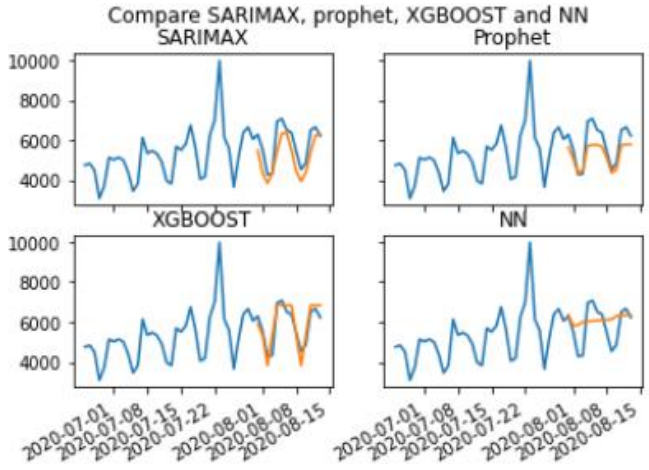
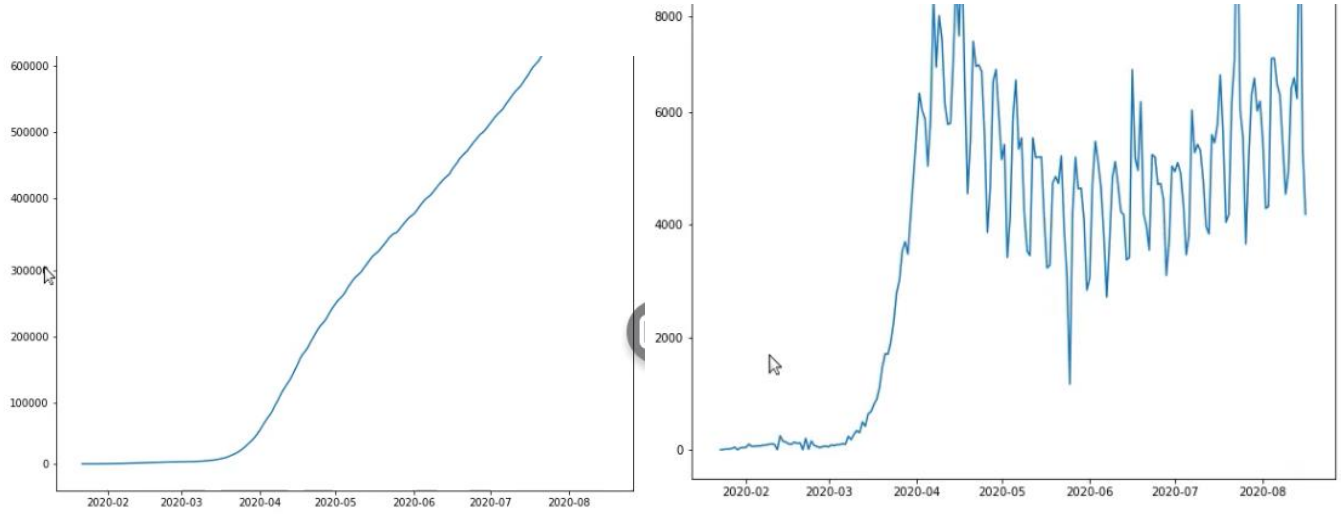
Comparing Time Series Predictions of Covid-19 Deaths

* Data file was downloaded from the official GitHub page of Johns Hopkins University



Part 1: Data Pre-Possessing

Plot the daily number of deaths and use diff. function to see how daily increase looks like:



Part 2: Forecasting Using SARIMAX Model

(Seasonal Auto Regressive Integrated Moving Average with Exogenous Regressors Model)

SARIMAX is an improvement upon traditional ARMA model since it incorporates seasonality. AR(p) captures mean reversion effects, MA(q) captures the shock effects observed in the white noise terms.

Before we apply the model, we will split our dataset into training and test sets.

We take all data prior to July 31st as training set, on which the model will be optimized, and all data from July 31st as test set, on which we will test the accuracy of the predictions that the optimized model will make.

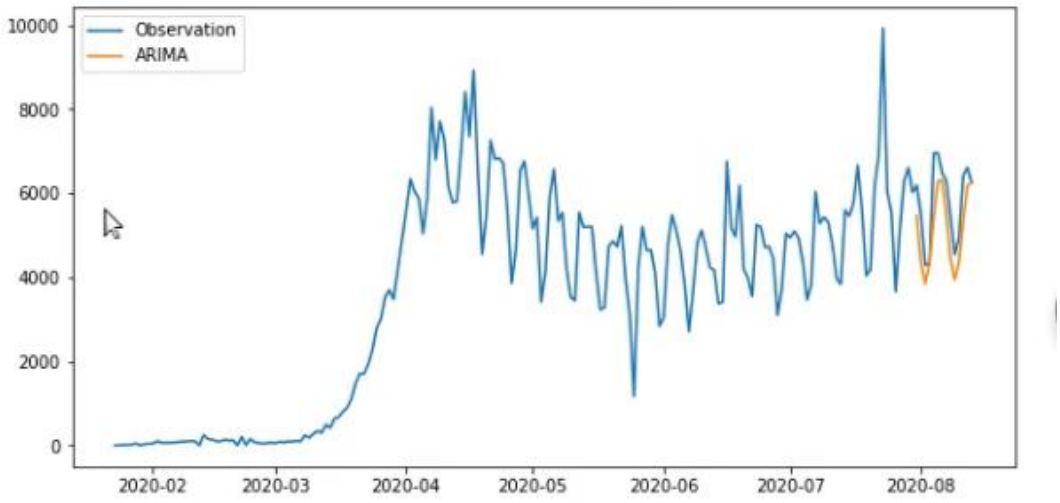
```
start_date = '2020-07-31'

train = dataset.loc[dataset.index < pd.to_datetime(start_date)]
test = dataset.loc[dataset.index >= pd.to_datetime(start_date)]
```

Now let's talk about SARIMAX. There are three hyperparameters that go into the order tuple: p, q and d
Here we've used p=2, q=1 and d=3.

```
model = SARIMAX(train, order=(2, 1, 3))
```

Next, we call the fit method to optimize the model:

```
sarimax_prediction = results.predict(
    start=start_date, end='2020-08-13', dynamic=False)
plt.figure(figsize=(10, 5))
l1, = plt.plot(dataset, label='Observation')
l2, = plt.plot(sarimax_prediction, label='ARIMA')
plt.legend(handles=[l1, l2])
plt.savefig('SARIMAX prediction', bbox_inches='tight', transparent=False)
```



Part 3: Forecasting Using Facebook's Prophet Model

this is an open-source time series library which does not require you specify parameters. It acts like a black box that does all computations on its own. Unlike SARIMAX, it expects data frame to have two columns.

```
train['ds'] = train.index.values
```

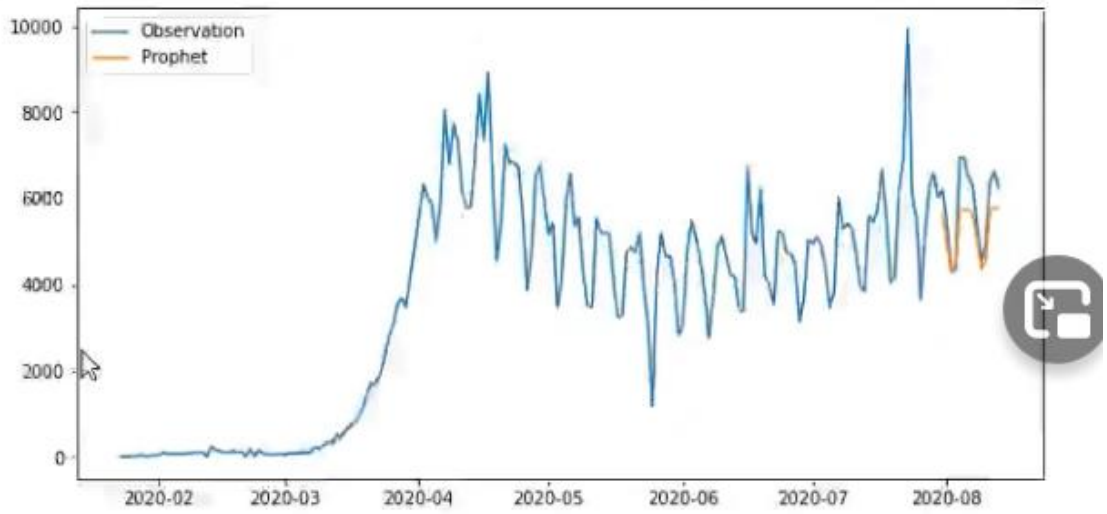
Then we create a new Prophet object and call the `fit()` method

```
m = Prophet()
m.fit(train)
```

Now let's forecast:

```
future = m.make_future_dataframe(periods=dataset.shape[0]-train.shape[0])
prophet_prediction = m.predict(future)
```

```
plt.figure(figsize=(10, 5))
l1, = plt.plot(dataset, label='Observation')
l2, = plt.plot(prophet_future, label='Prophet')
plt.legend(handles=[l1, l2])
plt.savefig('Prophet predictions',
            bbox_inches='tight', transparent=False)
```



Part 5: Predict with XGBOOST and NN Models

Unlike SAIMAX and Prophet, these two models are supervised machine learning models, which assumes each data point is independent from the rest.

XGBoost is decision-tree-based algorithm that widely used in small-to-medium structured/tabular data.

Artificial neural networks is mathematical based and is widely in unstructured data.

Part 6: Compare the Mean Absolute Error of All Models

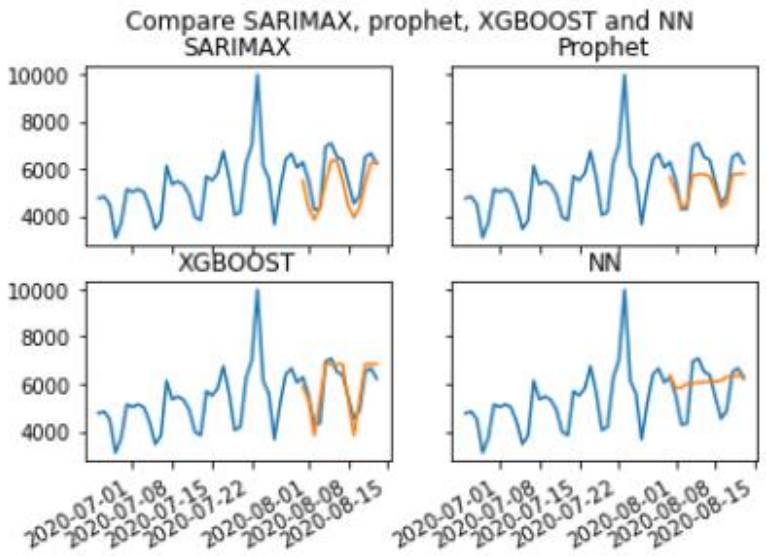
We can see that the winner of this contest is the XGBOOST, which has the lowest mean absolute of all of them. There was a fitting compared to the other models.

Let's compare the MAE values.

```
print('XGBOOST MAE = ', mean_absolute_error(XGBOOST_prediction, y_test))
print('Prophet MAE = ', mean_absolute_error(prophet_future, test))
print('SARIMAX MAE = ', mean_absolute_error(sarimax_prediction, test))
print('NN MAE = ', mean_absolute_error(NN_prediction, test))
```

```
XGBOOST MAE =  392.1467982700893
Prophet MAE =  572.5555342749124
SARIMAX MAE =  650.6157483614192
NN MAE =  727.6703752790179
```

Part 7: Who is the WINNER?



XGBoost indeed simulate the best. MAE data is validated by the charts.

The neural network model, even though it seems to get the trend, IT does not really capture the seasonality in the data.

This project shows how important it is to use various approaches when performing time series analysis.

Important note: This time series analysis is a simplified analysis which is fully data driven. To obtain more accurate predictions, scientists and epidemiologists incorporate other variables into the models as well as utilize epidemiological simulations or infection simulations to simulate the progress of the spread of infections, geographical, demographic, and other factors.