Task 1 Version 3

November 24, 2021

1 F06 G05 2D project 2021

In this project, we are tasked with supporting WHO in developing a model to predict the number of deaths in each country. There are several factors that contribute to the deaths caused by COVID-19, but not all data can be accurately transcribed into values that can be used to predict deaths in a nation or region. However, information such as the number of infected patients, a country's wealth, facilities, the number of vaccinated, the number of completely vaccinated, and the human development index can easily acquire and be incorporated in our model. Considering data from the actual world is not perfectly linear, we will look for data that are strongly correlated. We will be able to predict the number of deaths in a specific nation with assumptions made in any given day using this model.

In this project, we will be using 5 external libraries to aid us in building our model. The libraries are mainly pandas, numpy, seaborn, matplotlib and math. This project is tested and working on python v3.8.3. The cell below will install the requirements for this models.

[ ]:
```
pip install -r requirements.txt
```

[ ]:
```
from IPython.display import display
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
```

This model will make use of modified functions from the homework. Instead of using the mean and standard deviation in normalization, we will use the its maximum and minimum. The data are normalized to avoid variables from overflowing. Normalizing using standard deviation performs for normally distributed data, but because the data in this circumstance is not normally distributed and tends to follow a trend, min-max scaling will be adopted. The feasibility of the model will be determined using functions such as mean squared error, r2 score, adjusted r2 and correlation. While the rest will be employed to predict the model's coefficient.

[ ]:
```
def normalize_z(df):
    return (df-df.min(axis=0))/(df.max(axis=0)-df.min(axis=0))

def normalize_predict(df, dfx):
    return (dfx-df.min(axis=0))/(df.max(axis=0)-df.min(axis=0))
```

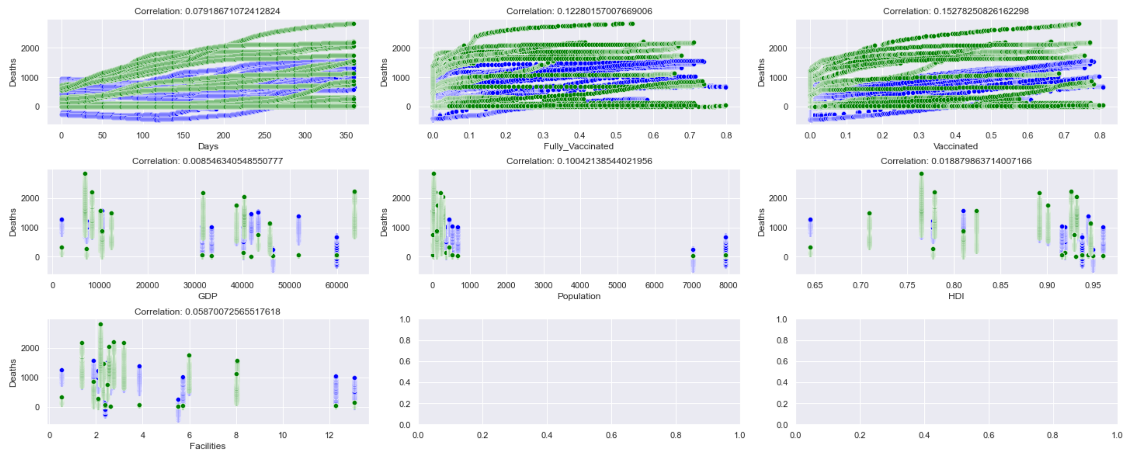
```python
def correlation(x, y):
    return x.corr(y)

def prepare_x(df_x):
    xAxis = df_x.to_numpy()
    array1 = np.ones((df_x.shape[0], 1))
    return np.concatenate((array1, xAxis), axis=1)

def prepare_y(df_y):
    return df_y.to_numpy()

def predict(df_x, beta):
    x = prepare_x(df_x)
    return np.matmul(x, beta)

def r2_score(y, ypred):
    ybar = np.mean(y)
    SStot = np.sum((y-ybar)**2)
    SSres = np.sum((y-ypred)**2)
    return 1-SSres/SStot

def mean_squared_error(target, pred):
    n = target.shape[0]
    s = np.sum((target-pred)**2)
    return (1/n)*s

def gradient_descent(X, y, beta, alpha, num_iters):
    m = X.shape[0]
    for n in range(num_iters):
        delta = np.matmul(X.T, np.matmul(X, beta) - y)
        beta = beta - (alpha/m)*delta
    return beta

def compute_cost(X, y, beta):
    error = np.matmul(X, beta) - y
    return (1/(2*X.shape[0]))*np.matmul(error.T, error)
```

Since WHO is interested in the prediction of COVID-19 death rate, we will create a statistical representation of the death rate at the global level. Deaths, populations, and policies of individual nations varies by thousands, making it difficult to develop a prediction model. To compensate for the variance, we will normalize all data sets, such as death in thousands, percentage of vaccination in the country, and infected per million. While working on modeling uncertainty in an excel sheet, data is cleaned up. To avoid future indications of unknown null data, we will set the value of all null cells to 0.

```
[ ]: # Read csv file
     dataFrame = pd.read_csv("2Ddata8.csv")
     # Y axis Data
     yVal = pd.DataFrame(dataFrame.loc[:, "total_deaths_per_million"].fillna(0))
     # X axis Data
     xVal = pd.DataFrame(columns=[])
     xVal["Days"] = pd.DataFrame(dataFrame["days"].fillna(0))
     xVal["Fully_Vaccinated"] = pd.
      ↪DataFrame(dataFrame["fully_vaccination_percentage"].fillna(0))
     xVal["Vaccinated"] = pd.DataFrame(dataFrame["vaccination_percentage"].fillna(0))
     xVal["GDP"] = pd.DataFrame(dataFrame["gdp_per_capita"].fillna(0))
     xVal["Population"] = pd.DataFrame(dataFrame["population_density"].fillna(0))
     xVal["HDI"] = pd.DataFrame(dataFrame["human_development_index"].fillna(0))
     xVal["Facilities"] = pd.DataFrame(dataFrame["hospital_beds_per_thousand"].
      ↪fillna(0))

     # Normalise all x-axis columns, convert x-axis and y-axis into arrays
     xAxis = prepare_x(normalize_z(xVal))
     yAxis = prepare_y(yVal)

     iterations, alpha, beta = 10500, 0.05, np.zeros((xAxis.shape[1], 1))
     beta = gradient_descent(xAxis, yAxis, beta, alpha, iterations)
```

We will compare our prediction to the actual data to determine the global trend in deaths where green represents the actual data and blue to represent the predicted.

```
[ ]: prediction = pd.DataFrame(predict(normalize_z(xVal), beta))
     fig, axs = plt.subplots(math.ceil(len(xVal.columns)/3), 3, figsize=(20, 8),␣
      ↪constrained_layout=True)
     for index, x in enumerate(xVal.columns):
         sns.set()
         corr = correlation(xVal[x], yVal["total_deaths_per_million"])
         scat0 = sns.scatterplot(x=xVal[x], y=prediction[0], ax=axs[int(index/3),␣
      ↪index%3,], color="blue")
         scat1 = sns.scatterplot(x=xVal[x], y=yVal["total_deaths_per_million"],␣
      ↪ax=axs[int(index/3), index%3,], color="green")
         scat0.set_title("Correlation: " + str(corr**2))
         scat0.set_ylabel("Deaths")
         scat0.set_xlabel(x)
     plt.show()
     print("Beta Coefficient: ", [", ".join(x) for x in np.round(beta, 2).
      ↪astype(str)])
     val = r2_score(prepare_y(yVal), prediction).values
     print("r^2 Value:", np.round(val, 2), ", Adjusted r^2 Value:", np.
      ↪round(1-((1-val)*(xVal.shape[0]-1)/(xVal.shape[0]-1-xVal.shape[1])), 2))
```

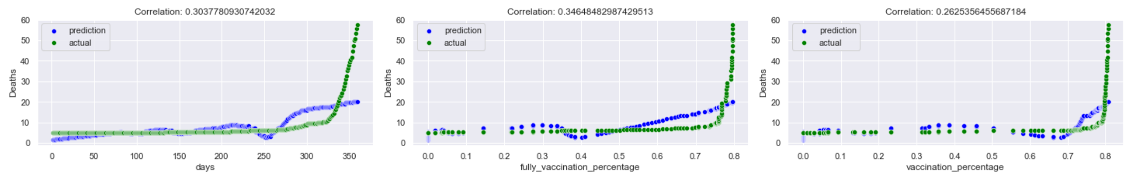


```
Beta Coefficient:  ['1009.41', '-533.86', '131.12', '1257.6', '178.18',
'-943.52', '-355.31', '-550.67']
r^2 Value: [0.34] , Adjusted r^2 Value: [0.34]
```

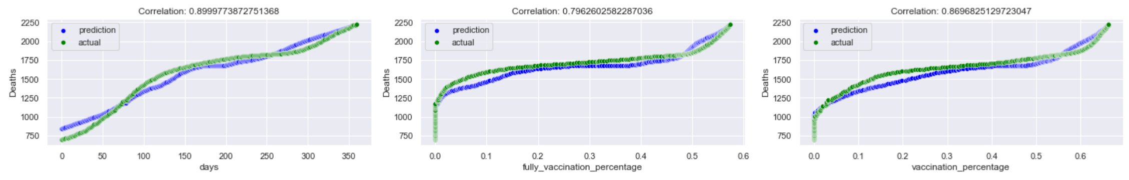
The graphs above show a clear relationship between deaths and each individual component. The adjusted r2 score suggests a poor prediction of deaths, which is understandable considering that various countries have differing policies, which may result in skewed statistics. Adjusted r2 score is used to prevent overly optimistic result with the use of unbias data. Countries such as the United States have a high death rate, making it an outlier in this regard. As a result, the actual death rate in the United States significantly differs from the projection. The fully vaccinated and vaccinated chart is as anticipated, as countries with higher vaccination rates should experience lower death rates. In terms of population densities and healthcare facilities, we see that regions with greater population densities have lower death rates, which can be explained by the stringent restrictions that are followed because smaller regions are easier to enforce rules in. Moreover, higher population densities result in a lower health-care facility coefficient. Countries with higher GDP and HDI had higher death rates, which could be explained by the practice of freedom and human rights in those regions where rules are not strictly enforced. In these two graphs, Singapore and Hong Kong are underrepresented due to their low death rates. The model accounts for nations with high and low death rates, which compensate for discrepancies in data. Despite the low adjusted r2 values, the model provides a good average forecast in the global death prediction.

[ ]:
```
cData, cYData, betaS = {}, {}, {}
for x in dataFrame["location"].drop_duplicates():
    cData[x] = pd.DataFrame.reset_index(dataFrame.loc[(dataFrame['location'] ==␣
 ↪x), :])
    cYData[x] = pd.DataFrame(cData[x]["total_deaths_per_million"])
    cData[x] = pd.DataFrame(cData[x].loc[:,["days",␣
 ↪"fully_vaccination_percentage", "vaccination_percentage"]])



for i in cData:
    xAxis = prepare_x(normalize_z(cData[i]))
    yAxis = prepare_y(cYData[i])
    iterations, alpha, beta1 = 10500, 0.05, np.zeros((xAxis.shape[1], 1))
    beta1 = gradient_descent(xAxis, yAxis, beta1, alpha, iterations)
    betaS[i] = beta1
    fig1, axs1 = plt.subplots(1, 3, figsize=(20, 3), constrained_layout=True)
    prediction1 = pd.DataFrame(predict(normalize_z(cData[i]), beta1))
    val = r2_score(prepare_y(cYData[i]), prediction1).values
    print(i, "- Beta Coefficient: ", [", ".join(x) for x in np.round(beta1, 2).
↪astype(str)])
    print("r^2 Value:", np.round(val, 2), ", Adjusted r^2 Value:", np.
↪round(1-((1-val)*(cData[i].shape[0]-1)/(cData[i].shape[0]-1-cData[i].
↪shape[1])), 2))
    for index1, x in enumerate(cData[i].columns):
        sns.set()
        corr = correlation(cData[i][x], cYData[i]["total_deaths_per_million"])
        scat0 = sns.scatterplot(x=cData[i][x], y=prediction1[0],␣
↪ax=axs1[index1], color="blue", label="prediction")
        scat1 = sns.scatterplot(x=cData[i][x],␣
↪y=cYData[i]["total_deaths_per_million"], ax=axs1[index1], color="green",␣
↪label="actual")
        scat0.set_title("Correlation: " + str(corr**2))
        scat0.set_ylabel("Deaths")
        scat0.set_xlabel(x)
        scat0.legend()
    plt.show()
```

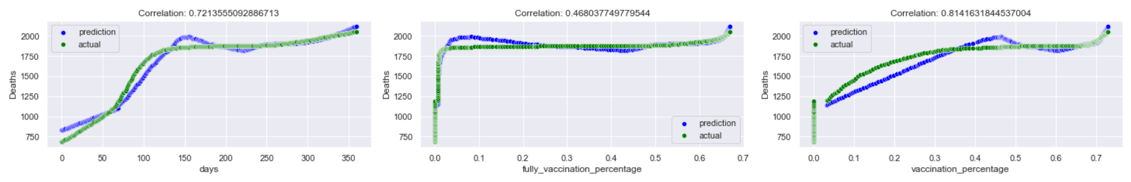
```
Singapore - Beta Coefficient:  ['1.53', '16.5', '35.72', '-33.61']
r^2 Value: [0.47] , Adjusted r^2 Value: [0.47]
```



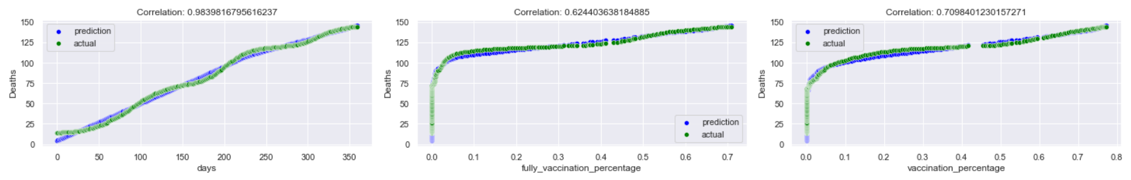
```
United States - Beta Coefficient:  ['839.37', '1379.16', '-1161.53', '1165.12']
r^2 Value: [0.96] , Adjusted r^2 Value: [0.96]
```



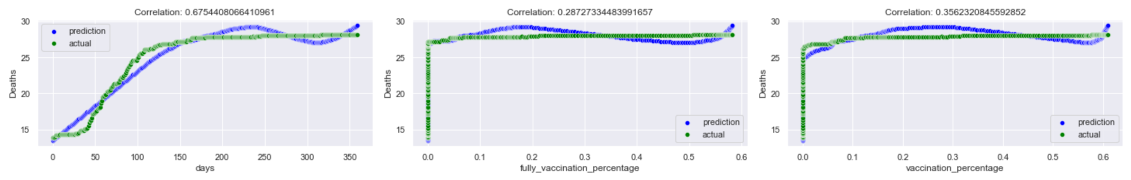
```
United Kingdom - Beta Coefficient:  ['826.75', '1422.71', '-1229.05', '1099.01']
r^2 Value: [0.97] , Adjusted r^2 Value: [0.97]
```



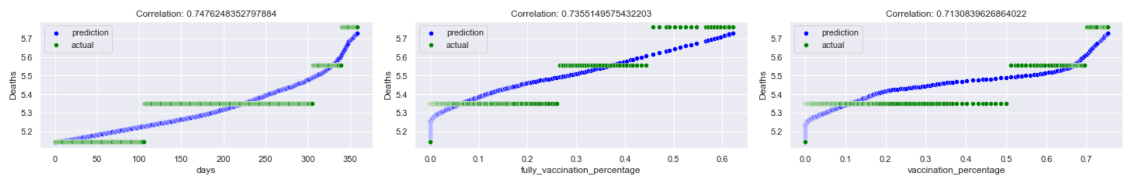
```
Japan - Beta Coefficient:  ['4.46', '163.35', '-40.1', '18.27']
r^2 Value: [0.99] , Adjusted r^2 Value: [0.99]
```



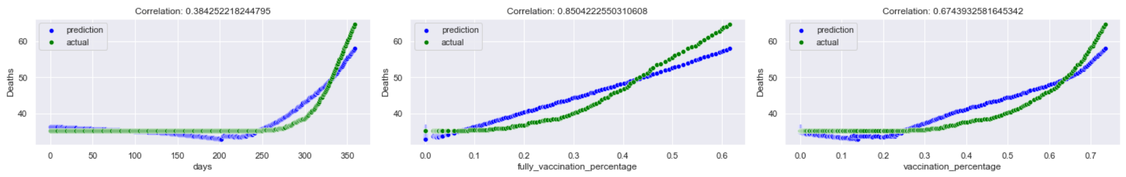
```
Hong Kong - Beta Coefficient:  ['13.46', '34.5', '-10.67', '-7.86']
r^2 Value: [0.95] , Adjusted r^2 Value: [0.95]
```



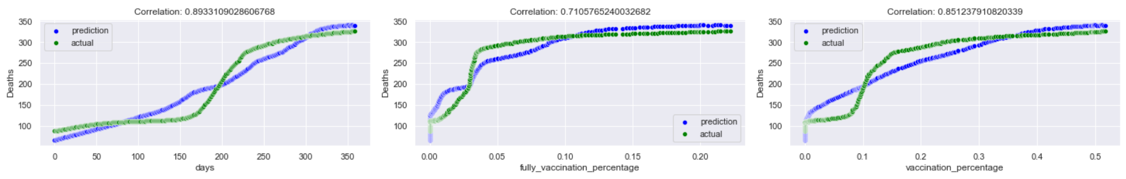
```
New Zealand - Beta Coefficient:  ['5.14', '0.3', '0.39', '-0.1']
r^2 Value: [0.82] , Adjusted r^2 Value: [0.82]
```



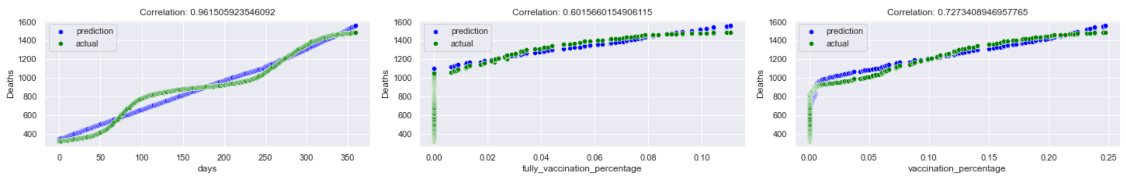
```
Australia - Beta Coefficient:  ['36.38', '-4.62', '31.03', '-4.82']
r^2 Value: [0.9] , Adjusted r^2 Value: [0.89]
```



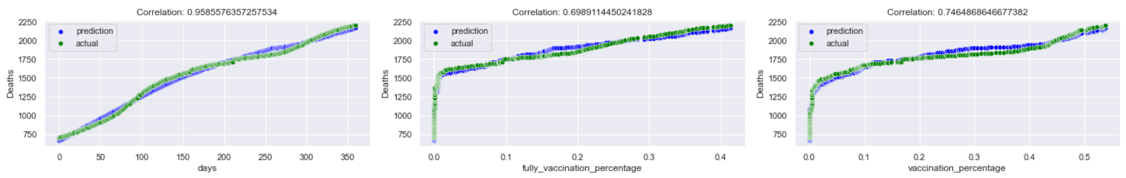
```
India - Beta Coefficient:  ['66.83', '188.3', '-147.72', '231.98']
r^2 Value: [0.93] , Adjusted r^2 Value: [0.93]
```



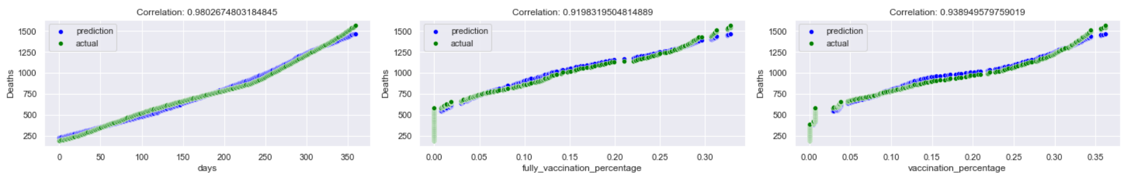
```
South Africa - Beta Coefficient:  ['343.88', '1136.66', '182.49', '-105.23']
r^2 Value: [0.97] , Adjusted r^2 Value: [0.97]
```



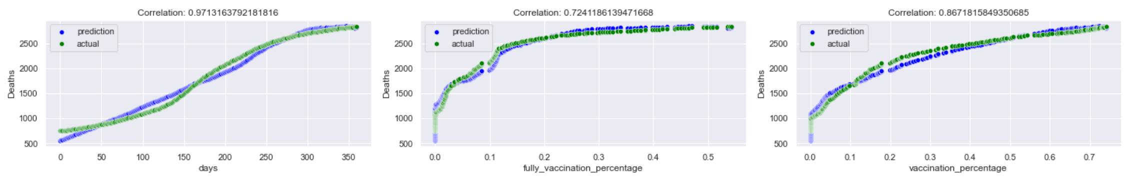
```
Mexico - Beta Coefficient:  ['664.59', '2137.92', '-258.86', '-372.65']
r^2 Value: [0.99] , Adjusted r^2 Value: [0.99]
```



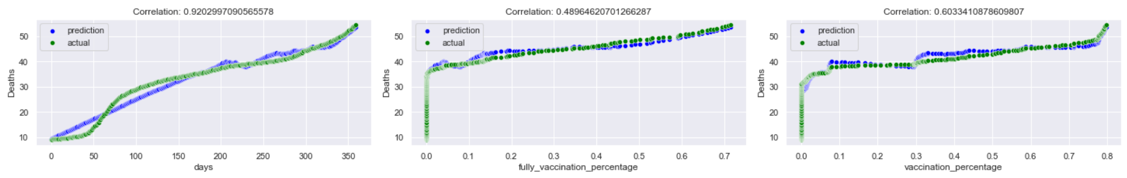
```
Russia - Beta Coefficient:  ['226.18', '912.94', '289.08', '41.8']
r^2 Value: [0.99] , Adjusted r^2 Value: [0.99]
```



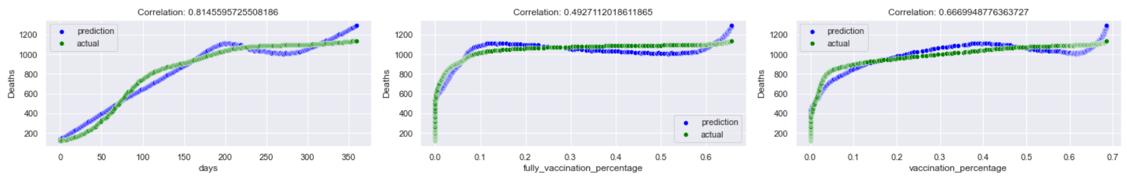
```
Brazil - Beta Coefficient:  ['552.14', '2362.65', '-1006.99', '907.86']
r^2 Value: [0.98] , Adjusted r^2 Value: [0.98]
```



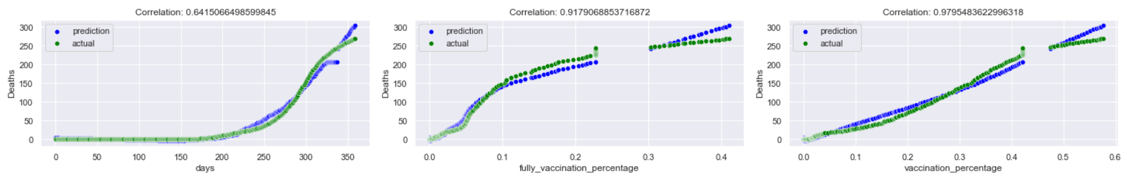
```
South Korea - Beta Coefficient:  ['9.44', '56.24', '11.46', '-23.6']
r^2 Value: [0.97] , Adjusted r^2 Value: [0.97]
```



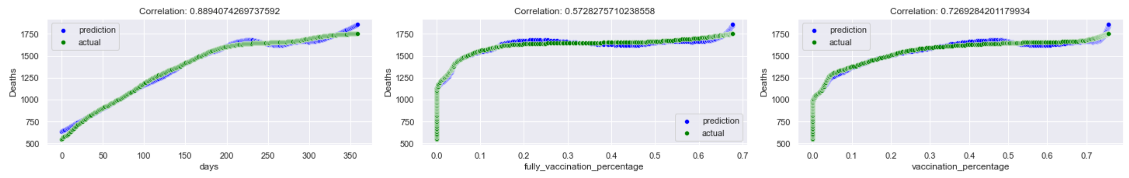
```
Germany - Beta Coefficient:  ['136.66', '1875.29', '-909.46', '189.37']
r^2 Value: [0.96] , Adjusted r^2 Value: [0.96]
```



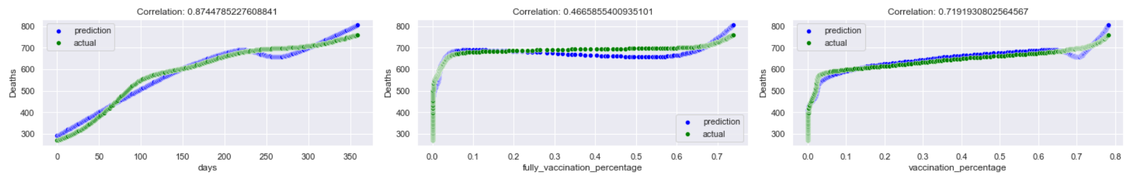
```
Thailand - Beta Coefficient:  ['3.97', '-18.99', '73.87', '245.91']
r^2 Value: [0.98] , Adjusted r^2 Value: [0.98]
```



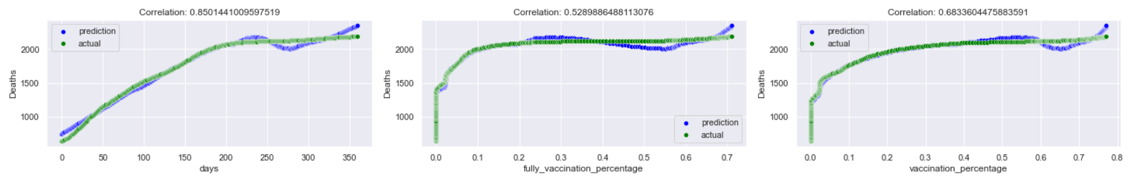
```
France - Beta Coefficient:  ['636.51', '1886.21', '-946.73', '281.29']
r^2 Value: [0.99] , Adjusted r^2 Value: [0.99]
```



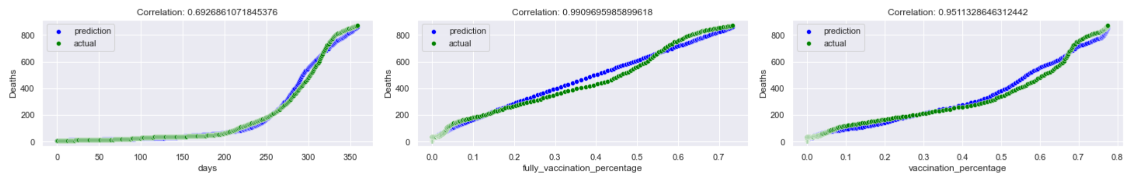
Canada - Beta Coefficient: ['290.53', '794.15', '-202.58', '-76.86']
r^2 Value: [0.97] , Adjusted r^2 Value: [0.97]



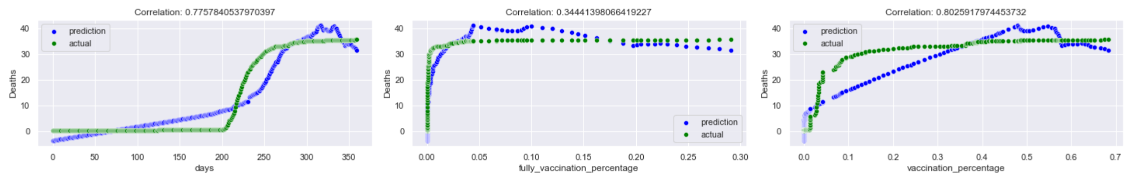
Italy - Beta Coefficient: ['749.88', '2663.94', '-1404.67', '336.71']
r^2 Value: [0.98] , Adjusted r^2 Value: [0.98]



Malaysia - Beta Coefficient: ['8.29', '52.88', '715.86', '82.15']
r^2 Value: [0.99] , Adjusted r^2 Value: [0.99]



Taiwan - Beta Coefficient: ['-3.64', '19.34', '-30.18', '45.85']
r^2 Value: [0.9] , Adjusted r^2 Value: [0.9]



Prediction for specific countries:

```
[ ]: #'Singapore', 'United States', 'United Kingdom', 'Japan', 'Hong Kong', 'New␣
     ↪Zealand', 'Australia', 'India', 'South Africa', 'Mexico', 'Russia',␣
     ↪'Brazil', 'South Korea', 'Germany', 'Thailand', 'France', 'Canada', 'Italy',␣
     ↪'Malaysia', 'Taiwan'
     # Inpuit country index from the above list and the days away from 1 November␣
     ↪2020
     countryIndex, deathInDays = 1, 600
```

```
[ ]: countries = list(cData.keys())
     val = cData[countries[countryIndex]].iloc[-1:].copy()
     val["days"] = deathInDays
     deaths = predict(normalize_predict(cData[countries[countryIndex]], val),␣
     ↪betaS[countries[countryIndex]])
     print(f'Predicting for {countries[countryIndex]} assuming vaccinated and fully␣
     ↪vaccinated did not change')
     print("Predicted deaths:", deaths[0][0], ", Last deaths recorded: ",␣
     ↪cYData[countries[countryIndex]].values[-1][0])
```

```
Predicting for United States assuming vaccinated and fully vaccinated did not
change
Predicted deaths: 3147.967195487273 , Last deaths recorded:  2219.416
```