# **Epidemiological modelling of spread of COVID-19 in Germany and South Korea**

The combined\_epimodel\_ML notebook contains the main code required for running all the models and the results included in our report. The data files used for the same are ROK\_data.xlsx and worldometer\_germany\_04-12-2020.xlsx, also attached in the report.

The second notebook which acts as a GUI for the user is Epimodel\_Covid19\_GUI, the notebook allows the user to process data for any region and a sample data file by the name of KOR\_data2.csv for the same has been attached.

Functions used in the notebooks:

* **SEIRD(y, t, N, beta, gamma, delta, mu)** :

Susceptible, Exposed, Infected, Recovered and Dead people being modelled for a country

Input: Initial numbers for Susceptible, Exposed, Infected, Dead and Recovered(Eg: y = N-100, 10, 100, 0, 0), number of days to be plotted(Eg: t = 70), values for contact rate(Eg: beta = 0.02), recovery rate(Eg: gamma = 0.02), lag time(Eg: delta = 0.02), mortality rate(Eg: mu = 0.02)

Output: Rate of change of numbers for Susceptible, Exposed, Infected, Recovered and Dead

* **SEIRD\_integrate(initial, t, params)** :

Integrating the SEIRD model over time period t specified

Input: Initial numbers for Susceptible, Exposed, Infected, Dead and Recovered(Eg: initial = N-100, 10, 100, 0, 0), number of days to be plotted(Eg: t = 70), parameter values (params = contact rate(Eg: beta = 0.02), lag time(Eg: delta = 0.02), recovery rate(Eg: gamma = 0.02), mortality rate(Eg: mu = 0.02))

Output: Array containing the value of y for each desired time in t, with the initial value *y0* in the first row.

* **SEIRD\_lockdown(y, t, l1, l2, l3, gamma, delta, mu)** :

Change of equations for the SEIRD model to factor in phase based change in contact rate for a country

Input: Initial numbers for Susceptible, Exposed, Infected, Dead and Recovered(Eg: y = N-100, 10, 100, 0, 0), number of days to be plotted(Eg: t = 70), values for reproductive number phase 1(Eg: l1 = 9.0), reproductive number phase 2(Eg: l2 = 4.0), reproductive number phase 3(Eg: l3 = 1.0), recovery rate(Eg: gamma = 0.02), lag time(Eg: delta = 0.02), mortality rate(Eg: mu = 0.02)

Output: Rate of change of numbers for Susceptible, Exposed, Infected, Recovered and Dead

* **SEIRD\_integrate\_lockdown(initial, t, params)** :

Integrating the SEIRD lockdown model over time period t specified

Input: Initial numbers for Susceptible, Exposed, Infected, Dead and Recovered(Eg: initial = N-100, 10, 100, 0, 0), number of days to be plotted(Eg: t = 70), parameter values (params = reproductive number phase 1(Eg: l1 = 9.0), reproductive number phase 2(Eg: l2 = 4.0), reproductive number phase 3(Eg: l3 = 1.0), recovery rate(Eg: gamma = 0.02), lag time(Eg: delta = 0.02), mortality rate(Eg: mu = 0.02))

Output: Array containing the value of y for each desired time in t, with the initial value *y0* in the first row.

* **get\_dataframe(result)** :

Function to save the imported data of countries to data frame

Input: A list of lists containing the value of y(Susceptible, Exposed, Infected, Dead and Recovered) for each desired time in t

Output: A dataframe with columns for Susceptible, Exposed, Infected, Dead and Recovered

* **model\_loss\_germany(params)** :

Function to calculate squared error loss between predicted and actual data for Germany

Input: Parameter values (params = contact rate(Eg: beta = 0.02), lag time(Eg: delta = 0.02), recovery rate(Eg: gamma = 0.02), mortality rate(Eg: mu = 0.02))

Output: Root of sum of squared error between predicted numbers and actual data for Germany

* **model\_loss\_sk(params)** :

Function to calculate squared error loss between predicted and actual data for South Korea

Input: Parameter values (params = contact rate(Eg: beta = 0.02), lag time(Eg: delta = 0.02), recovery rate(Eg: gamma = 0.02), mortality rate(Eg: mu = 0.02))

Output: Root of sum of squared error between predicted numbers and actual data for South Korea

* **model\_loss\_lockdown\_germany(params)** :

Function to calculate squared error loss between predicted and actual data taking into account the three phase model for Germany

Input: Parameter values (params = reproductive number phase 1(Eg: l1 = 9.0), reproductive number phase 2(Eg: l2 = 4.0), reproductive number phase 3(Eg: l3 = 1.0), recovery rate(Eg: gamma = 0.02), lag time(Eg: delta = 0.02), mortality rate(Eg: mu = 0.02))

Output: Root of sum of squared error between predicted numbers and actual data for Germany

* **model\_loss\_lockdown\_sk(params)** :

Function to calculate squared error loss between predicted and actual data taking into account the three phase model for South Korea

Input: Parameter values (params = reproductive number phase 1(Eg: l1 = 9.0), reproductive number phase 2(Eg: l2 = 4.0), reproductive number phase 3(Eg: l3 = 1.0), recovery rate(Eg: gamma = 0.02), lag time(Eg: delta = 0.02), mortality rate(Eg: mu = 0.02))

Output: Root of sum of squared error between predicted numbers and actual data for South Korea

* **parametrize() :**

Function to help find parameters that minimise loss when modelling spread of COVID-19 using machine learning model

Input: None

Output: Parameter values (reproductive number phase 1, reproductive number phase 2, reproductive number phase 3, recovery rate, lag time, mortality rate)

* **model\_loss\_lockdown\_up(params):**

Function to calculate squared error loss between predicted and actual data taking into account the three phase model for uploaded data

Input: Parameter values (params = reproductive number phase 1(Eg: l1 = 9.0), reproductive number phase 2(Eg: l2 = 4.0), reproductive number phase 3(Eg: l3 = 1.0), recovery rate(Eg: gamma = 0.02), lag time(Eg: delta = 0.02), mortality rate(Eg: mu = 0.02))

Output: Root of sum of squared error between predicted numbers and actual data for uploaded data

* **class analyze\_input:**
  + **def \_\_init\_\_(self, N, region\_name, region\_data, upload\_name):**

Initialise an instance of the uploaded data

Input: The population size in the region, name of the region for which data has been uploaded, the data file uploaded, name of the file being uploaded

Output: None

* + **def read\_input(self):**

Read input file

Input: Object instance

Output: Dataframe of the information in the file uploaded

* + **def visualise\_input(self):**

Visualise the numbers for Infected, Dead and Recovered in input data frame

Input: Object instance

Output: A plotly graph figure object visualising input data

* + **def analyze\_parameters\_manual(self):**

Find the best fitting parameters optimising on the model loss error between actual and predicted data using an iterative process over different ranges

Input: Object instance

Output: Parameter values(reproductive number phase 1, reproductive number phase 2, reproductive number phase 3, recovery rate, lag time, mortality rate), start day for phase 2, start day for phase 3 of public health interventions

* + **def analyze\_parameters\_ml(self):**

Find the best fitting parameters optimising on the model loss error between actual and predicted data using machine learning model

Input: Object instance

Output: Parameter values(reproductive number phase 1, reproductive number phase 2, reproductive number phase 3, recovery rate, lag time, mortality rate), start day for phase 2, start day for phase 3 of public health interventions

* + **def R0\_heatmap(self):**

Heatmap for visualisation of root of sum of squared error with different reproductive number values for phase 1 and phase 2

Input: Object instance

Output: Prints a seaborn heatmap visualisation

* + **def mortality\_recovery\_heatmap(self):**

Heatmap for visualisation of root of sum of squared error with different mortality and recovery rates for the region

Input: Object instance

Output: Prints a seaborn heatmap visualisation

* + **def actual\_predicted(self, L1, L2, params):**

Visualise the numbers for Infected, Dead and Recovered in input data frame versus the predicted curve obtained after fitting the parameters

Input: Object instance

Output: A plotly graph figure object visualisation