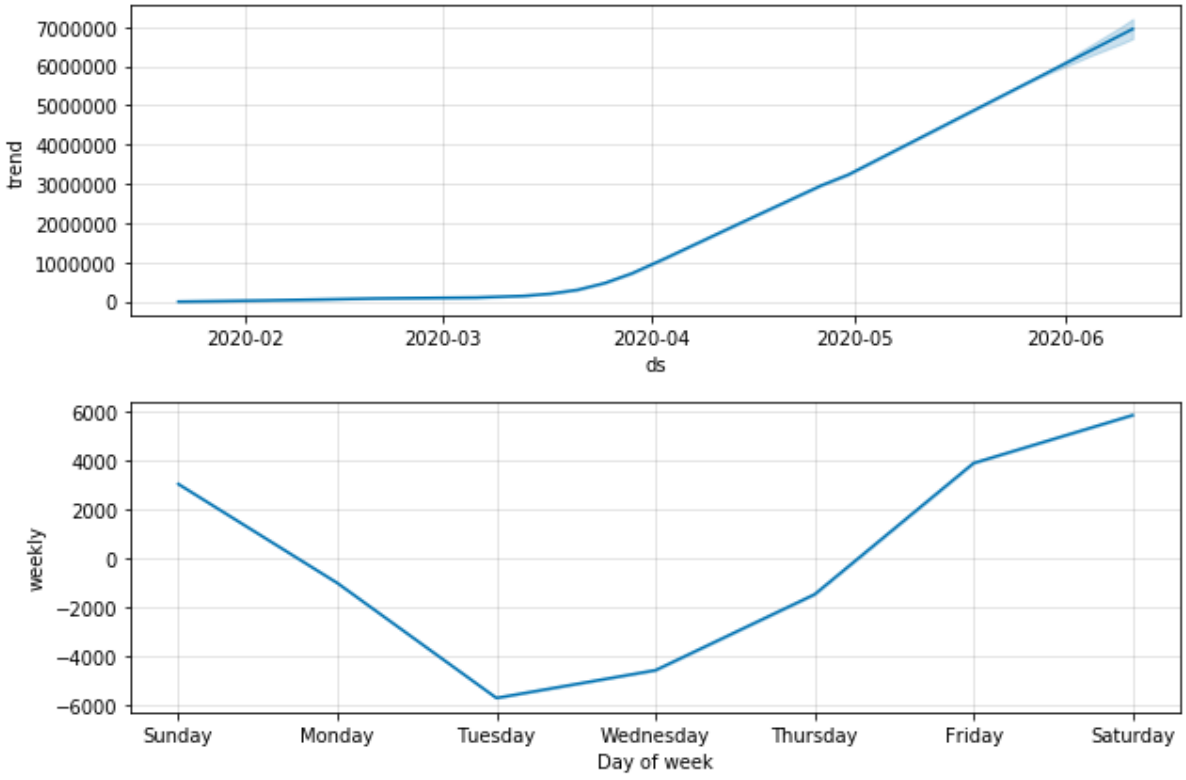
In [66]:
```
print(prophet_c.plot(confirmed_forecast))
```

```
Figure(720x432)
```



In [67]:
```
print(prophet_c.plot_components(confirmed_forecast))
```

```
Figure(648x432)
```



In [68]:
```
model_names=["Linear Regression","Polynomial Regression","Support Vector Machine Regressor","Holt's Linear","Holt's Winter Model", "Facebook's Prophet 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
```

Out[68]:

	Model Name	Root Mean Squared Error
5	Facebook's Prophet Model	12535.658624
4	Holt's Winter Model	19744.217512
3	Holt's Linear	24450.459350
1	Polynomial Regression	92458.332703
2	Support Vector Machine Regressor	311298.709269
0	Linear Regression	1455469.999521

COVID-19 doesn't have very high mortatlity rate as we can see which is the most positive take away. Also the healthy Recovery Rate implies the disease is cureable. The only matter of concern is the exponential growth rate of infection.