Kschool
Master's Degree in Data Science

Predicting the scope of a pandemic

-Master's Thesis-

By Albert Gil Martínez
September 4, 2020

Summary

Research on disease spread has a long tradition, and most are based on stochastic or deterministic mathematical models. These models are based on probabilities or initial conditions to determine the likely outcome of a pandemic. Machine Learning models have rarely been used to predict that outcome. It is due to the lack of pandemic data since these models are based on learning from a large amount of data, which is not available. This work suggests using the mathematical models to simulate a considerable amount of data about the spread of different diseases and use the data to feed a machine learning model. Of this way the ML model has enough data to be able to learn from them and make predictions from diseases never seen before.

In particular, for this purpose, I create a deterministic mathematical model based on the well-known SIR model. The model adapts to a global pandemic situation, allowing interactions between different countries and allowing quarantine in each one. The necessary data for the mathematical model are obtained from public sources and include information about countries, total population and number of international arrivals and departures. Moreover, I use data from airports around the world to know which routes exist between countries. These data allow creating an origin-destination matrix, simulating the movement of individuals between countries and being able to identify which countries are the most travelled and therefore, the primary sources of spread of the disease. Also, linking the information to a bidirectional graph, I compute features of each country based on graph theory, such as degree or betweenness.

Next, I explore a wide range of parameters, mixing different types of diseases, different focal points of the pandemic and quarantine intensities. Hence, it allows the mathematical model to generate distinct pandemic situations. Once the data are generated, the ML model learns from them. The features used in the ML model do not include any classic SIR model parameter, which cannot be calculated or are challenging to compute in a real pandemic situation. Furthermore, I use data from the first two weeks after the first deceased, to predict the outcome of the pandemic. So the input parameters of the ML model are not the same as those of the mathematical model, which complicates the learning process.

Finally, after the corresponding exploratory analysis of the data generated, I train different ML models, getting a satisfactory result with tree-based models and a neural network. The models that best suit the dataset are a XGBoost that let explains the 92 % of the variability observed in the response variable and a Neural Network that is capable of explaining the 93 % of the variability. Therefore this work concludes with an acceptable result, being able to identify the patterns that produce a pandemic outbreak.

Contents

1 Introduction

The Spanish flu, also known as the 1918 flu pandemic, was an unusually deadly influenza pandemic. Lasting from February 1918 to April 1920, infected 500 million people (about a third of the world's population at that time). The death toll is estimated to have been somewhere between 17 million and 50 million, which makes it one of the deadliest pandemic in history [1]. In the past several years, virology experts claim that we have to be prepared to face new health challenges, due to the emerging of new unknown viruses and that we are going to have to learn to live with some of them. They assert that if we do not act with prevention and with awareness of the danger in which we find ourselves, the results can be devastating [2][3]. There is no doubt that today pandemics are one of the greatest concerns of society, due to the COVID-19 virus it seems that people are beginning to realize the situation. Fortunately, when humanity works with a common goal, it can achieve extraordinary results. In 2020 all fields of science all turning to study the causes and effects of the pandemic produced by COVID-19, and what is learned from it, undoubtedly, it will help in the future to face new health global challenges.

Therefore, it is of public interest to know how diseases spread according to their characteristics. The aim of this work is to predict the scope of a pandemic with the help of mathematical modelling, data analysis, statistic techniques and machine learning models. On the other hand, research has found the important role of population heterogeneity and human mobility in the spread of infectious diseases [4], we live in an interconnected world where anyone can catch a plane and travel to the other side of the world. Consequently, this work emphasizes in the mobility of people between countries and it is applied to the models used.

It should be noted that this work is not intended to serve as a tool when facing a real pandemic. There are experts in the field who dedicate their entire lives to the study of infectious diseases and we have to thank them for their hard work and dedication. This work is posed with the question of whether a machine learning model is able to learn from data from past diseases and generalize when unknown diseases emerge.

1.1 State of the art

Mathematical models can forecast how infectious diseases progress to show the likely outcome of a pandemic. They try to predict how a disease spreads, the total number infected, the duration of an epidemic, and to estimate various epidemiological parameters such as the reproductive number. Mathematical models use basic assumptions or collected statistics beside with mathematics to find parameters for various infectious diseases and use those parameters to calculate, for instance, the effects of mass vaccination programmes.

There are two types of mathematical epidemic models: *stochastics* and *deterministics*. *Stochastic* models allow estimate probability distributions of potential outcomes by allowing random variation in one or more inputs. These kinds of models depend on the chance variations in the risk of exposure, disease and other dynamics. In *deterministic* models, on the other hand, individuals in the population are assigned to different subgroups or compartments, each representing a specific stage of the epidemic. The transition rates from one subgroup to another are mathematically expressed as derivatives, so the model uses differential equations. The progress in population compartments can be unequivocally calculated from the initial variables. When dealing with large populations, these models are preferred [5].

In 1927, W. O. Kermack and A. G. McKendrick created a deterministic model in which they considered a fixed population with three compartments: susceptible, $S(t)$; infected, $I(t)$; and recovered, $R(t)$. It was called *SIR model* [6]. The model is reasonably predictive for infectious diseases that are transmitted from human to human, and where recovery confers lasting resistance. It is one of the simplest compartmental models, and many models are derivatives of it. For instance, the SIR model with vital dynamics and constant population where it is considered a birth rate and a natural death rate; the SIS model where some infections do not give long-lasting immunity; the SIRD model which differentiates between recovered and deceased; the SEIR model when there is a significant incubation period during which individuals have been infected but are not yet infectious themselves; and the models that include vaccination. There are many others, but all derive from the same original SIR model from 1927 [7].

2 Methodology

This section describes the methodology used during the project and the steps followed to achieve the final result. First, I propose a solution for the lack of data, then I describe a mathematical model and the methodology used to collect the data. Finally I detail the steps in the machine learning process trying to predict the response variable.

The repository with all the code and figures can be found in `https://github.com/Albert-GM/TFM`

2.1 Lack of pandemics data

Real data about pandemics are very scarce or difficult to find. This may be due to the fact that major pandemics have occurred decades ago, when we still did not have the necessary means to collect the information generated in a methodical way and thus accessible data, if they exists, are for very specific diseases. In recent times more data is being collected, indeed, never before has so much real-time data been available about the spread of a disease like with the COVID-19.

However, the data available is not enough. Machine learning models require a large amount of data. In order to predict how an unknown disease will progress, we need data from many diseases with different characteristics. The machine learning model has to find the pattern in the data and be able to generalize to new diseases never seen before.

Given the lack of data, in this work I suggest a mathematical model described in Section 2.2 to simulate them. Once the data is simulated, a machine learning model can learn from them. Figure 1 shows a diagram of the process.

Figure 1: Data flow between models.

Since the ML model does not learn from data from real diseases, the predictions have the limitations associated with the mathematical model used to simulate the data. Section 2.2 describes the limitations and simplifications made.

2.2 SIRD model

Section 1.1 introduces the classic SIR model and others. In this work, I use the SIRD model. The model consists of four compartments:

- $S(t)$ is used to represent individuals not yet infected with the diseases at time t and are susceptible to the disease of the population.
- $I(t)$ refers to the individuals of the population who have been infected with the disease and are capable of spreading it to those in the susceptible category.

- $R(t)$ denotes the individuals who have been infected and then recovered from the disease. Thus, in this compartment individuals are not able to be infected again or to transmit the infection to others.
- $D(t)$ is the compartment for the deceased individuals due to the disease.

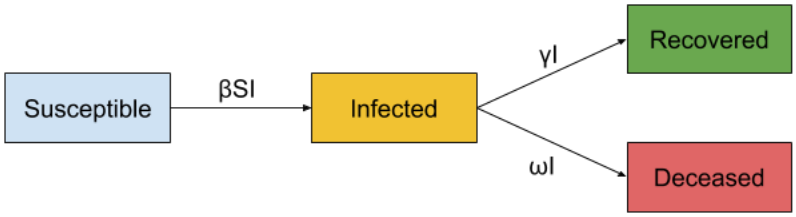


Figure 2: SIRD model transitions between compartments.

Each individual of the population progresses from susceptible, to infectious and to recovered or deceased, how shown in Figure 2. The arrows of Figure 2 are annotated with the transition rates between groups, ω refers to the disease mortality rate and the greek letters β and γ can be better understood if we say that the typical time between contacts is $T_c = \beta^{-1}$ and the typical time until recover is $T_r = \gamma^{-1}$. Finally, the model can be expressed by the following set of ordinary differential equations:

$$\frac{dS}{dt} = -\frac{\beta I S}{N} \tag{1}$$

$$\frac{dI}{dt} = \frac{\beta I S}{N} - \gamma I - \omega I \tag{2}$$

$$\frac{dR}{dt} = \gamma I \tag{3}$$

$$\frac{dD}{dt} = \omega I \tag{4}$$

Other important parameter is the basic reproduction number R_0. It is equivalent to the expected number of new infections from a single infection in a population where all subjects are susceptible, and it can be deduced from:

$$R_0 = \frac{\beta}{\gamma} = \frac{T_r}{T_c}$$

Note that the role of both the basic reproduction number, R_0 and the initial susceptible, $S(0)$, are extremely important since if $R_0 \cdot S(0) > N$ there will be an epidemic outbreak with an increase of the number of infected individuals that can reach a large fraction of the population. Whereas if $R_0 \cdot S(0) < N$, independently from the initial size of the susceptible population the disease can never start an epidemic outbreak. Consequently, if the susceptible population is all the population, $S(0) = N$, if $R_0 > 1$ there will be an epidemic outbreak.

The function `SIRD_model_simple`[1] implements the differential equations and produces a SIRD model given the initial inputs in Table 1 . The inputs have been selected for their ease of

[1] In `tfm-master/src/models/sird_model_simple.py`

N	Initial population
I_0	Initial infected
R_0	Reproduction number
T_r	Recovery time (days)
ω	Mortality rate (days^{-1})
T	Simulation time (days)

Table 1: Function inputs.

interpretation, for instance, parameters as γ or β are avoided as they are less intuitive and it is preferred to use others as R_0 or T_r that give the same information although it is needed to do some preliminary calculations.

Figure 7 shows the difference between two models where the only difference is the reproduction number. We can see clearly the weight of the parameter, multiplying it by a factor of two, the disease progress much faster and the maximum peak of infected is approximately triple. Moreover, the curve of infected individuals becomes much more pointed.

2.2.1 SIRD model modified

In this work, the SIRD model considers when populations of different countries interact between them. Therefore in the new model, there is a SIRD model for each country, and each of them has its own population of susceptible, infected, recovered and deceased. The interactions are defined as movements of individuals from one country (origin) to another (destination). The individuals at the moment of moving can be susceptible, infected or recovered and they become part of the destination population. The new model also enables countries to close the interactions with other countries with a reaction time ($Reaction \sim Exp(\lambda = R_t)$). This allows for modelling the quarantine of the countries and the containment of the disease.

Figure 8 shows the difference when trying to contain the progression of the disease. We can see in the world population how the peak without quarantine is more pointed, has a higher value and is reached earlier. Looking at the population compartments of the initial country we can see both are very similar, since once the disease begins to spread within a country, the model does not allow it to be contained. Last, we see that the number of infected countries in the model with quarantine is slightly lower.

The class `SIRD_model`[2] implements this new model and can be consulted in the work repository. Appendix B shows an extract of the code that computes a simulation. The inputs of the SIRD model are shown in Table 2.

2.2.2 Model considerations

The mathematical model used allows for some simplifications to make it possible to model a process as complex as the spread of a disease. The interactions between populations are individuals travelling by plane from one country to another, so flight data can be used to model the movement. The model assumes a constant population in all countries, that is, there are no births or deaths for

[2] In `tfm-master/src/models/sird_model.py`

I_0	Initial infected
R_0	Reproduction number (no units)
T_r	Recovery time (days)
ω	Mortality rate (days^{-1})
T	Simulation time (days)
$country_i$	Country where disease begins
K_c	Countries to close
R_t	React time (days)
OD	Origin-destination matrix
DF	Data about countries

Table 2: SIRD model modified inputs.

natural causes or for reasons other than the disease studied. Only the compartments described in Section 2 are considered and other heterogeneities of societies are excluded.

2.3 Disease data simulation

This subsection describes the process of generating the data that feeds the ML model, from its collection to its transformations.

2.3.1 Data collection

The data collected is public and is used as input to feed the mathematical model as shown in Figure 1. Of all the inputs in Table 2 we only need to collect data about the origin-destination matrix, OD, and information about the countries, DF. The rest of the parameters can be chosen arbitrarily by the user and during the massive data generation, a space of combinations of all the parameters is explored. The OD matrix is generated from arrivals and departures of each country and routes between airports. The information about the countries has to be related to features that can be used to model the spread of a disease in a population. Hence, the files used to extract the raw data are [3]:

- `world_indicators_data.csv`: data about relevant and internationally comparable statistics about global development. It contains 1 600 time series indicators for 217 economies, included number of arrivals, departures and population.
- `country_population.csv`: supplementary population data of some countries.
- `routes.csv`: contains 67 663 routes between 3 321 airports spanning the globe. Routes are directional and unique.
- `airport_codes.csv`: maps the IATA code airport to the country to which it belongs.
- `country_to_continent.csv`: maps the name of the country with the continent.
- `table_convert_iso.csv`: maps the name of the country with the ISO 3166-1 alpha-3 and alpha-2 codes and with its location in coordinates.

[3] Raw data in `tfm-master/data/raw`

2.3.2 Data transformation

In Figure 3 is shown the data flow between files through the data transformation process[4]. Blue rectangles represent scripts, yellow rectangles represent inputs and green rectangles represent outputs.

- `make_routes.py`: returns to graphs, one with information at the country level of detail and other with information at the airport level of deatil. The graph contains the information about which countries are connected between them, countries are the nodes and the edges are the routes between those countries.
- `make_country.py`: returns a data frame with information about all the countries on the world, for instance number of arrivals, departures and total population.
- `fill_nans_countries.py`: returns a data frame without missing values. In the data frame returned in the previous step there are few missing values. A random forest regressor predicts those values instead of using the typical approach of fill with means or means.
- `modelling_movement.py`: returns the previous data frame but adding information about how many individuals travel each day from one country to another and the origin-destination matrix.

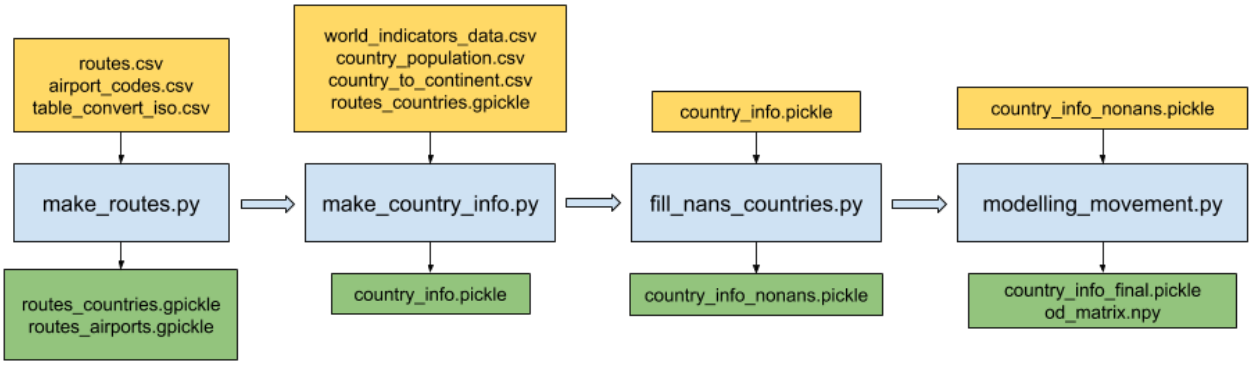


Figure 3: Data flow in the data transformation process.

Notice that `country_info_final.pickle` and `od_matrix.npy` are DF and OD respectively from Table 2. In this way, all the necessary data to feed the SIRD model are prepared.

2.3.3 Modelling movements

In order to model the movements of individuals a non-symmetric and constant origin-destination matrix (OD), is computed from the information about arrivals per year and departures per year of each country in `world_indicators_data.csv` as follows[5]

[4] Data transformation scripts in `tfm-master/src/data`

[5] In `tfm-master/src/data/modelling_movement.py`

A_i	arrivals per year of country i
D_i	departures per year of country i
a_i	arrivals per day of country i
d_i	departures per day of country i
pa_i	probability of arrival to country i
p_{ij}	probability of going from i to j
I_{ij}	individuals going from country i to country j
S_i	set of possible destinations o country i
N	total number of countries
OD	origin-destination matrix

$$a_i = \frac{A_i}{365} \qquad \text{for } i = 1, 2, \dots, N$$

$$d_i = \frac{D_i}{365} \qquad \text{for } i = 1, 2, \dots, N$$

$$pa_i = \frac{a_i}{\sum_{i=1}^{N} a_i} \qquad \text{for } i = 1, 2, \dots, N$$

$$p_{ij} = \frac{pa_i}{\sum_{j \in S_i} pa_j} \qquad \text{for } i, j = 1, 2, \dots, N \text{ and } i \neq j$$

$$I_{ij} = d_i \cdot p_{ij} \qquad \text{for } i, j = 1, 2, \dots, N \text{ and } i \neq j$$

$$OD = \begin{bmatrix} I_{11} & I_{12} & \dots \\ \vdots & \ddots & \\ I_{n1} & & I_{nn} \end{bmatrix}$$

2.3.4 Parameter space exploration

The input of the ML model has to represent the whole space of possible pandemic scenarios to be able to generalize. Therefore, a generator is used on parameters sampled from the following distributions, thinking in typical disease values: R_0 and T_r from a uniform continuous distribution; ω from a truncated exponential continuous distribution; and finally L_d, K_c and R_t from uniform discrete distributions. The generator iterates over random candidate combinations for hyper-parameter search. In Figure 4 samples of those distributions are shown. The generator also samples $Country_i$ and therefore N from the set of possible countries. The rest of parameters are constant in all the simulations despite can be changed by the user, for instance: $I_0 = 1$ and $T = 730$ days (2 years).

Once the input parameter space of the SIRD model is generated, all the simulations are carried out, in particular $\mathbf{X}$ simulations are computed to feed de ML model[6]. The most important outputs of the model are shown in Table 3.

[6] In `tfm-master/src/features/make_simulation.py`

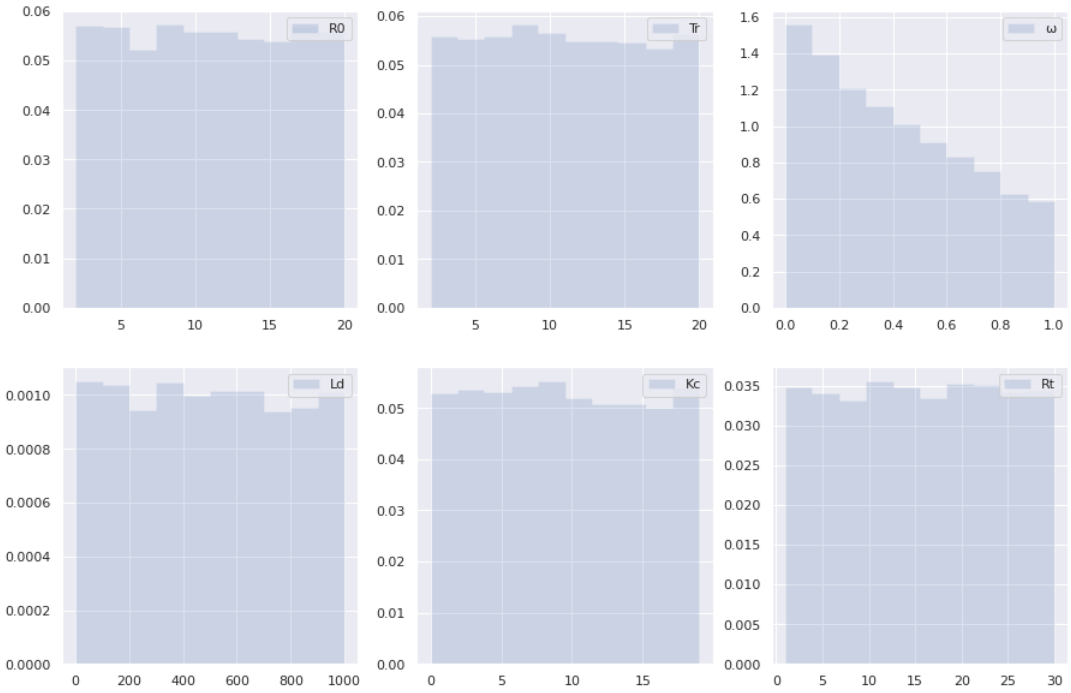


Figure 4: Samples of the distributions used for generating the parameter space of the SIRD model.

Name	Description	Size
$new\ infected\ world_t$	Global number new infected in each instant t	[1,T]
$new\ recovered\ world_t$	Global number new recovered in each instant t	[1,T]
$new\ deceased\ world_t$	Global number new recovered in each instant t	[1,T]
$new\ infected_t$	Number of new infected in each instant t by country	[N,T]
$new\ recovered_t$	Number of new recovered in each instant t by country	[N,T]
$new\ deceased_t$	Number of new recovered in each instant t by country	[N,T]
$SIRD\ world_t$	Global number of individuals in each compartment at t	[4, T]
$SIRD_t$	Number of individuals in each compartment at t by country	[4, N, T]
$total\ infected$	Number of total infected	1
$total\ recovered$	Number of total recovered	1
$total\ deceased$	Number of total deceased	1

Table 3: Subset of outputs of the SIRD model.

2.4 Machine Learning

The goal of the machine learning model is to predict the number of deceased individuals by an arbitrary infectious disease. Consequently, the ML model uses features computed from the data generated by the SIRD model. Those features have to be available during the first stage of disease progression, if not, the predictions would not make much sense and they would be useless. Therefore, it is not possible to use features such as R_0 or ω, since they are parameters usually calculated once there is a lot of information about the disease. Furthermore, the parameter T_c, defined in Section 2.2 as time between contacts, is very challenging to calculate in a real scenario, because the time between contacts of individuals where the disease is spread can not be monitored.

We are interested in a model able of explaining the variation in the target variable, consequently, the performance of the different models will be measured with the coefficient of determination, denoted as R^2 or r^2. This coefficient measures the proportion of the variance in the target variable that is predictable from the explanatory variables, in this case, from the different features of the model. The models are evaluated with a train-validation set, using cross-validation. From the beginning, once we get the data from the SIRD model, we set aside a test set since the human brain is an amazing pattern detection system, it is highly prone to overfitting. So if we look at the test set, we will be overfitting it, and when we estimate the generalization error using the test set, the estimation will be too optimistic. This is called *data snooping* bias.

In particular, different types of models are tested to see which of them best suits the characteristics of the dataset and is capable of extracting the pattern that generates the target variable. Once the models are trained, in a first approach, I analyse the most significant variables for each algorithm and the types of errors the models make. Based on these errors, I simulate the SIRD model with parameters that the ML model has more difficulty predicting the target. Moreover, feature engineering uses those errors to find new features that may improve the model. Next, I fine-tune the system using cross-validation, preferring random search over grid search when there are a lot of hyperparameters to explore. It is important to use as much data as possible with the best models and try to automate the steps. Finally once I am confident with the final model I measure its performance on the test set to estimate the generalization error. The test set is composed of 120 000 samples and the train-validation set is composed of about 500 000 samples.

2.4.1 Feature engineering

Since, as previous mentioned, we can not use parameters R_0, T_c and *omega* to train the model. Hence we need to found other parameters capable of providing us with the same information.

In this work, it is assumed that an infectious disease really becomes known, when it causes a serious health condition in an individual or when it even die. It could be that someone is infected but if it does not cause serious symptoms on her/him, the virus is not given much importance. Therefore, it is considered that the information of the new disease begins to be monitored, from the first death, which is a fairly realistic assumption. From then on, the calculations of the parameters that define the power of the pandemic are carried out with data from the first two weeks after the first death, longer would be too late to know how the diseases is spreading. Hence, the following parameter related to the infection power are defined (notice that $SIRD\ world_t$ keeps track of the individuals in each compartment at t whereas $new \dots world_t$ keep tracks of the new infected, recovered or deceased in that t):

day_a	Day of the first decease
day_b	Day of the first decease plus 2 weeks
$inf\ pow_1$	Approximate slope of infected in the first two weeks
$inf\ pow_2$	Infected percentage change

$$inf\ pow_1 = \frac{\sum new\ infected\ world_t[day_a : day_b]}{day_b - day_a} \tag{5}$$

$$inf\ pow_2 = \frac{SIRD\ world_t[1, day_b] - SIRD\ world_t[1, day_a]}{SIRD\ world_t[1, day_a]} \tag{6}$$

Also, we define the parameters related to the mortality power[7]

$mort\ pow_1$	Percentage of new deaths compared to new infected
$mort\ pow_2$	Percentage of new deaths compared to the variation of number of infected
$mort\ pow_3$	Percentage of new deaths compared to the variation of number of recovered

$$mort\ pow_1 = \frac{\sum new\ deceased\ world_t[day_a : day_b]}{\sum new\ infected\ world_t[day_a : day_b]} \tag{7}$$

$$mort\ pow_2 = \frac{\sum new\ deceased\ world_t[day_a : day_b]}{SIRD\ world_t[1, day_b] - SIRD\ world_t[1, day_a]} \tag{8}$$

$$mort\ pow_3 = \frac{\sum new\ deceased\ world_t[day_a : day_b]}{SIRD\ world_t[2, day_b] - SIRD\ world_t[2, day_a]} \tag{9}$$

Additionally, Section 1 mentions the importance of human mobility in the spread of infectious diseases, so countries that are more connected with others will be sources of spread of the disease, and if the first infected is in those countries, or infected individuals reach those countries, the disease will spread out quickly. Using a feature able to capture this characteristic, the ML model will be able to share the knowledge learned from that sample with samples where the initial country has similar connectivity. Moreover, the $country_i$ has a large number of possible categories, one for each of the 204 countries in the dataset, so the one-hot encoding will would result in a large number of input features that may slow down training and degrade the performance of the ML model. Replacing the categorical input with numerical features related to the category can be extremely useful.

We can think of the set of countries as a bidirected graph, where the nodes are countries and the edges are routes between countries. The countries most connected with others will be those that have a greater importance within the network. In this case, the importance of a node within a graph can be measured with different parameters, some of them are:

- **Degree**. It is the number of edges that are incident to the node. Meaning that important nodes have many connections.

[7] Disease features in `tfm-master/src/utils/sird_support.py`

- **Betweenness**. It is a measure of centrality in a graph based on shortest paths. Meaning that important node connect other nodes.
- **Closeness**. It is also a measure of centrality of a node in a network. Meaning that important node are close to other nodes.

These parameters computed for $country_i$ are added as features. For more information about them and how are calculated check [8]. In particular, in this work, using a library for the manipulation and study of networks these parameters are calculated automatically from the file `routes_countries.gpickle` presented in Section 2.3.2.

Finally, other parameters related to the the country where the disease begins are added:

- The total population of the initial country. Despite many countries being confined, the initial country will be completely affected by the disease.
- The total number of departures per day from the initial country. It represents the spread of the disease to other countries.
- The sum of the population of the countries that are possible destinations from the initial country. It represents the main population of other countries different from the initial exposed to the disease.

In conclusion, the features that can be used to feed the ML model are: related to the disease T_r, $infpow_1$, $infpow_2$, $mortpow_1$, $mortpow_2$, $mortpow_3$; related to the quarantine K_c and R_t; and finally related to the spread of the disease *degree*, *betweenness*, *closeness*, *population* $country_i$, *departures* $country_i$ and *exposed population*[8].

[8]Spread and quarantine features in `tfm-master/src/features/add_features.py`

3 Analysis

This section presents analyses and processes that have been significant in obtaining the final model. First the exploratory analysis of the data generated by the SIRD model is described. Second, different modelsare explored and the best ones are selected. Finally the best models are fine-tuned to achieve the final solution. From now on I will use the code notation to name the variables.

3.1 Data Exploration

Once the target value is defined, it is essential to see its distribution. Figure 5 show that the response variable is a bit unbalanced, sometimes it reach high values, but most are usually beyond $0,5 \times 10^9$ (note that world population around 2019 was $7,7 \times 10^9$, so they really are relative high values). Next, it is necessary to explore the data in order to see any statistical relationship or pattern. In particular, it is of special interest the correlation of the target with the rest of the features. Note that correlation only indicates a linear relationship between variables, so if there is another type of relationship, but it is not linear, it will not be able to capture it. Therefore, it is necessary to plot the response variable against the rest of the features to see if we can observe some pattern.

Figure 9 shows the correlations between features. In a first approach, data reveals that the features with more predictive power are `R0`, `Tc`, `omega` and `Tr`. Unfortunately how mentioned in Section 2.4, three of them can not be used. Furthermore, from the features created in Section 2.4.1, trying to capture the same information as `R0` and `omega`, it seems that only `mort_pow_3` has a higher correlation with them. On the other hand, the network features do not show a linear correlation with the target value.

However, applying a simple logarithmic transformation to the disease features, we get a strong improvement and a high correlation with the target value. Indeed, the correlation with the target value is higher in all the logarithmic transformations of the new features than in the classic parameters from the SIRD model. The data suggest that are features with predictive power, although some of them may provide the same information about the pandemic due to the correlation they have between them. Although, it will depend on the trained ML model, in particular, the decision about what features are useful to predict the target.

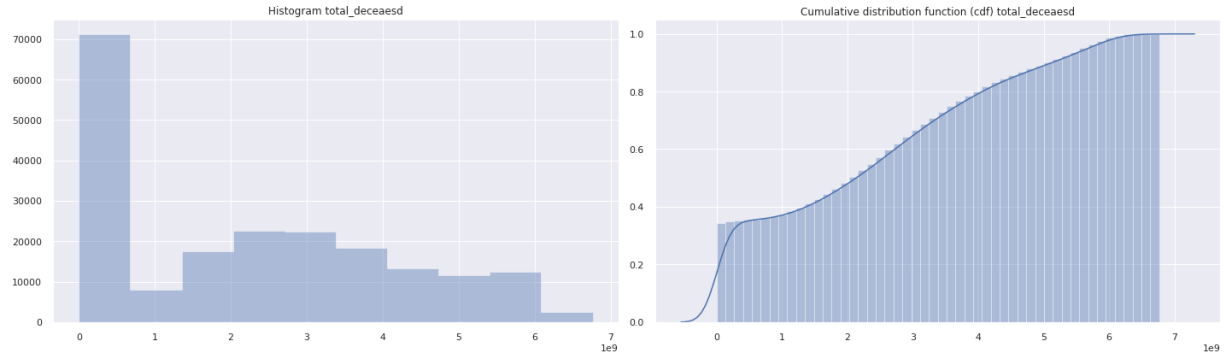


Figure 5: Target distribution (histogram and cdf).

3.2 ML Models

The idea is to test many different models in a reasonable time, to see which ones are the most promising. Sometimes it is needed to take smaller subsets of data, this penalizes complex models such as Random Forest or large Neural Networks. Hence, I test models from different categories such as Linear Regression, Support Vector Regression, Decision Trees, Random Forest, XGBooost and Neural Networks[9].

Figure 6 shows the score, coefficient of determination R^2, and the standard deviation of different ML models . Note that the score is the mean on all the test splits of the cross-validation evaluation. Decision Tree, Random Forest and Xgboost get the highest scores, being Xgboost the model with a highest R^2. Linear models as the SGD Regressor or the Ridge Regressor do not have a good performance in this dataset, although they perform better than a Dummy Regressor which always predicts the mean of the target value or the SVR that does not learn anything at all. The exact values can be seen in Table 4.

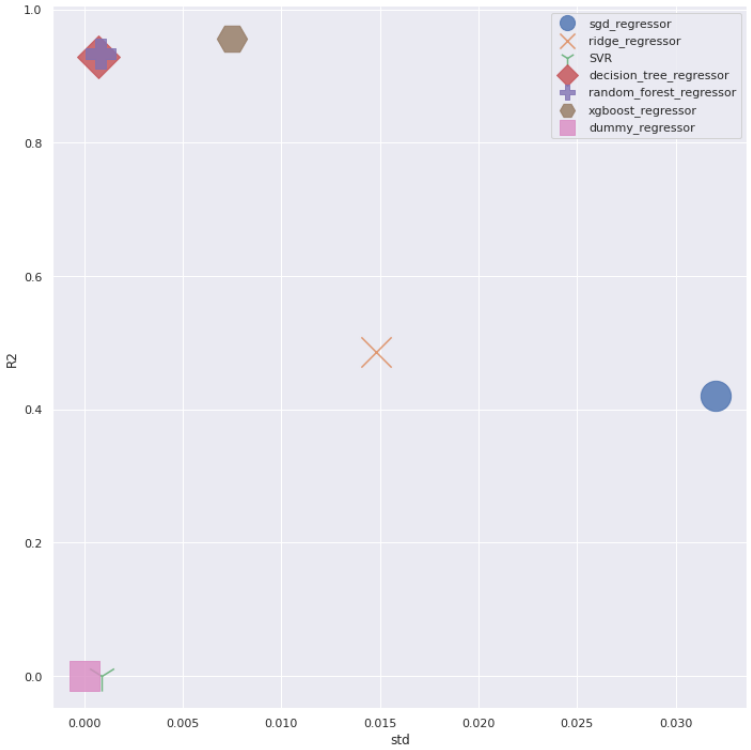


Figure 6: Scores of different models, cross-validated. Y-axis represents the coefficient of determination and x-axis the standard deviation.

The results demonstrate that tree-based models outperform linear regressions and support vector regressor. The reasons are that the relationships between the features and the target are not linear, given the nature of the SIRD model created. It is true that as the infectious or mortality power of the disease increases, the deceased individuals tend to be higher, but it depends on other factors such as the connectivity in the network of the initial country or whether quarantine occurs. On

[9]Models in `tfm-master/src/models/`

Model	R^2	SD
Dummy Regressor	6.92e−6	4.33e−6
SGD Regressor	0.420	0.03295
Ridge Regressor	0.487	0.01482
SVR	6.27e−4	8.81e−4
Decision Tree	0.928	7.32e−4
Random Forest	0.935	8.34e−4
XGboost	0.956	7.53e−3

Table 4: Scores of different models, cross-validated.

the other hand, tree-based models are able to learn non-linear relationships easily and they divide better the hyperspace according the input parameters.

In Figure 10 we can see the feature importance of the different models that support `feature_importances_` or `coef_` attributes. Note how Ridge Regressor assigns low weights to a lot of features, due to his normalization component. Leaving aside the models that perform worse, from the tree-based models, the features `mort_pow_3_log` and `inf_pow_2` are the most relevant, despite random forest gives weight to `inf_pow_2_log`. Therefore, it is clear that the most important features seem to be those related to the characteristics of the disease. Of the features related to the quarantine, both `n_closed` and `react_time` are of similar importance. Last of the features related to the connectivity of the initial country, Xgboost uses `country_departures` and `degree` most.

3.2.1 XGBoost

I showed that excluding Neural Networks, XGBoost is the ML model which best results obtained. This model provides parallel tree boosting, that is a highly effective and widely used machine learning method [9]. Indeed, from the learning curve of the model in Figure 11 we can see how if we feed the model with more data the score would continue increasing very slowly, this improvement also takes place with other tree-based models. In the last 50 000 samples, the increase of R^2 has been only around 0,005. Analysing the residuals of the model, Figure 12, we can see how the model predicts above the real value when the target is low, however the distribution of the residuals is approximately centered at 0.

Finally, we measure the performance of the model with the test set we set aside in Section 2.4, scoring $R^2 = 0.9235$, $RMSE = 6.19\text{e+}8$ and $MAE = 2.32\text{e+}8$. As it usually happens, the generalization error is a bit lower than the validation error. Figure 13 shows 50 random predictions done by the model in the test set.

3.2.2 Neural Networks

Last but not least, we train a neural network with the dataset. This trendy model allows to tune a lot of parameters, so it is not a simple task. Some of them are the number of hidden layers, the number of neurons per hidden layer, the learning rate, the optimizer, the batch size, the activation function and the number of training iterations. There are a large number of combinations that can be made between the parameters, therefore it is needed to follow a methodical procedure to explore

the most of them.

In this work I test deep neural networks with different combinations of hidden layers and number of neurons per hidden layer, always with a final layer with an unique nueron due to we are in a regreession supervised learning task. The configuration used is: LeCun initialization as a kernel initializer, SELU as a activation functions, early stopping as regularization and Nadam as the optimizer. I choose Nadam because it is composed by Adam optimizer, so is an adaptative leraning rate algorithm and it requires less tuning of the learning rate hyperparameter, moreover sometimes it converges slightly faster than Adam [10]. Using SELU as activation functions allows the network self-normalize, meaning the output of each layer will tend to preserve mean 0 and standard deviation 1 during training, which solves the vanishing/exploding gradients problem, additionally the input features must be standarized and the network's architecture must be sequential [11].

The best neural network achieved a $R^2 = 0.9668$ in the validation set, with an architecture of 7 hidden layers and 80 neurons per hidden layer. Figure **??** shows the coefficient of determination and epoch loss for three different types of architectures including the best one mentioned. From the data we can see how the shallower neural network takes longer to reach an acceptable score, whereas as we increase the depth, the loss becomes smaller and the model needs less epochs to learn from the data. The performance in the test set is: $R^2 = 0.9305$, $RMSE = 5.79\text{e+}8$ and $MAE = 1.82\text{e+}8$. Figure 15 shows some samples test predictions of the neural network.

4 Conclusions

In conclusion, it has been possible to create a mathematical model capable of representing, with certain simplifications, the spread of disease among the different countries of the world. Additionally, a ML model has been trained with the data produced by the model to predict the number of deceased individuals in a pandemic, being able to generalize to new unknown diseases. I tested different ML models, however, the ones that obtained the best results have been those based on trees and neural networks. The Xgboost, described in Section 3.2.1, explains the 92,3 % of the variability observed in the response variable, moreover, a neural network described in Section 3.2.2, is capable of explaining the 93 % of the variability. The results from the tree-based models suggest that by feeding these models with more data, the coefficient of determination continue increasing slightly, nevertheless, the computations cost that this entails must be taken into account and assess the performance with which we are satisfied. It would be interesting to study to what extent these models continue learning and if there comes a point where the loss in the validation set begins to increase, indicating we are overfitting the training set.

The limitations of the ML model, in addition to those of this work, are related with the limitations of the deterministic SIR model, presented in 2.2. The main difficulties faced have been to be able to create a model capable of simulating pandemic data, in a sufficiently realistic way, and on the other hand, to find a ML model that adatps to the data set simulated. However, I consider that a acceptable result has been achieved, being able to predict the outcome of a global pandemic.

4.1 Future work

Future work could fruitfully explore the increase in complexity of the mathematical model, sot it would be able to more reliably represent the spread of a pandemic. Moreover, it would be useful to increase the number of simulations to produce more data to feed the tree-based models, since I commented, data suggest the performance could increase. Finally working in feature engineering could boost the performance of the ML models.

References

[1] Antoni Trilla, Guillem Trilla, and Carolyn Daer. The 1918 spanish flu in spain. *Clinical infectious diseases*, 47(5):668–673, 2008.

[2] Richard Levins, Tamara Awerbuch, Uwe Brinkmann, Irina Eckardt, Paul Epstein, Najwa Makhoul, Cristina Albuquerque de Possas, Charles Puccia, Andrew Spielman, and Mary E Wilson. The emergence of new diseases. *American Scientist*, 82(1):52–60, 1994.

[3] Peter Palese. Influenza: old and new threats. *Nature medicine*, 10(12):S82–S87, 2004.

[4] Stefano Merler and Marco Ajelli. The role of population heterogeneity and human mobility in the spread of pandemic influenza. *Proceedings of the Royal Society B: Biological Sciences*, 277(1681):557–565, 2010.

[5] Fred Brauer, Carlos Castillo-Chavez, and Carlos Castillo-Chavez. *Mathematical models in population biology and epidemiology*, volume 2. Springer, 2012.

[6] William Ogilvy Kermack and Anderson G McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

[7] Linda JS Allen, Fred Brauer, Pauline Van den Driessche, and Jianhong Wu. *Mathematical epidemiology*, volume 1945. Springer, 2008.

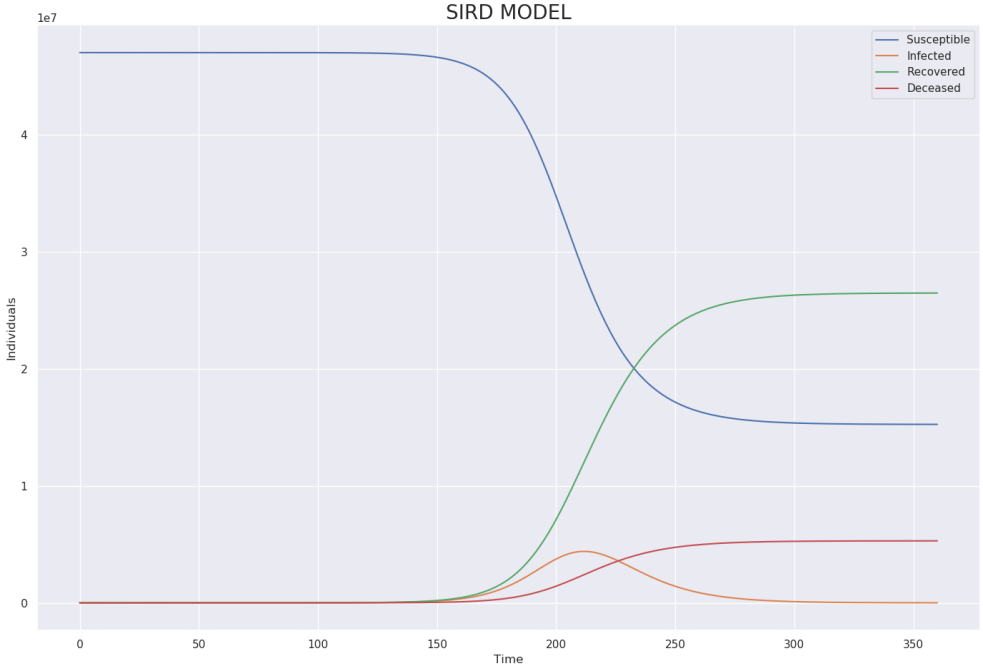
[8] Florian Boudin. A comparison of centrality measures for graph-based keyphrase extraction. In *Proceedings of the sixth international joint conference on natural language processing*, pages 834–838, 2013.

[9] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[10] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[11] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

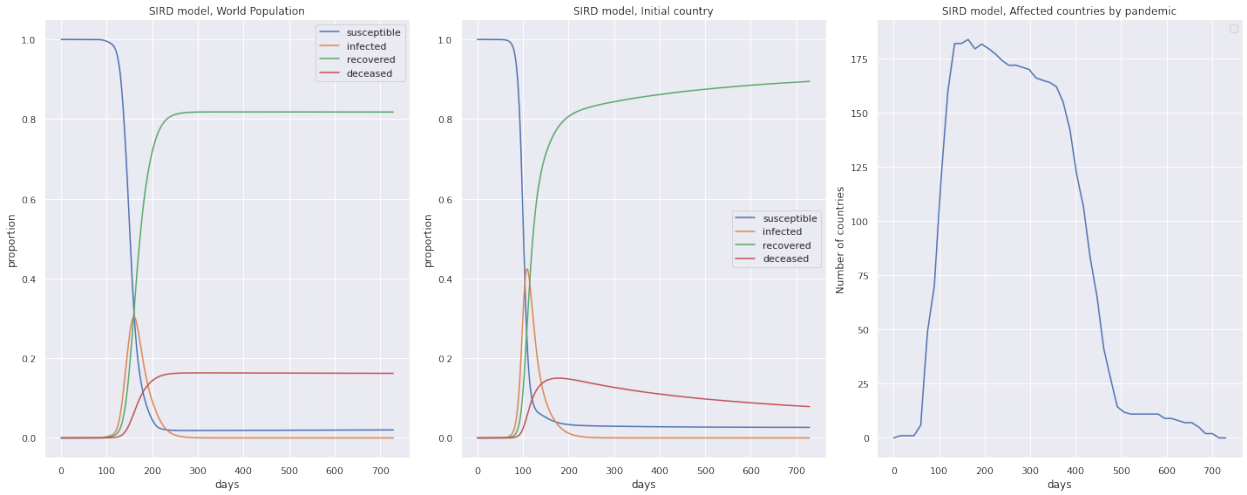
A Graphics



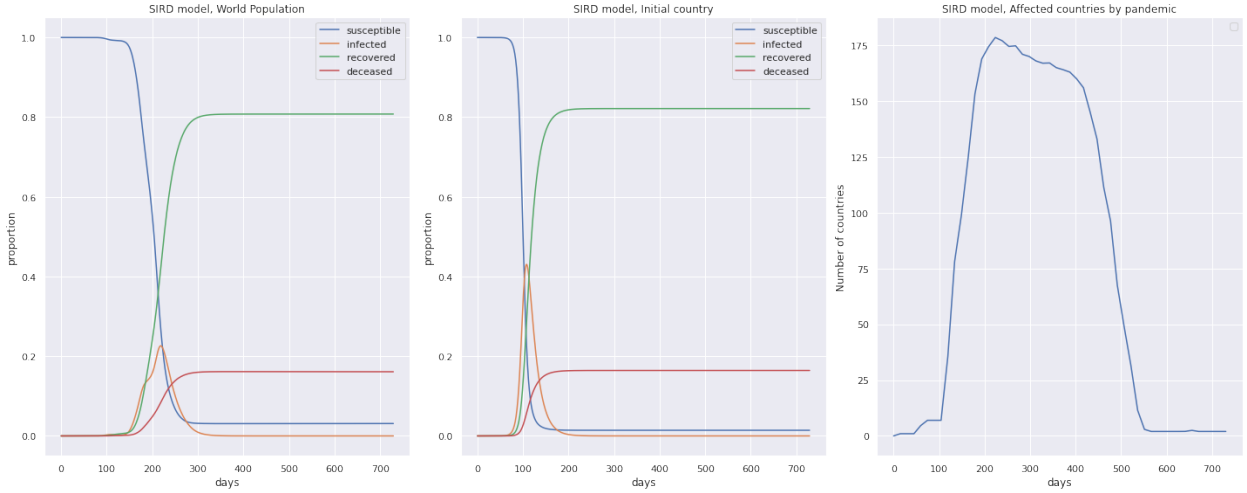
(a) $R_0 = 2$

(b) $R_0 = 4$

Figure 7: Difference between two SIRD models, multiplying R_0 by a factor of two. Input parameters: $N = 4,7 \times 10^7$, $I_0 = 1$, $T_r = 10$, $\omega = 0,01$ and $T = 360$



(a) No quarantine



(b) With quarantine, $K_c = 5$ and $R_t = 5$

Figure 8: Difference between two SIRD models, one with quarantine and other without. First figure represents the progression of the disease in the world population, second figure represents the progression of the disease in the initial country and finally the last figures shows the number of countries where at least one individual is infected at each t. Input parameters: $R_0 = 5$, $T_r = 20$, $\omega = 0,01$, $country_i$=Spain, $I_0 = 1$.

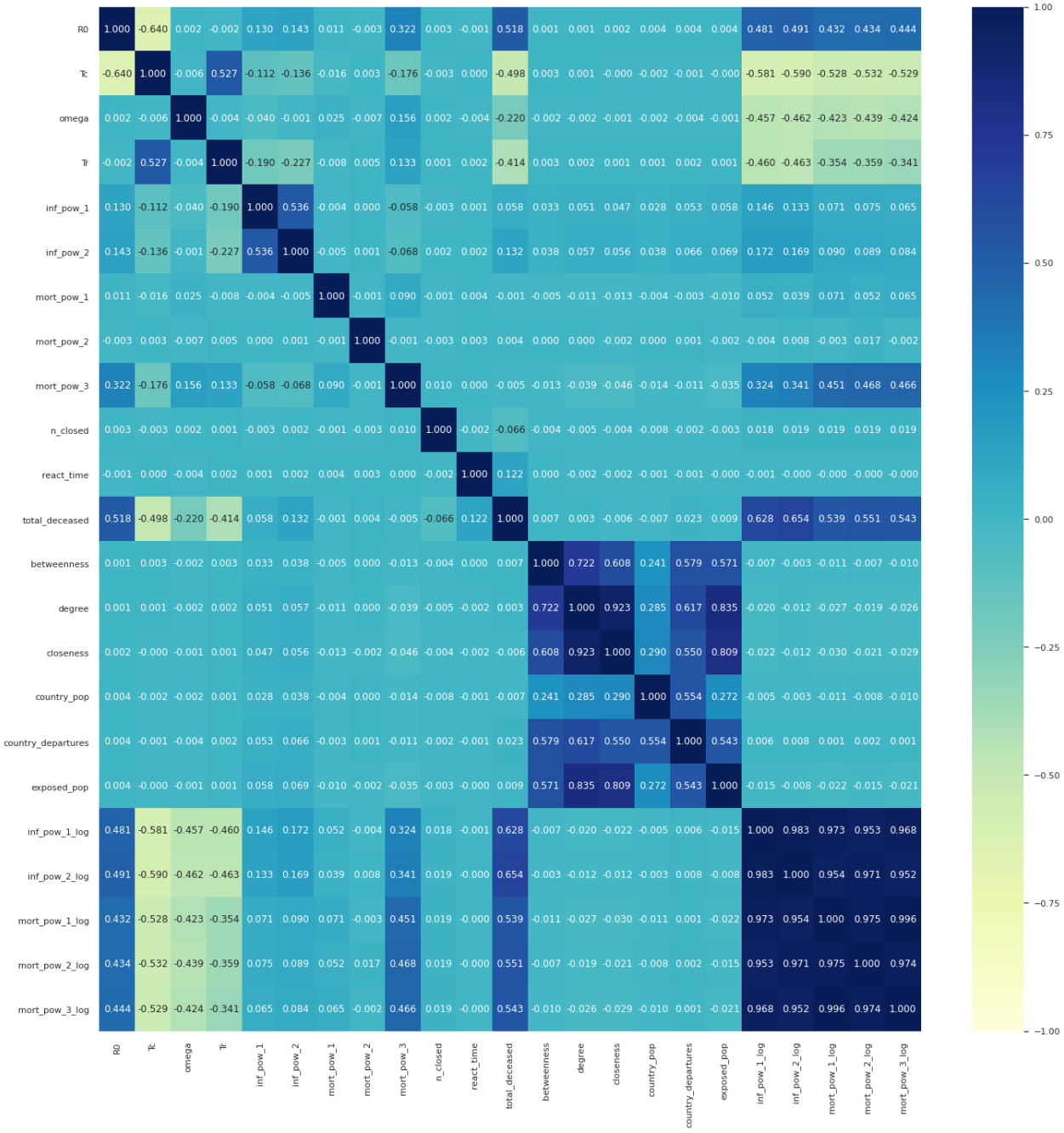
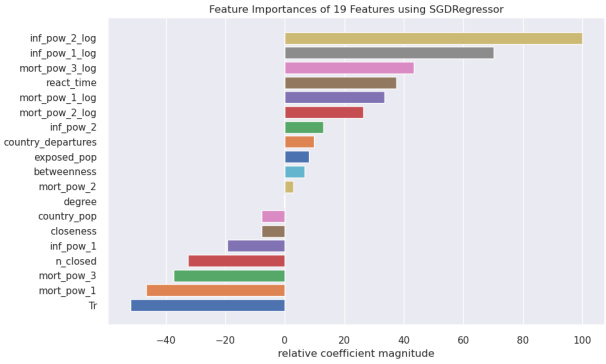
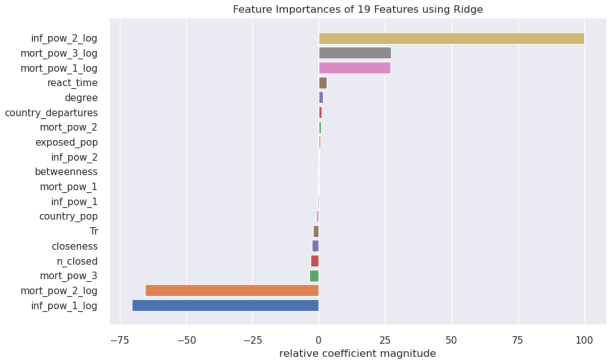


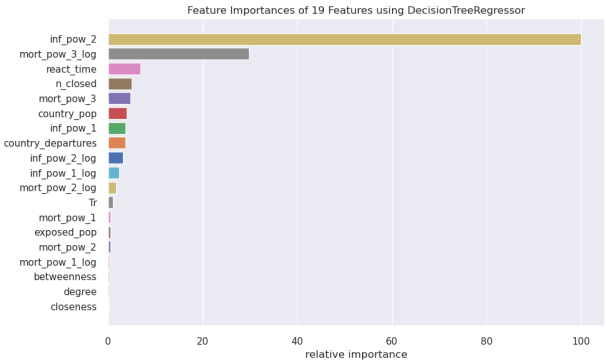
Figure 9: Correlation matrix between features.



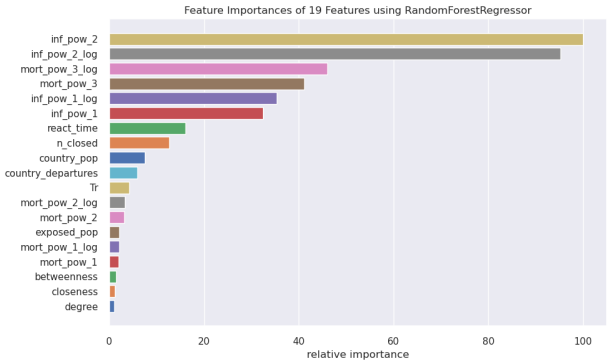
(a) SGD Regressor



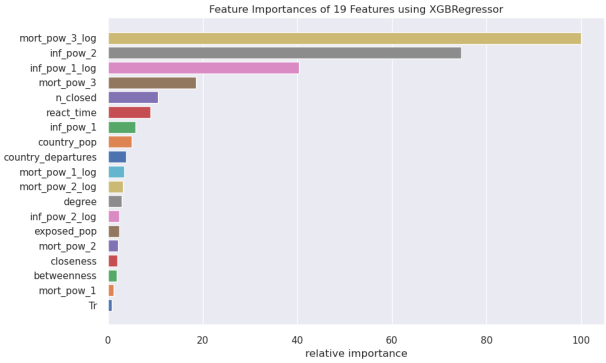
(b) Ridge Regressor



(c) Decision tree



(d) Random Forest



(e) XGBoost

Figure 10: Feature importances of different ML models.

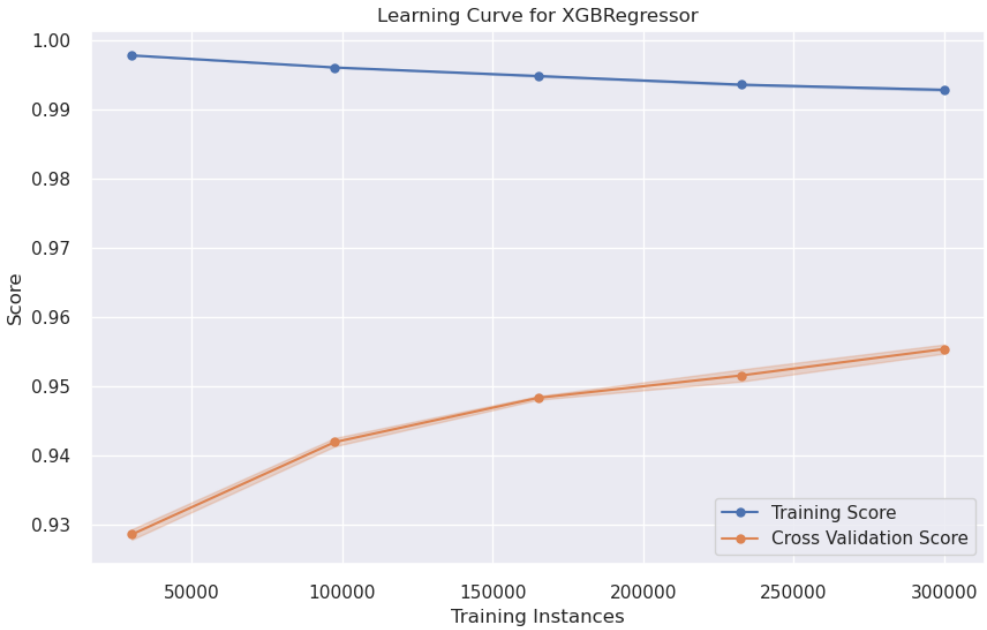


Figure 11: Learning curve of XGBoost.

Figure 12: XGBoost residuals. Test set is the validation set, results without cross-validation.

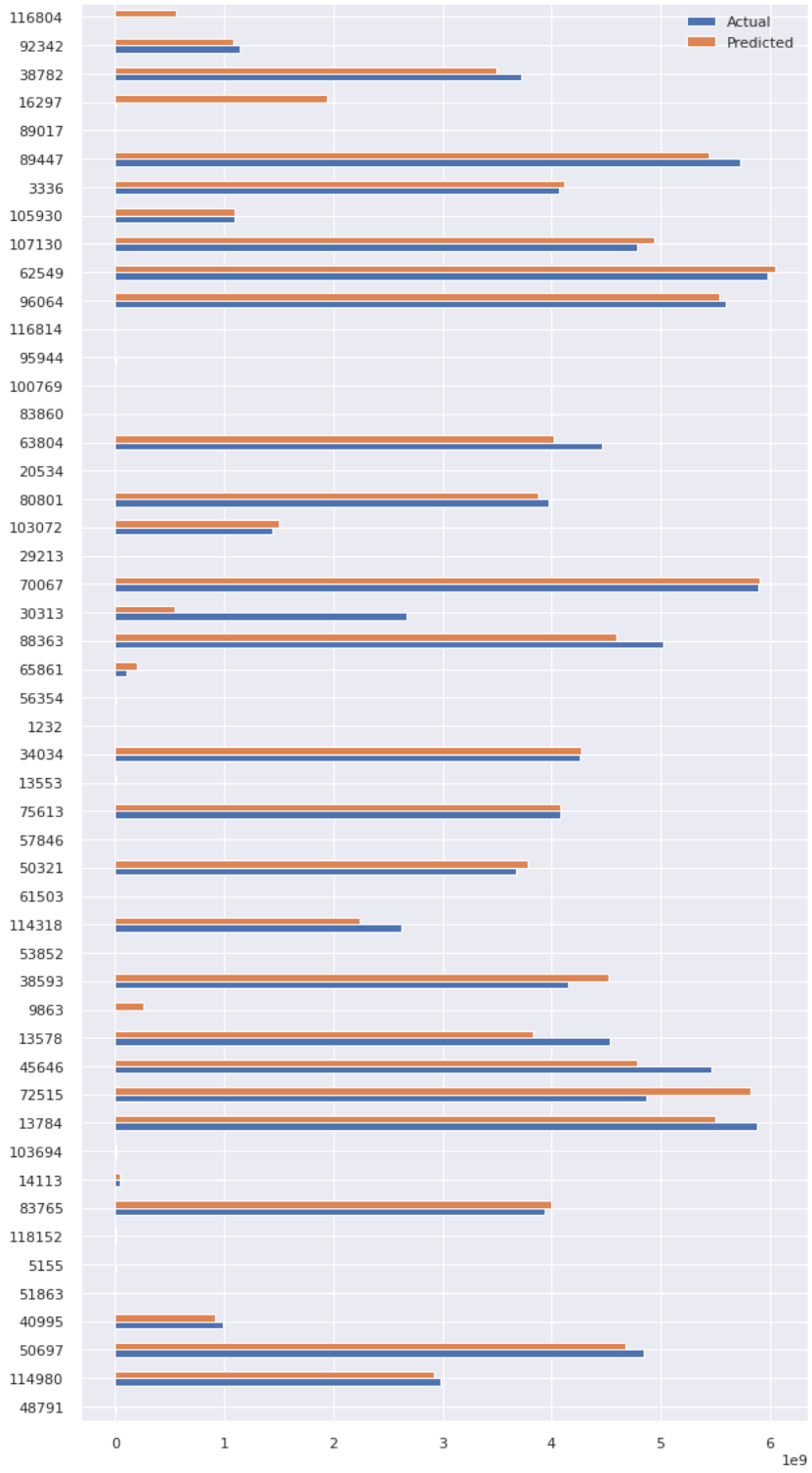
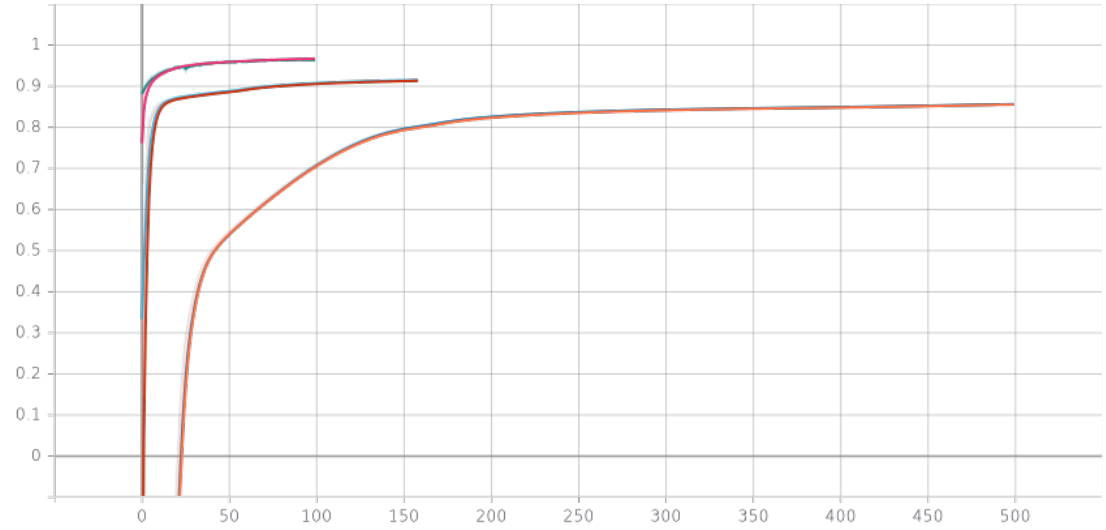
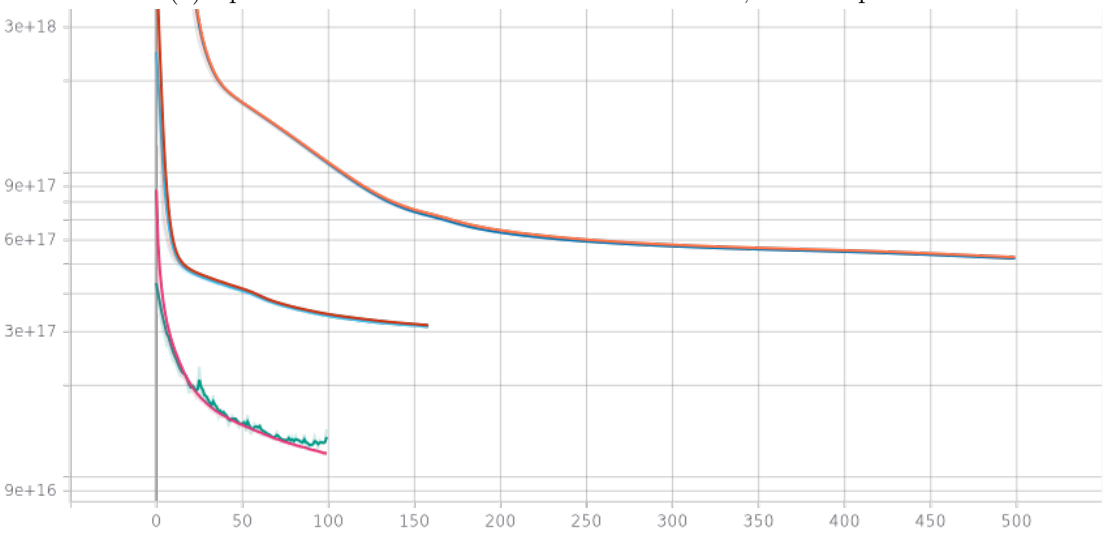


Figure 13: Sampled test predictions from XGBoost.



(a) Epoch coefficient of determination. Y-axis:R^2, x-axis: epoch



(b) Epoch loss. Y-axis:MSE, x-axis: epoch

Figure 14: Learning process of three neural networks. The one with a lower loss corresponds to a nn with 7 hidden layers and 80 neurons per hidden layer. The one with a higher loss corresponds to a nn with 3 hidden layers and 10 neurons per hidden layer. The remaining corresponds to a nn with 4 hidden layers and 40 neurons per hidden layer. For all nn the figure shows the values in train set and validation set.

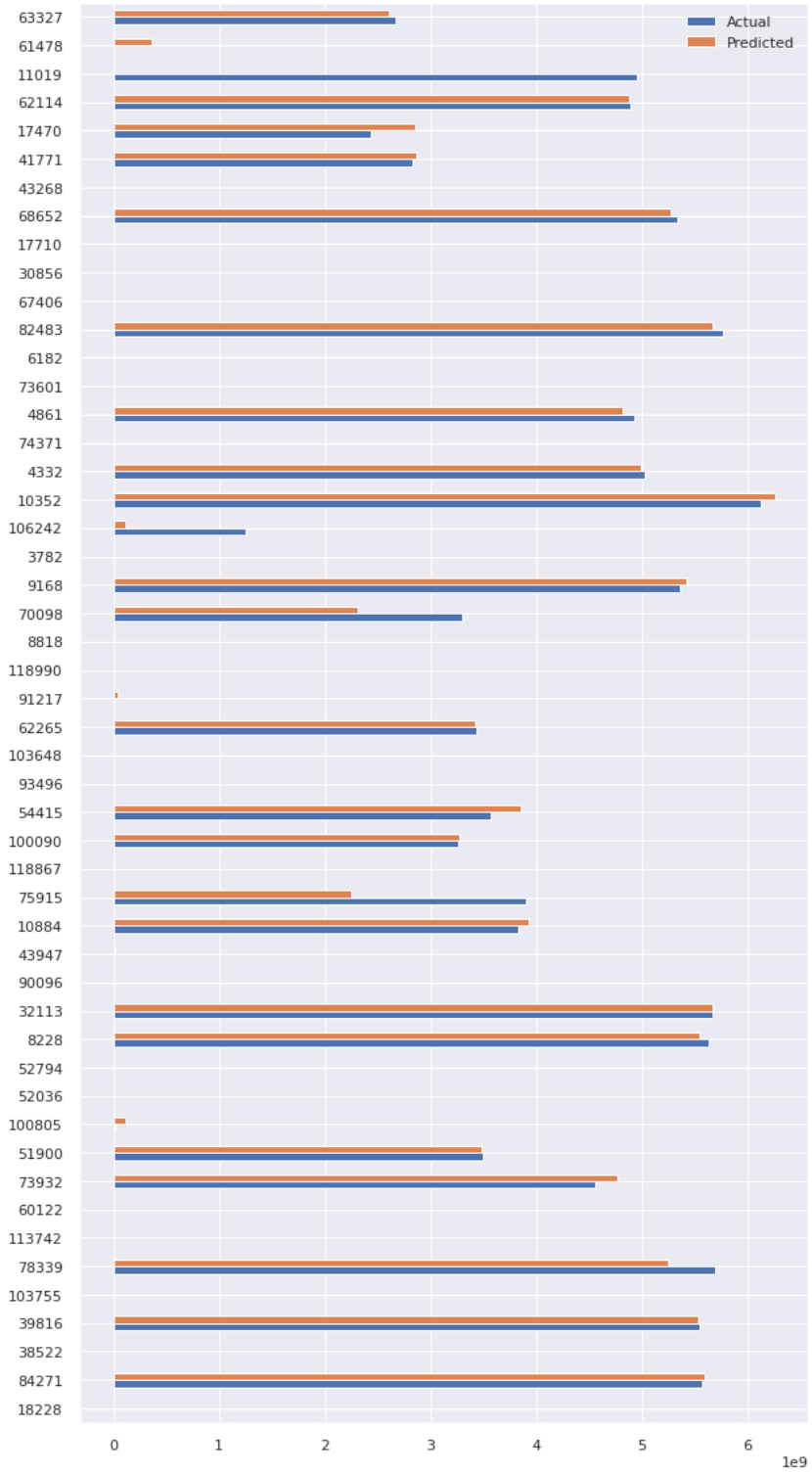


Figure 15: Sampled test predictions from a neural network.

B Code

Simulation code

```python
def simulate(self):
    """
    Begins the simulation with the parameters specified in the constructor

    Returns
    -------
    None.

    """

    self.idx_country = SIRD_model.df.loc[SIRD_model.df["country_code"]
                                         == self.i_country].index.item()
    N_i = SIRD_model.df['total_pop'].values  # Population of each country
    n = len(N_i)  # Total number of countries
    top_countries = top_k_countries(
        self.n_closed, SIRD_model.df, self.idx_country)
    # Start the SIRD matrix
    SIRD = np.zeros((n, 4))
    # Assign to the susceptible population all the population of the
    # country
    SIRD[:, 0] = N_i
    # Assign to the population of infected the initial number of infected and
    # subtracts it from the population of susceptible
    SIRD[self.idx_country, 1] += self.initial_infected
    SIRD[self.idx_country, 0] -= self.initial_infected
    SIRD_p = SIRD / SIRD.sum(axis=1).reshape(-1, 1)  # SIRD matrix normalized

    # Compute the epidemic's parameters of the SIRD model
    self.Tc = self.Tr / self.R0
    # Average number of contacts per person per time
    beta = self.Tc ** (-1)
    gamma = self.Tr ** (-1)  # Rate of recovered
    # R0 = beta / gamma
    # Make vectors from the parameters
    beta_v = np.full(n, beta)
    gamma_v = np.full(n, gamma)

    # Arrays to save the information from the simulation
    new_infected_t = np.zeros((n, SIRD_model.sim_time))
    new_recovered_t = np.zeros((n, SIRD_model.sim_time))
    new_deceased_t = np.zeros((n, SIRD_model.sim_time))
    SIRD_t = np.zeros((n, 4, SIRD_model.sim_time))

    self.OD_sim = SIRD_model.OD.copy()  # keep the original OD matrix

    flag = 1  # Flag for countries_reaction function

    # Start the simulation
    for t in range(SIRD_model.sim_time):
        # OD matrix of susceptible, infected, recovered and deceased
        S = np.array([SIRD_p[:, 0], ] * n).T
        I = np.array([SIRD_p[:, 1], ] * n).T
        R = np.array([SIRD_p[:, 2], ] * n).T
        D = np.array([SIRD_p[:, 3], ] * n).T
```

```
        # element-wise multiplication
        OD_S, OD_I, OD_R = np.floor(
            self.OD_sim * S), np.floor(self.OD_sim * I), np.floor(self.OD_sim * R)
        # People entering and leaving by group for each country
        out_S, in_S = OD_S.sum(axis=1), OD_S.sum(axis=0)
        out_I, in_I = OD_I.sum(axis=1), OD_I.sum(axis=0)
        out_R, in_R = OD_R.sum(axis=1), OD_R.sum(axis=0)
        # Update SIRD matrix according travels
        SIRD[:, 0] = SIRD[:, 0] - out_S + in_S
        SIRD[:, 1] = SIRD[:, 1] - out_I + in_I
        SIRD[:, 2] = SIRD[:, 2] - out_R + in_R
        # Cehck for negative values in SIRD
        SIRD = np.where(SIRD < 0, 0, SIRD)
        # Update population of each country
        N_i = SIRD.sum(axis=1)
        # Compute new infected at t.
        new_infected = (beta_v * SIRD[:, 0] * SIRD[:, 1]) / N_i
        # If the population N_i of a country is 0, new_infected is 0
        new_infected = np.where(np.isnan(new_infected), 0, new_infected)
        # New infected can't be higher than susceptible
        new_infected = np.where(new_infected > SIRD[:, 0], SIRD[:, 0],
                                new_infected)
        # Compute recovered at t
        new_recovered = gamma_v * SIRD[:, 1]  # New recovered at t
        # Compute deceased at t
        new_deceased = self.omega * SIRD[:, 1]
        # Updating SIRD matrix according epidemic transitions
        SIRD[:, 0] = SIRD[:, 0] - new_infected
        SIRD[:, 1] = SIRD[:, 1] + new_infected - new_recovered -new_deceased
        SIRD[:, 2] = SIRD[:, 2] + new_recovered
        SIRD[:, 3] = SIRD[:, 3] + new_deceased
        SIRD_p = SIRD / SIRD.sum(axis=1).reshape(-1, 1)
        # Checking for nan
        SIRD_p = np.where(np.isnan(SIRD_p), 0, SIRD_p)
        # Saving information of the day t
        SIRD_t[:, :, t] = np.floor(SIRD)
        new_infected_t[:, t] = np.floor(new_infected)
        new_recovered_t[:, t] = np.floor(new_recovered)
        new_deceased_t[:, t] = np.floor(new_deceased)

        if new_deceased.sum() > self.limit_deceased and flag:
            country_react, flag = countries_reaction(t, self.react_time,
                                                     top_countries)

        if not flag:
            self.OD_sim = closing_country(country_react, self.OD_sim, t)
```