In [1]:

```
import pandas as pd
import numpy as np
from fbprophet import Prophet
import matplotlib.pyplot as plt
from sklearn import metrics
from statsmodels.tools.eval_measures import rmse
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [2]:

```
no_confirmed = pd.read_csv("time_series_covid19_confirmed_global.csv")
no_deaths =  pd.read_csv("time_series_covid19_deaths_global.csv")
no_recovered =  pd.read_csv("time_series_covid19_recovered_global.csv")
no_confirmed.rename(columns={'Country/Region':'Country'}, inplace=True)
no_recovered.rename(columns={'Country/Region':'Country'}, inplace=True)
no_deaths.rename(columns={'Country/Region':'Country'}, inplace=True)
no_confirmed = no_confirmed.melt(id_vars=["Province/State","Country","Lat","Long"],var_
no_deaths = no_deaths.melt(id_vars=["Province/State","Country","Lat","Long"],var_name =
no_recovered = no_recovered.melt(id_vars=["Province/State","Country","Lat","Long"],var_
no_confirmed["Deaths"] = no_deaths.Deaths
no_confirmed["Recovered"] = no_recovered.Recovered
```

In [3]:

```
X = no_confirmed
confirmed = X.groupby('Date').sum()['Confirmed'].reset_index()
deaths = X.groupby('Date').sum()['Deaths'].reset_index()
recovered = X.groupby('Date').sum()['Recovered'].reset_index()
confirmed.columns = ['ds','y']
confirmed['ds'] = pd.to_datetime(confirmed['ds'])
```

In [4]:

```
m = Prophet(interval_width=0.95,yearly_seasonality=True,daily_seasonality=True)
m.fit(confirmed)
```

```
INFO:numexpr.utils:NumExpr defaulting to 8 threads.
```

Out[4]:

```
<fbprophet.forecaster.Prophet at 0x2061059d280>
```

In [5]:

```
future = m.make_future_dataframe(periods=7)
forecast = m.predict(future)
forecast
```

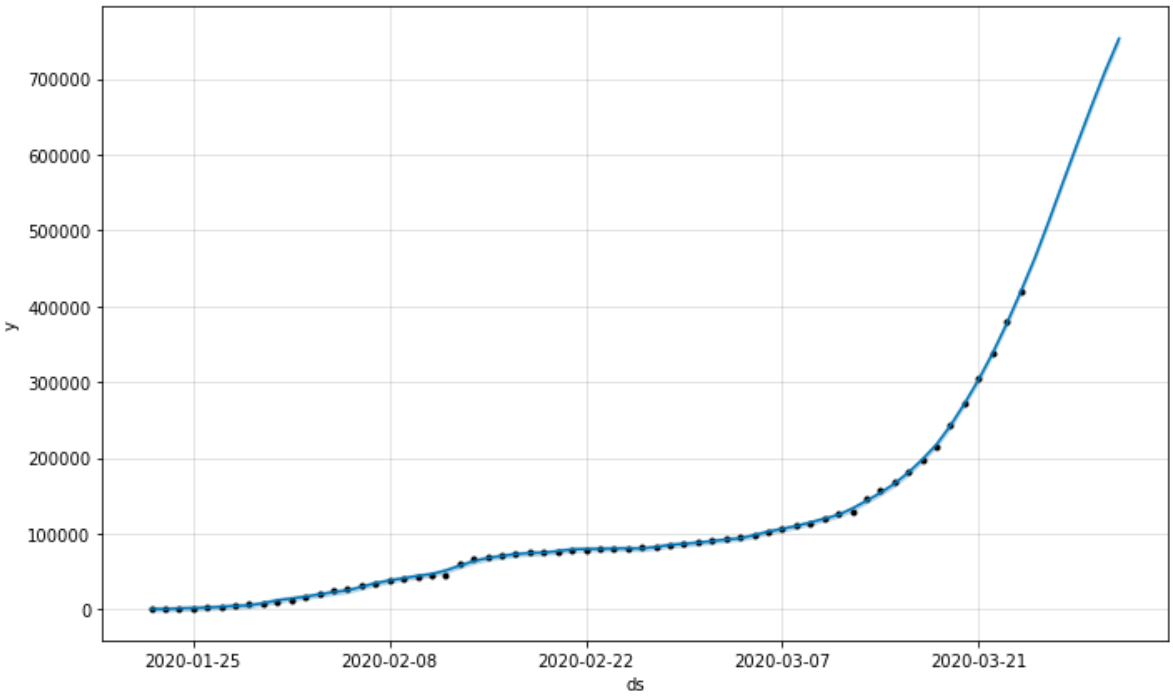
Out[5]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_te
0	2020-01-22	35704.627812	-2557.575599	3617.237167	35704.627812	35704.627812	-35110.284
1	2020-01-23	33169.312159	-2812.961180	3233.718244	33169.312159	33169.312159	-32889.769
2	2020-01-24	30633.996507	-1680.020415	4187.147898	30633.996507	30633.996507	-29342.309
3	2020-01-25	28098.680853	-1305.761399	4645.687253	28098.680853	28098.680853	-26438.940
4	2020-01-26	25563.365200	-833.999946	5335.555881	25563.365200	25563.365200	-23385.360
...	...	...	...	...	...	...	
65	2020-03-27	7712.357215	560038.876053	566233.756196	6822.186643	8641.675354	555400.846
66	2020-03-28	8169.743986	609806.696081	616539.275992	6729.944982	9594.104207	604932.134
67	2020-03-29	8627.130757	658371.632418	665712.065606	6684.340543	10613.756733	653505.748
68	2020-03-30	9084.517528	705504.300909	713530.483086	6547.859622	11680.596125	700338.355
69	2020-03-31	9541.904299	748521.817242	757119.848824	6363.724726	12723.741275	743301.076

70 rows × 22 columns

In [6]:

```
1 confirmed_forecast_plot = m.plot(forecast)
```



In [7]:

```
deaths.columns = ['ds','y']
deaths['ds'] = pd.to_datetime(deaths['ds'])
m_deaths = Prophet(interval_width=0.95,yearly_seasonality=True,daily_seasonality=True)
m_deaths.fit(deaths)
future_deaths = m_deaths.make_future_dataframe(periods=365)
forecast_deaths = m_deaths.predict(future_deaths)
forecast_deaths
```

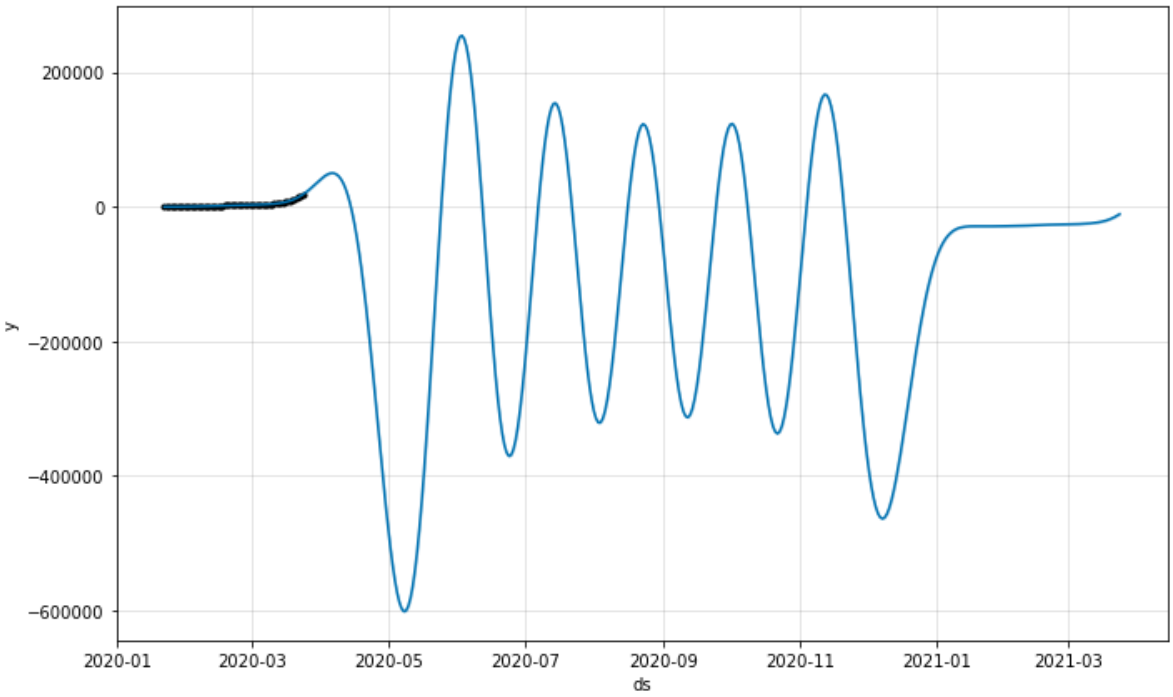
Out[7]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_
0	2020-01-22	-5203.215943	-57.491151	126.781090	-5203.215943	-5203.215943	5233.2
1	2020-01-23	-5282.115649	-94.958721	94.047149	-5282.115649	-5282.115649	5281.4
2	2020-01-24	-5361.015355	-68.492486	117.350966	-5361.015355	-5361.015355	5386.7
3	2020-01-25	-5439.915060	-40.151701	137.073860	-5439.915060	-5439.915060	5487.7
4	2020-01-26	-5518.814766	-39.043611	152.010447	-5518.814766	-5518.814766	5580.6
...	...	...	...	...	...	...	
423	2021-03-20	-38567.525060	-17932.086472	-17680.585565	-38651.311744	-38486.427409	20758.2
424	2021-03-21	-38646.391918	-16476.510056	-16217.682573	-38730.607732	-38564.884255	22298.0
425	2021-03-22	-38725.258777	-14800.413151	-14563.288588	-38809.903537	-38643.389156	24039.9
426	2021-03-23	-38804.125636	-12929.842663	-12684.200386	-38889.199239	-38721.896894	25998.6
427	2021-03-24	-38882.992494	-10850.313715	-10601.782676	-38968.494908	-38800.411507	28157.3

428 rows × 22 columns

In [8]:

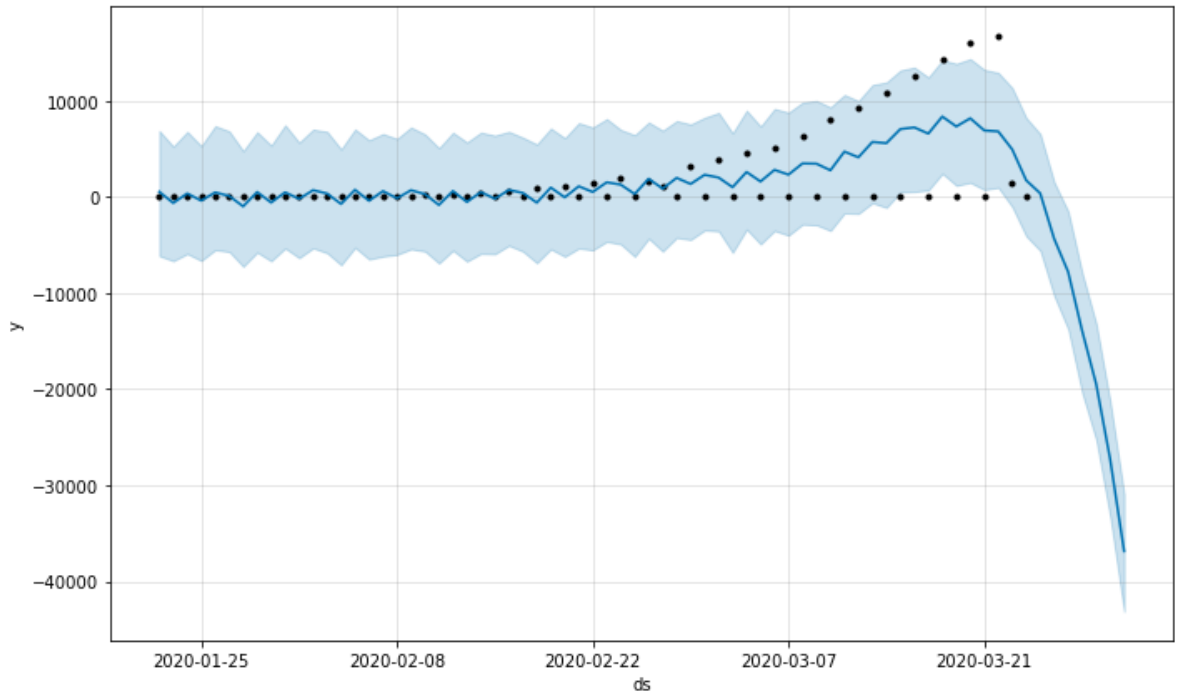
```
1 confirmed_forecast_plot_deaths = m_deaths.plot(forecast_deaths)
```

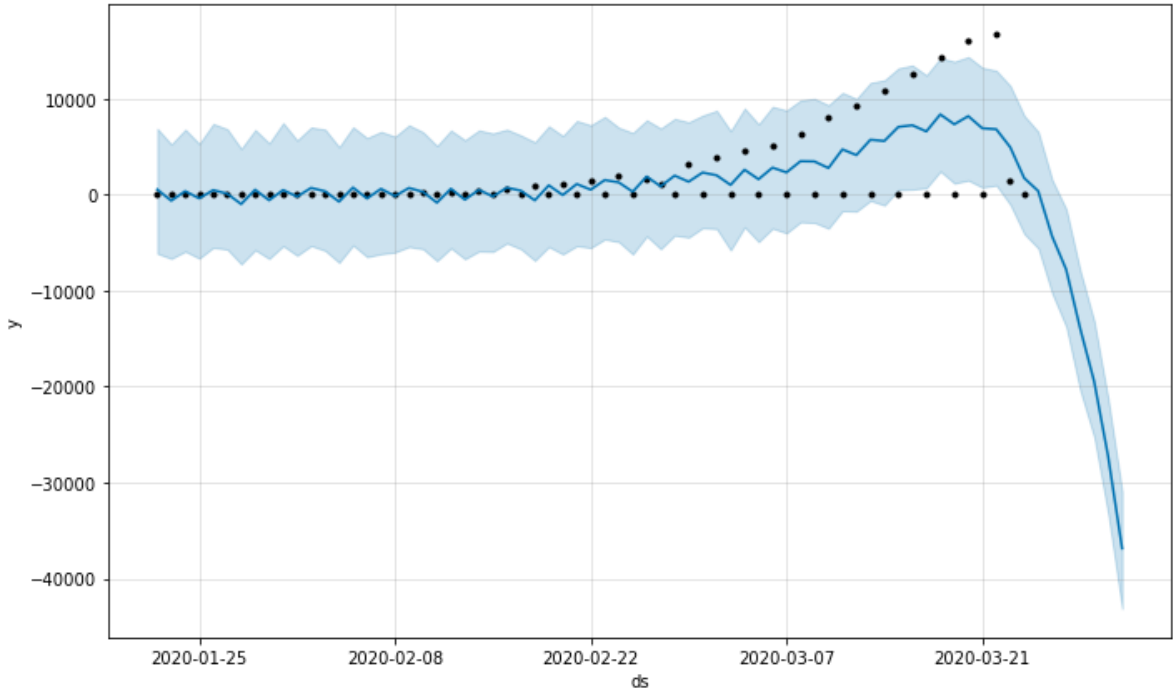


In [14]:

```
recovered.columns = ['ds','y']
recovered['ds'] = pd.to_datetime(recovered['ds'])
m_recovered = Prophet(interval_width=0.95,yearly_seasonality=True,daily_seasonality=Tru
m_recovered.fit(recovered)
future_recovered = m_recovered.make_future_dataframe(periods=7)
forecast_recovered = m_recovered.predict(future_recovered)
m_recovered.plot(forecast_recovered)
```

Out[14]:




10000
0
−10000
−20000
−30000
−40000
y
2020-01-25
2020-02-08
2020-02-22
2020-03-07
2020-03-21
ds

In [15]:

```
prophet_pred = pd.DataFrame({"Date" : forecast[:63]['ds'], "Pred" : forecast[:63]["yhat
prophet_pred = prophet_pred.set_index("Date")
prophet_pred
```

Out[15]:

Date	Pred
2020-01-22	594.343443
2020-01-23	279.542915
2020-01-24	1291.686609
2020-01-25	1659.740548
2020-01-26	2178.004669
...	...
2020-03-20	271885.805375
2020-03-21	303381.427348
2020-03-22	338318.809312
2020-03-23	377210.891781
2020-03-24	419244.875825

63 rows × 1 columns

In [16]:

```
confirmed.values[:,1]
```

Out[16]:

```
array([555, 654, 941, 1434, 2118, 2927, 5578, 6166, 8234, 9927, 12038,
       42762, 44802, 45221, 60368, 66885, 69030, 71224, 73258, 75136,
       75639, 16787, 76197, 76819, 78572, 78958, 79561, 80406, 81388,
       82746, 84112, 86011, 19881, 23892, 27635, 30794, 34391, 37120,
       40150, 88369, 118602, 125875, 128353, 145209, 156104, 167454,
       181573, 197150, 214909, 242706, 90306, 272164, 304519, 337089,
       378547, 418678, 92840, 95120, 97882, 101794, 105831, 109805,
       113571], dtype=object)
```

In [17]:

```
1 prophet_pred.Pred
```

Out[17]:

```
Date
2020-01-22       594.343443
2020-01-23       279.542915
2020-01-24      1291.686609
2020-01-25      1659.740548
2020-01-26      2178.004669
                   ...
2020-03-20    271885.805375
2020-03-21    303381.427348
2020-03-22    338318.809312
2020-03-23    377210.891781
2020-03-24    419244.875825
Name: Pred, Length: 63, dtype: float64
```

In [18]:

```
1 prophet_rmse_error = rmse(confirmed.values[:,1],prophet_pred.Pred)
2 print(f'RMSE Error: {prophet_rmse_error}')
```

```
RMSE Error: 94555.53044523322
```

In [19]:

```
1 MAE=metrics.mean_absolute_error(confirmed.values[:,1],prophet_pred.Pred )
2 print(f'Mean Absolute Error:{MAE}')
```

```
Mean Absolute Error:60189.39839964654
```

In [20]:

```
1 EPSILON = 1e-10
2 def _error(actual: np.ndarray, predicted: np.ndarray):
3     return actual - predicted
4
5 def _percentage_error(actual: np.ndarray, predicted: np.ndarray):
6     return _error(actual, predicted) / (actual + EPSILON)
7
```

In [21]:

```
1 def rrse(actual: np.ndarray, predicted: np.ndarray):
2     return np.sqrt(np.sum(np.square(actual - predicted)) / np.sum(np.square(actual - np
3 RRSE=rrse(confirmed.values[:,1],prophet_pred.Pred)
4 print(f'Root Relative Squared Error:{RRSE}')
```

```
Root Relative Squared Error:1.0327752934003431
```

In [22]:

```
def mape(actual: np.ndarray, predicted: np.ndarray):
    return np.mean(np.abs(_percentage_error(actual, predicted)))
MAPE=mape(confirmed.values[:,1],prophet_pred.Pred)
print(f'Mean Absolute Percentage Error:{MAPE}')
```

```
Mean Absolute Percentage Error:0.6910617774729575
```