Master's Degree in Data Science
Kschool

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 may be due to the lack of pandemic data since these models are based on learning from large amounts of them. This work suggests the use of a mathematical model to simulate a considerable amount of data about the spread of different diseases and use that information to feed a machine learning model. Of this way, the ML model is provided with enough power to learn and make accurate predictions.

In particular, with this aim, I create a deterministic mathematical model based on the well-known SIR model. The model is adapted to a global pandemic situation, allowing interactions between different countries and the quarantine of each one if needed. The necessary data for the mathematical model are collected from public sources and include information about countries, as total population, number of international arrivals or departures. Moreover, I use data from airports around the world to know which routes exist between countries. All of 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 based on graph theory for each country, 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. In this way, the mathematical model can simulate distinct pandemic situations. Once the data are generated, they are provided to the ML model. It should be noted that the features used in the ML model do not include any characteristic parameter of the SIR model, that cannot be calculated or are challenging to compute in a real pandemic situation. Furthermore, I use data generated in 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 makes the learning process more challenging.

Finally, after the corresponding exploratory analysis of the data simulated, 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 extreme gradient boosting that let explains the 93 % of the variability observed in the response variable and a deep neural network that is capable of explaining the 94 % 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. Moreover, 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 individuals 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. SIS model where some infections do not give long-lasting immunity. SIRD model which differentiates between recovered and deceased. 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, it proposes a solution for the lack of data, then it details the mathematical model and the methodology used to collect the data. Finally it describes the steps in the machine learning process trying to predict the response variable.

The repository with all the code, figures and steps to reproduce the results 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. Major pandemics occurred decades ago, when we still did not have the necessary means to collect the information generated methodically and thus accessible data, if they exist, are from particular 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, 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.

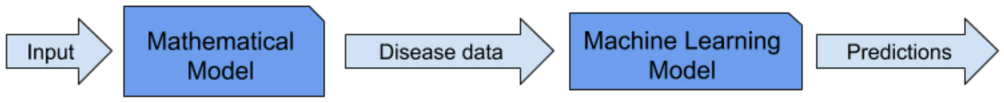


Figure 1: Data flow between models.

Since the ML model is not learning 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.

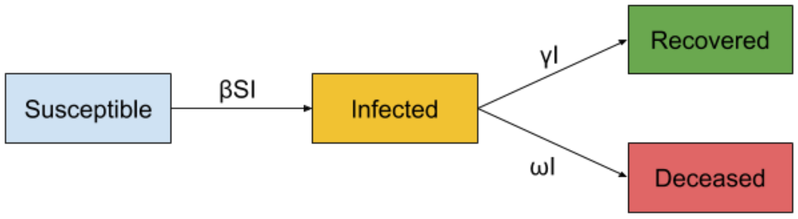


Figure 2: SIRD model transitions between compartments.

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:

$$\frac{dS}{dt} = -\frac{\beta IS}{N} \tag{1}$$
$$\frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I - \omega I \tag{2}$$
$$\frac{dR}{dt} = \gamma I \tag{3}$$
$$\frac{dD}{dt} = \omega I \tag{4}$$

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, $S(0)$, is crucial 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.

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.

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 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 the weight of this 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, there is a SIRD model for each country, and each of them has its 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 ($Reaction\, time \sim Exp(\lambda = R_t)$). It 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 a disease. We can see in the world population how the peak in both models are different, without quarantine is more pointed, has a higher value and is reached earlier. Looking at the compartments of the initial countries, we can see both are similar because once the virus begins to spread within a country, the model does not have a way to contain the disease. 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 repository. Appendix B shows an extract of the code that computes a simulation. The inputs of the model are shown in Table 2.

[1]In `TFM/src/features/sird_model_simple.py`
[2]In `TFM/src/features/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.

2.2.2 Model considerations

Obviously, there are assumptions in the model, many factors that influence a process as complex as the spread of a disease have been simplified. In this work, I consider some of them that I think are relevant. The interaction between countries (movements of individuals between populations) are simplified to travels by plane since the mobility data available correspond to information about airports and international departures and arrivals. I neglect when an individual returns to their destination of origin. Also, the model does not take into account other types of deceases that are not due to the analysed disease. Only the compartments described in Section 2 are considered and other heterogeneities of societies are excluded.

2.3 Disease data simulation

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

2.3.1 Data collection

The data collected are public available and feeds the mathematical model as shown in Figure 1. From the inputs in Table 2, it is only necessary to collect data to create OD, the origin-destination matrix, and DF, containing information about the countries. The rest of the parameters can be chosen arbitrarily by the user. 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 that make up the raw data and the information that include 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, including number of arrivals, departures and population.
- `country_population.csv`: supplementary country population data.
- `routes.csv`: data about 67 663 routes between 3 321 airports spanning the globe. Routes are directional and unique.

[3]Raw data in `TFM/data/raw`

- `airport_codes.csv`: data mapping the IATA code airport to the country to which it belongs.
- `country_to_continent.csv`: data mapping the name of the country with the continent.
- `table_convert_iso.csv`: data mapping the name of the country with the ISO 3166-1 alpha-3 and alpha-2 codes and with its location in coordinates.

2.3.2 Data transformation

Figure 3 shows 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 detail. The graph contains 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, that were in the previous step. A random forest regressor predicts the missed data instead of using the typical approach of filling with means or medians.
- `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.

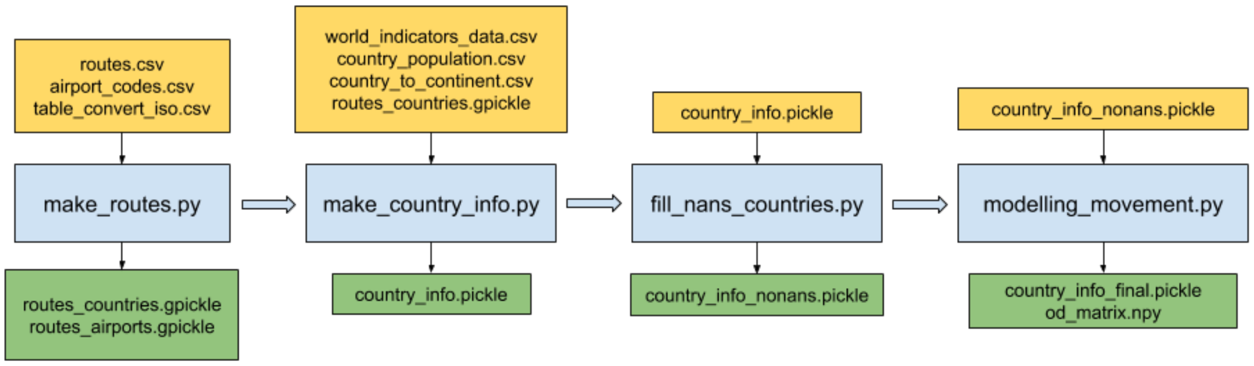


Figure 3: Data flow in the data transformation process.

Note that `country_info_final.pickle` and `od_matrix.npy` are *DF* and *OD* respectively from Table 2. Thus, all the necessary data to feed the SIRD model are ready.

[4]Data transformation scripts in `TFM/src/data`

2.3.3 Modelling movements

In order to model the movements of individuals, I define a non-symmetric constant origin-destination matrix (OD) which is determined by the information about arrivals and departures per year of each country in `world_indicators_data.csv`:[5]

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

$$\begin{aligned}
a_i &= \frac{A_i}{365} && \text{for } i = 1, 2, \ldots, N \\
d_i &= \frac{D_i}{365} && \text{for } i = 1, 2, \ldots, N \\
pa_i &= \frac{a_i}{\sum\limits_{i=1}^{N} a_i} && \text{for } i = 1, 2, \ldots, N \\
p_{ij} &= \frac{pa_j}{\sum\limits_{j \in S_i} pa_j} && \text{for } i, j = 1, 2, \ldots, N \text{ and } i \neq j \\
I_{ij} &= d_i \cdot p_{ij} && \text{for } i, j = 1, 2, \ldots, N \text{ and } i \neq j
\end{aligned}$$

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

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, I use a generator on parameters sampled from the following distributions (thinking in usual and logical 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. Figure 4 shows samples of the distributions. 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).

[5] In `TFM/src/data/modelling_movement.py`

Once the parameter space is generated, it is passed to the SIRD model, and computes over 620 000 simulations to feed the different ML models[6]. Table 3 shows a subset of outputs of the SIRD model.

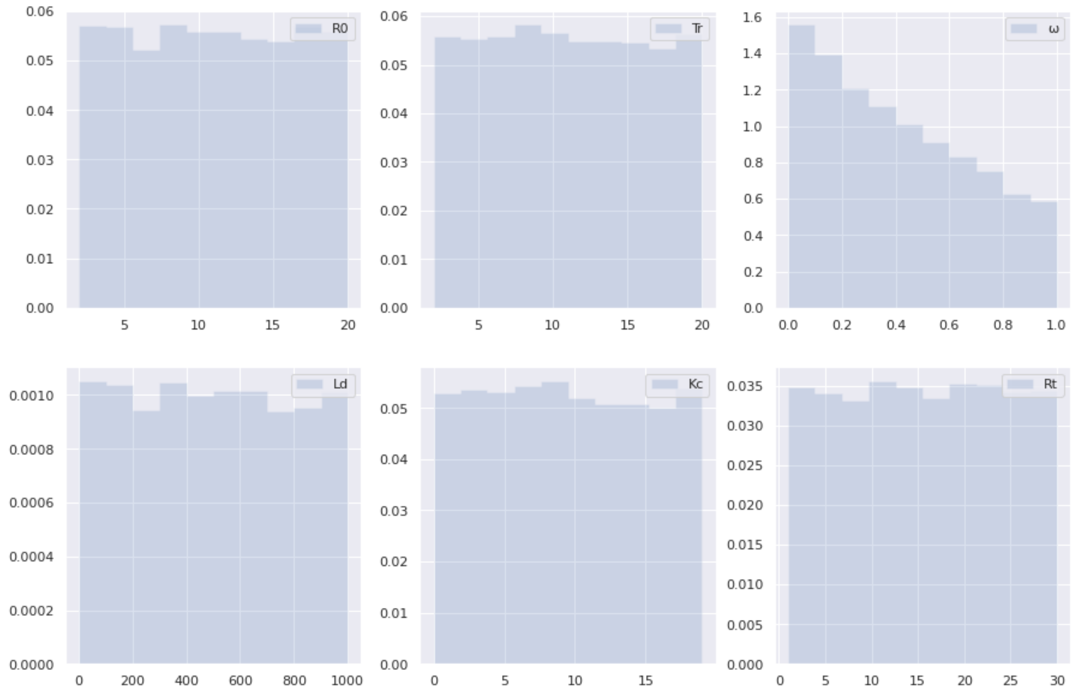


Figure 4: Distribution samples used for generating the parameter space of the SIRD model.

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.

[6]In `TFM/src/features/make_simulation.py`

2.4 Machine Learning

The goal of the machine learning model is to predict the number of deceased individuals by an arbitrary infectious disease. As mentioned, the ML models use features computed from the data generated by the SIRD model. It is crucial that the features have to be available during the first stage of the disease, 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.

The aim is a model capable of explaining the variation in the response variable. Consequently, the performance 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 response 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. Once the SIRD model simulates the data, I 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, I test a different typology of models to see which of them best suits the characteristics of the dataset and is capable of extracting the pattern that generates the response 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 again the SIRD model with parameters that the ML model has more difficulty predicting the associated target value. Furthermore, as a part of the feature engineering, I use the 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. At this step, it is essential to use as much data as possible with the best models and try to automate everything possible. Finally, once I feel confident with the final model, I measure its performance on the test set to estimate the generalisation 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

As previous mentioned, I am not using parameters R_0, T_c and *omega* to train the model. Hence it is necessary to think of other parameters capable of providing with the same information.

In this work, I assume that an infectious disease becomes known, when it causes a severe health condition in an individual or when it even die. It could be that someone is infected, but if it does not cause symptoms on her/him, the virus is not given much importance. Therefore, I consider that the information about the new disease begins to be monitored, from the first death, which is a reasonably realistic assumption. From then on, the computations of the parameters that define the power of the pandemic are carried out with data from the first two weeks after the first death, since longer would be too late to know how the disease is spreading. Hence, I define the following parameters related to the infection power (note 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

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

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

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

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 infection 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 feature $country_i$ has a large number of possible categories, specifically 204 (the number of countries in the dataset), so a one-hot encoding would result in a large number of inputs 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 useful.

One 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 greater importance within the network. The weight of a node within a graph can be measured with different parameters, some of them are:

[7]Disease features in `TFM/src/utils/sird_support.py`

- **Degree**. It is the number of edges that are incident to the node. Meaning that important nodes have many connections.
- **Betweenness**. It is a measure of centrality in a graph based on shortest paths. Meaning that important nodes connect other nodes.
- **Closeness**. It is also a measure of centrality of a node in a network. Meaning that important nodes are close to other nodes.

I compute the three parameters for each country and are added as features. For more information about them and how they are calculated check [8]. Finally, I add other parameters related to the initial country:

- 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 used in the ML models 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 $degree$, $betweenness$, $closeness$, $populationcountry_i$, $departurescountry_i$ and $exposed\,population$[8].

[8]Spread and quarantine features in `TFM/src/features/add_features.py`

3 Analysis

This section presents analyses and processes that have been significant in obtaining the final model. First it describes the exploratory analysis of the data generated by the SIRD model. Next, it explores different models and finally the best models are fine-tuned to achieve the final solution. From now on the code notation is used to name the variables.

3.1 Data Analysis

Once the response variable is defined, it is essential to explore its distribution. Figure 5 shows that the target is a bit unbalanced, sometimes it reaches 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 are relatively high values). After, it is necessary to explore the features 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, at this point, I plot the response variable against the rest of the features to see if I observe some non-linear pattern. From the graphics, it seems that a logarithmic transformation can be promising.

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, I get a substantial 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. Figure 9 shows the all the correlations between features.

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 ML model, in particular, the decision about what features are useful to predict the target.

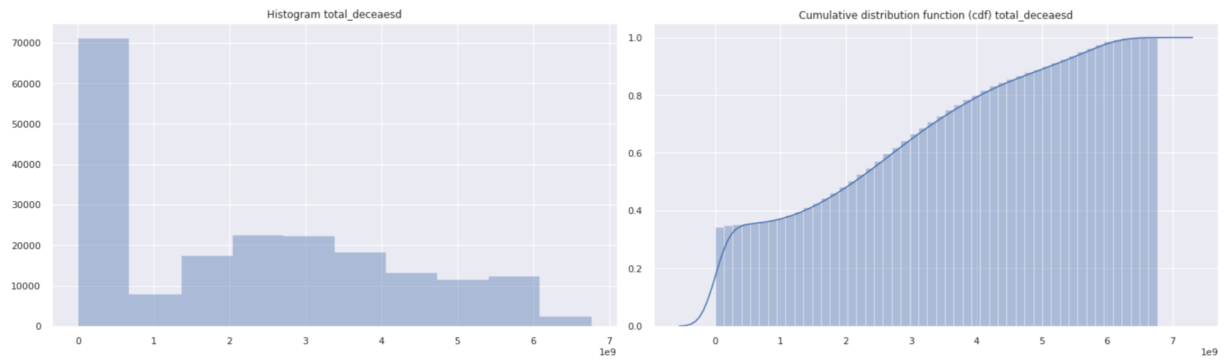


Figure 5: Target distribution (histogram and cdf).

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 coefficient of determination 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. However they perform better than a Dummy Regressor which always predicts the mean of the target value ($R^2 = 0$). The exact values besides the standard deviations get in the cross-validation can be seen in Table 4.

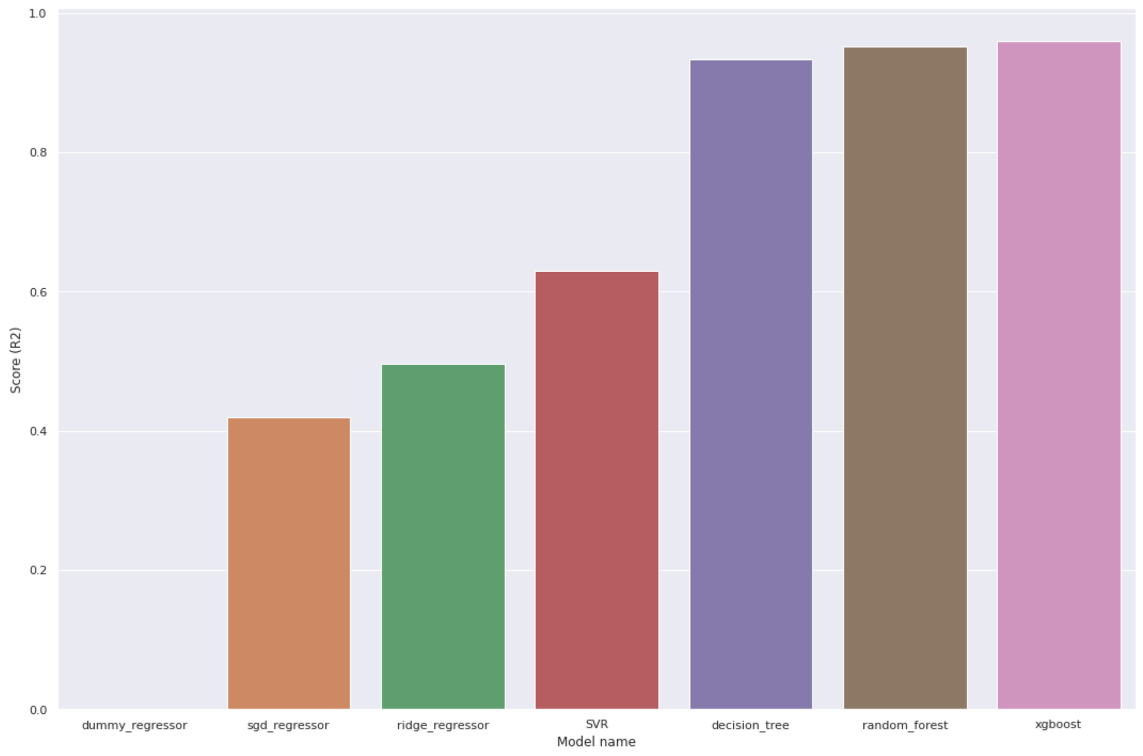


Figure 6: Scores of different models, cross-validated.

From the results, I conclude that tree-based models outperform linear regressions and support vector regressor. The reason may be 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 also depends on other factors such as the connectivity in the network of the initial country or whether quarantine occurs. On the other hand, tree-based models can learn non-linear relationships easily, and they

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

Model	R^2	SD
Dummy Regressor	6.92e−6	4.33e−6
SGD Regressor	0.420	3.29e−2
Ridge Regressor	0.496	3.11e−4
SVR	$0,630$	2.17e−2
Decision Tree	0.934	4.24e−4
Random Forest	0.951	1.17e−3
XGboost	0.959	1.24e−3

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

divide better the hyperspace according to the input parameters.

Figure 10 shows the feature importance of 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` have similar weight. Last, of the features related to the connectivity of the initial country, Xgboost uses `country_departures` and `degree` most.

3.2.1 XGBoost

The previous section shows that excluding Neural Networks, XGBoost is the model which get the best results. 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 shown in Figure 11, we can see how if we continue feeding 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 in Figure 12, we can see how the model predicts above the real value when the target is low, even though the distribution of the residuals is approximately centred at 0.

Finally, I measure the performance of the model with the test set we set aside in Section 2.4. The model scores $R^2 = 0.926$, $RMSE = 6.08\text{e+}8$ and $MAE = 2.16\text{e+}8$. 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, I train a neural network with the dataset. This trendy model allows to tune a lot of parameters, so the process is not trivial. Some parameters 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 them; therefore, a systematic procedure is needed to explore the most.

Hence, I test distinct deep neural networks with combinations of different hidden layers and number of neurons per hidden layer, adding a final layer with an individual neuron as we are in a regression supervised learning task. Figure 14 shows the search space from which the hyperparameters are randomly selected. The neural network architecture is composed by LeCun initialization as a kernel initializer, SELU as activation functions, early stopping as regularization method and Nadam as the optimizer. Nadam is prefered because it is composed by Adam optimizer, which is an adaptative learning 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 allow the network to self-normalize, meaning the output of each layer will tend to preserve mean zero and standard deviation one during training, which solves the vanishing/exploding gradients problem. Without forgetting that the input features must be standardized and the network's architecture must be sequential [11].

Figure 16 shows the coefficient of determination and loss (MSE) of different architectures of neural networks trained. The best architecture, 9 hidden layers with 70 neurons per layer shown in Figure 15, achieved a $R^2 = 0.972$ in the validation set. Figure 17 shows its score and loss in train and validation sets. Finally, the performance in the test set is: $R^2 = 0.936$, $RMSE = 5.64\text{e+}8$ and $MAE = 1.73\text{e+}8$. In Figure 18 can be seen 50 predictions samples from the test set.

4 Conclusions

In conclusion, it has been possible to create a mathematical model capable of simulating, with certain simplifications, the spread of disease among the different countries of the world. Additionally, a ML model has been trained with the data simulated, using features subject to restrictions presented during the work, to predict the number of deceased individuals in a pandemic and being able to generalize to new unknown diseases. I tested different ML models and 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 93 % of the variability observed in the response variable, moreover, a neural network described in Section 3.2.2, is capable of explaining the 94 % of the variability. The results from the tree-based models suggest that by feeding these models with more data, the coefficient of determination may continue increasing slightly, nevertheless, the computations cost that this entails must be taken into account and assess the performance with which one can be satisfied. It would be interesting to study to what extent these models continue learning and if there comes the point where the loss in the validation set begins to increase, indicating that they are overfitting the training set.

The limitations of the ML model, 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 suits the data set. On this basis, 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 to increase the complexity of the mathematical model, so it would be able to represent the spread of a pandemic more reliably. Moreover, it would be useful to increase the number of simulations to produce more data to feed the tree-based models, since previously mentioned, data suggest the performance could increase. Finally, efforts 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.

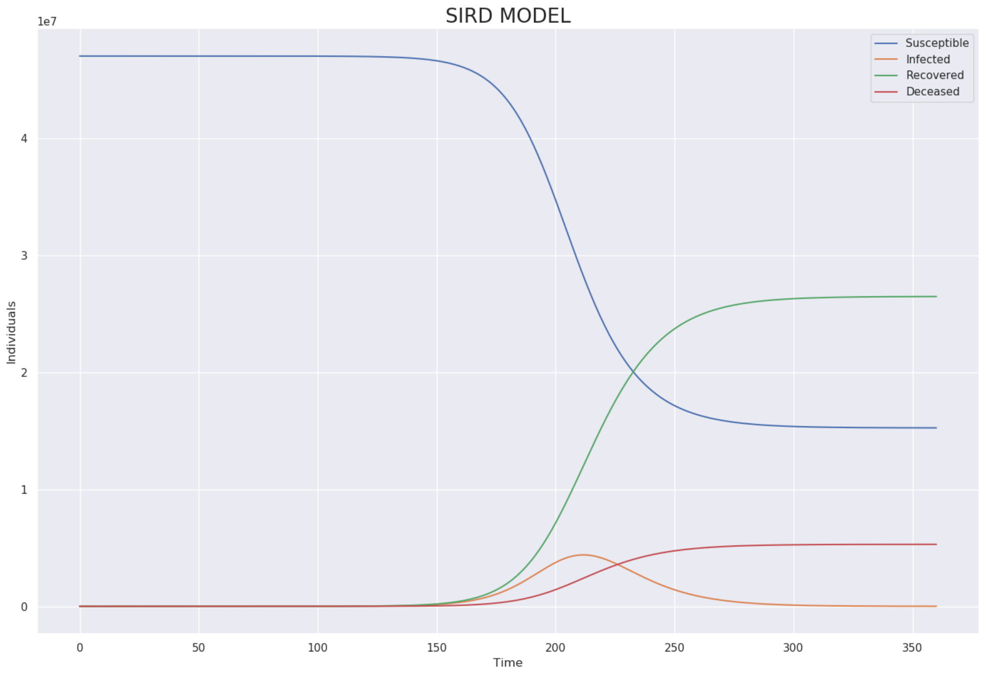
[8] Florian Boudin. A comparison of centrality measures for graph-based keyphrase extraction. In *Proceedings of the sixth international joint conference on natural language processing*, pages 834–838, 2013.

[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.

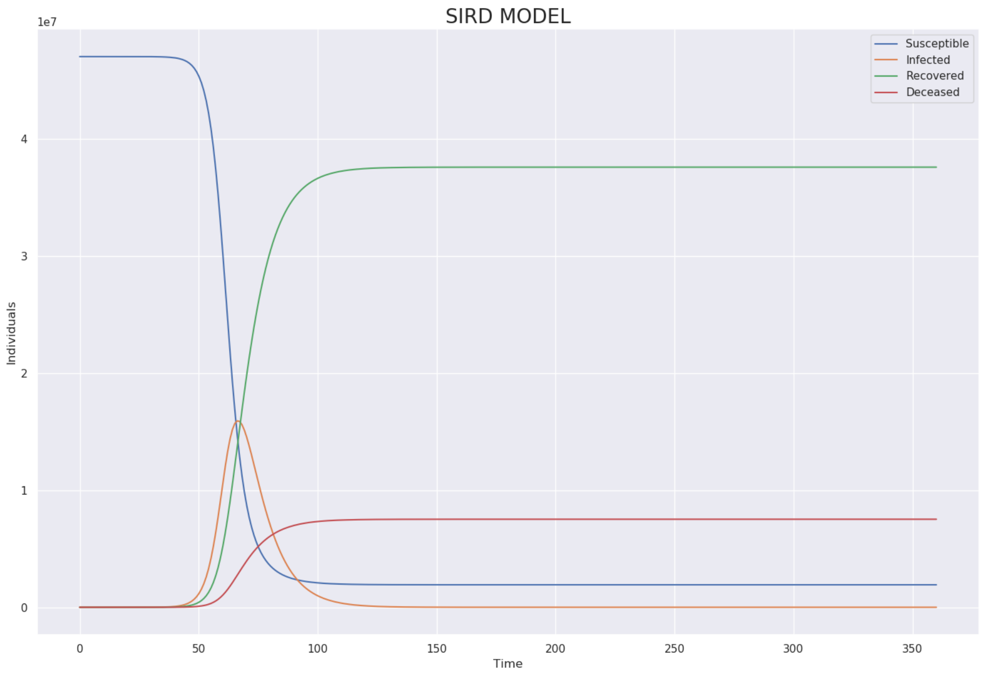
[10] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[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

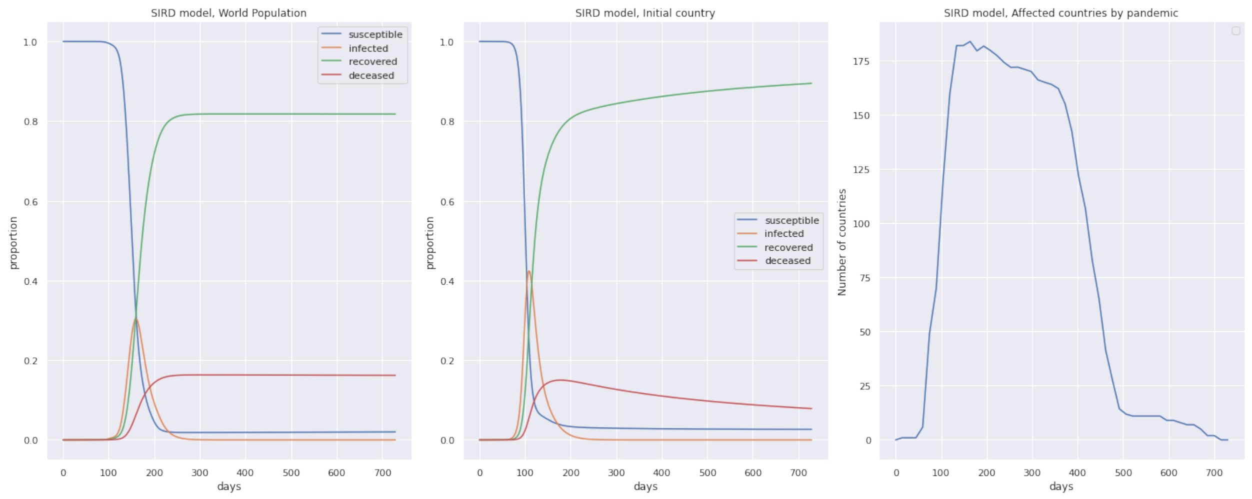


(a) $R_0 = 2$

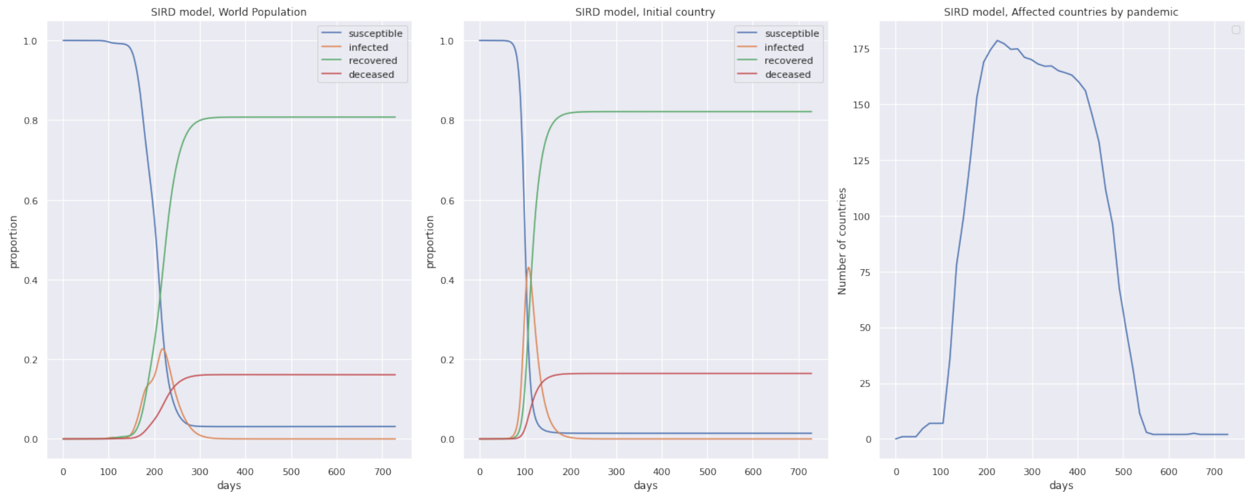


(b) $R_0 = 4$

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 $T = 360$

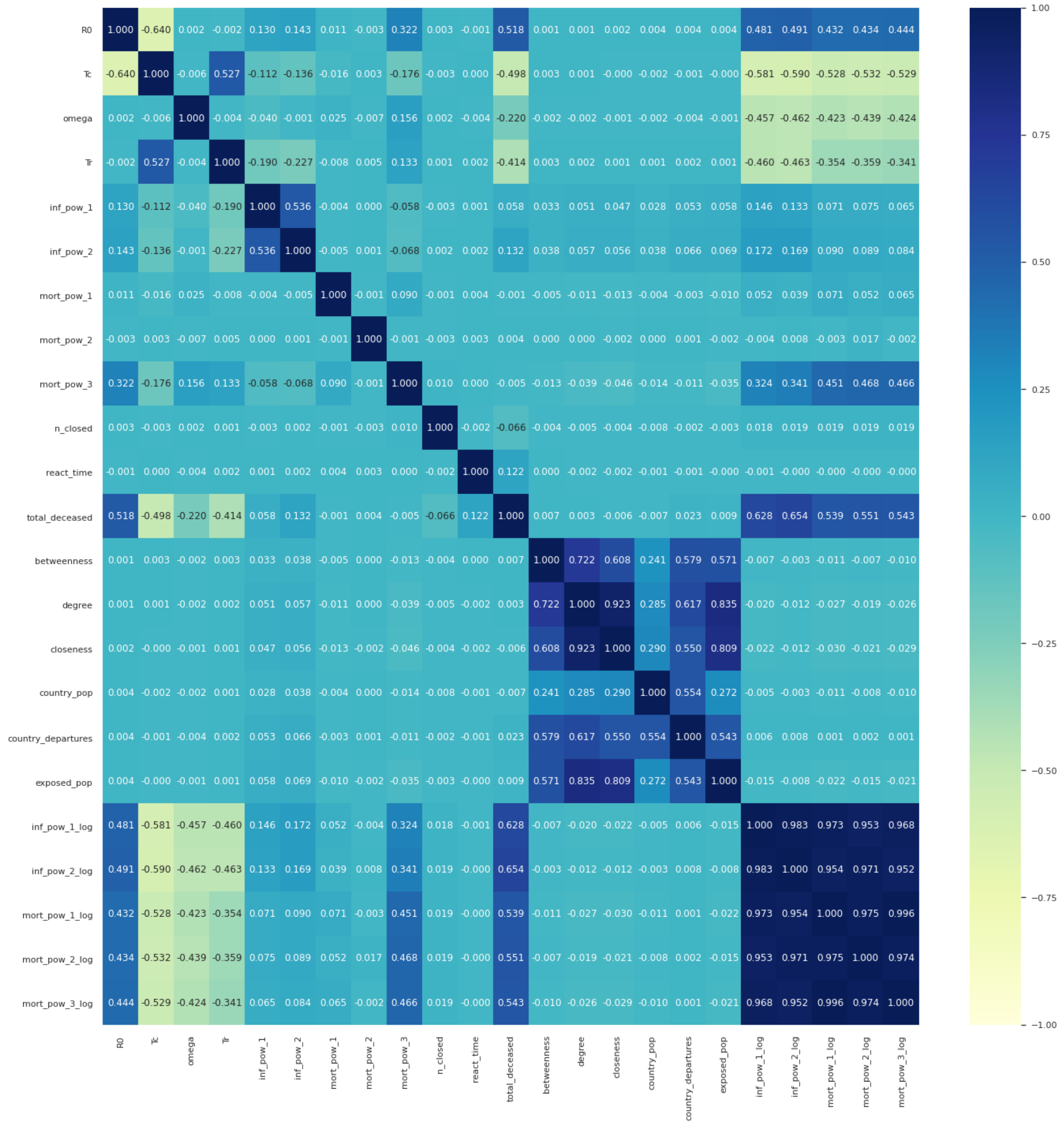


(a) No quarantine



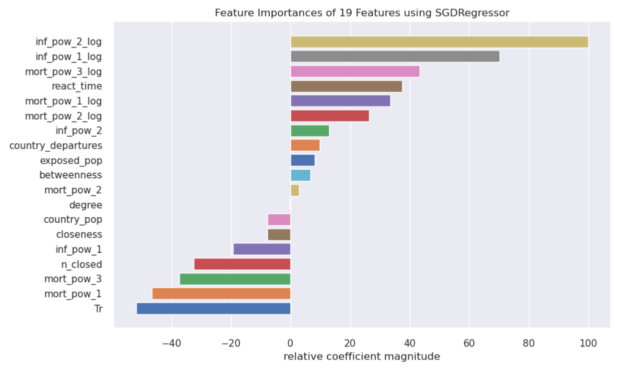
(b) With quarantine, $K_c = 5$ and $R_t = 5$

Figure 8: Difference between two SIRD models, one with quarantine and other without. The first plot represents the progression of the disease in the world population, the second plot represents the progression of the disease in the initial country.Finally, the last plot 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, $I_0 = 1$.

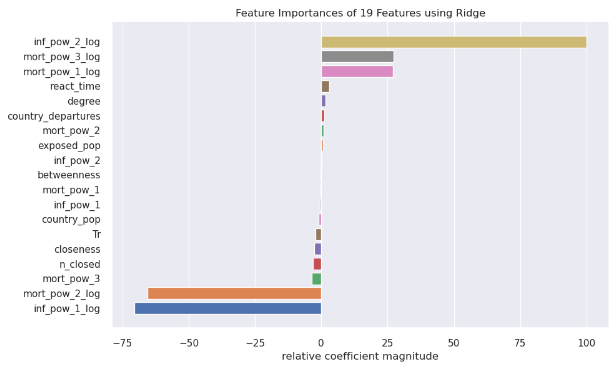


	R0	Tc	omega	Tr	inf_pow_1	inf_pow_2	mort_pow_1	mort_pow_2	mort_pow_3	n_closed	react_time	total_deceased	betweenness	degree	closeness	country_pop	country_departures	exposed_pop	inf_pow_1_log	inf_pow_2_log	mort_pow_1_log	mort_pow_2_log	mort_pow_3_log
R0	1.000	-0.640	0.002	-0.002	0.130	0.143	0.011	-0.003	0.322	0.003	-0.001	0.518	0.001	0.001	0.002	0.004	0.004	0.004	0.481	0.491	0.432	0.434	0.444
Tc	-0.640	1.000	-0.006	0.527	-0.112	-0.136	-0.016	0.003	-0.176	-0.003	0.000	-0.498	0.003	0.001	-0.000	-0.002	-0.001	-0.000	-0.581	-0.590	-0.528	-0.532	-0.529
omega	0.002	-0.006	1.000	-0.004	-0.040	-0.001	0.025	-0.007	0.156	0.002	-0.004	-0.220	-0.002	-0.002	-0.001	-0.002	-0.004	-0.001	-0.457	-0.462	-0.423	-0.439	-0.424
Tr	-0.002	0.527	-0.004	1.000	-0.190	-0.227	-0.008	0.005	0.133	0.001	0.002	-0.414	0.003	0.002	0.001	0.001	0.002	0.001	-0.460	-0.463	-0.354	-0.359	-0.341
inf_pow_1	0.130	-0.112	-0.040	-0.190	1.000	0.536	-0.004	0.000	-0.058	-0.003	0.001	0.058	0.033	0.051	0.047	0.028	0.053	0.058	0.146	0.133	0.071	0.075	0.065
inf_pow_2	0.143	-0.136	-0.001	-0.227	0.536	1.000	-0.005	0.001	-0.068	0.002	0.002	0.132	0.038	0.057	0.056	0.038	0.066	0.069	0.172	0.169	0.090	0.089	0.084
mort_pow_1	0.011	-0.016	0.025	-0.008	-0.004	-0.005	1.000	-0.001	0.090	-0.001	0.004	-0.001	-0.005	-0.011	-0.013	-0.004	-0.003	-0.010	0.052	0.039	0.071	0.052	0.065
mort_pow_2	-0.003	0.003	-0.007	0.005	0.000	0.001	-0.001	1.000	-0.001	-0.003	0.003	0.004	0.000	0.000	-0.002	0.000	0.001	-0.002	-0.004	0.008	-0.003	0.017	-0.002
mort_pow_3	0.322	-0.176	0.156	0.133	-0.058	-0.068	0.090	-0.001	1.000	0.010	0.000	-0.005	-0.013	-0.039	-0.046	-0.014	-0.011	-0.035	0.324	0.341	0.451	0.468	0.466
n_closed	0.003	-0.003	0.002	0.001	-0.003	0.002	-0.001	-0.003	0.010	1.000	-0.002	-0.066	-0.004	-0.005	-0.004	-0.008	-0.002	-0.003	0.018	0.019	0.019	0.019	0.019
react_time	-0.001	0.000	-0.004	0.002	0.001	0.002	0.004	0.003	0.000	-0.002	1.000	0.122	0.000	-0.002	-0.002	-0.001	-0.001	-0.000	-0.001	-0.000	-0.000	-0.000	-0.000
total_deceased	0.518	-0.498	-0.220	-0.414	0.058	0.132	-0.001	0.004	-0.005	-0.066	0.122	1.000	0.007	0.003	-0.006	-0.007	0.023	0.009	0.628	0.654	0.539	0.551	0.543
betweenness	0.001	0.003	-0.002	0.003	0.033	0.038	-0.005	0.000	-0.013	-0.004	0.000	0.007	1.000	0.722	0.608	0.241	0.579	0.571	-0.007	-0.003	-0.011	-0.007	-0.010
degree	0.001	0.001	-0.002	0.002	0.051	0.057	-0.011	0.000	-0.039	-0.005	-0.002	0.003	0.722	1.000	0.923	0.285	0.617	0.835	-0.020	-0.012	-0.027	-0.019	-0.026
closeness	0.002	-0.000	-0.001	0.001	0.047	0.056	-0.013	-0.002	-0.046	-0.004	-0.002	-0.006	0.608	0.923	1.000	0.290	0.550	0.809	-0.022	-0.012	-0.030	-0.021	-0.029
country_pop	0.004	-0.002	-0.002	0.001	0.028	0.038	-0.004	0.000	-0.014	-0.008	-0.001	-0.007	0.241	0.285	0.290	1.000	0.554	0.272	-0.005	-0.003	-0.011	-0.008	-0.010
country_departures	0.004	-0.001	-0.004	0.002	0.053	0.066	-0.003	0.001	-0.011	-0.002	-0.001	0.023	0.579	0.617	0.550	0.554	1.000	0.543	0.006	0.008	0.001	0.002	0.001
exposed_pop	0.004	-0.000	-0.001	0.001	0.058	0.069	-0.010	-0.002	-0.035	-0.003	-0.000	0.009	0.571	0.835	0.809	0.272	0.543	1.000	-0.015	-0.008	-0.022	-0.015	-0.021
inf_pow_1_log	0.481	-0.581	-0.457	-0.460	0.146	0.172	0.052	-0.004	0.324	0.018	-0.001	0.628	-0.007	-0.020	-0.022	-0.005	0.006	-0.015	1.000	0.983	0.973	0.953	0.968
inf_pow_2_log	0.491	-0.590	-0.462	-0.463	0.133	0.169	0.039	0.008	0.341	0.019	-0.000	0.654	-0.003	-0.012	-0.012	-0.003	0.008	-0.008	0.983	1.000	0.954	0.971	0.952
mort_pow_1_log	0.432	-0.528	-0.423	-0.354	0.071	0.090	0.071	-0.003	0.451	0.019	-0.000	0.539	-0.011	-0.027	-0.030	-0.011	0.001	-0.022	0.973	0.954	1.000	0.975	0.996
mort_pow_2_log	0.434	-0.532	-0.439	-0.359	0.075	0.089	0.052	0.017	0.468	0.019	-0.000	0.551	-0.007	-0.019	-0.021	-0.008	0.002	-0.015	0.953	0.971	0.975	1.000	0.974
mort_pow_3_log	0.444	-0.529	-0.424	-0.341	0.065	0.084	0.065	-0.002	0.466	0.019	-0.000	0.543	-0.010	-0.026	-0.029	-0.010	0.001	-0.021	0.968	0.952	0.996	0.974	1.000

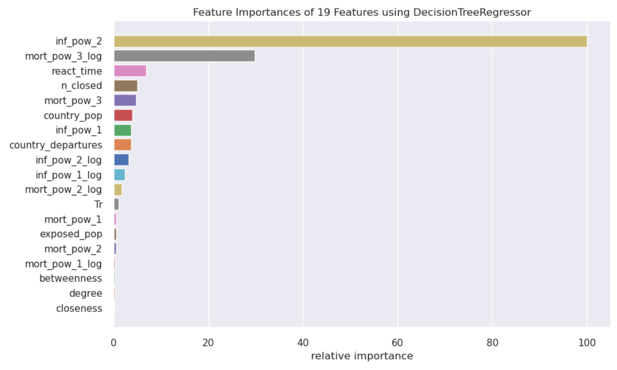
Figure 9: Correlation matrix between features.



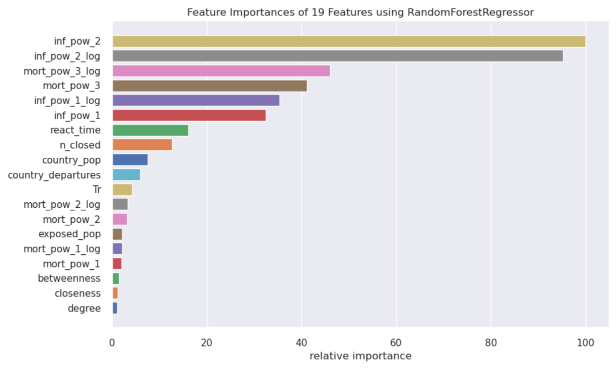
(a) SGD Regressor



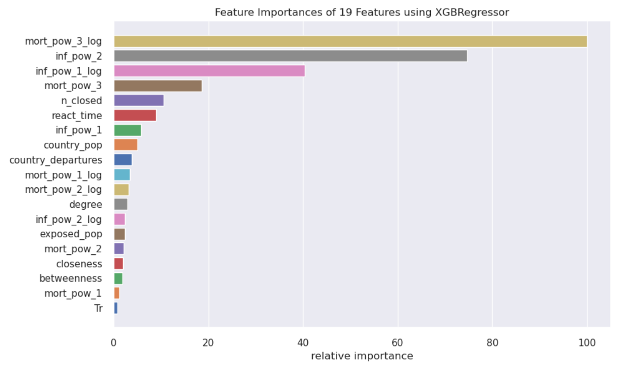
(b) Ridge Regressor



(c) Decision tree



(d) Random Forest



(e) XGBoost

Figure 10: Feature importance of different ML models.

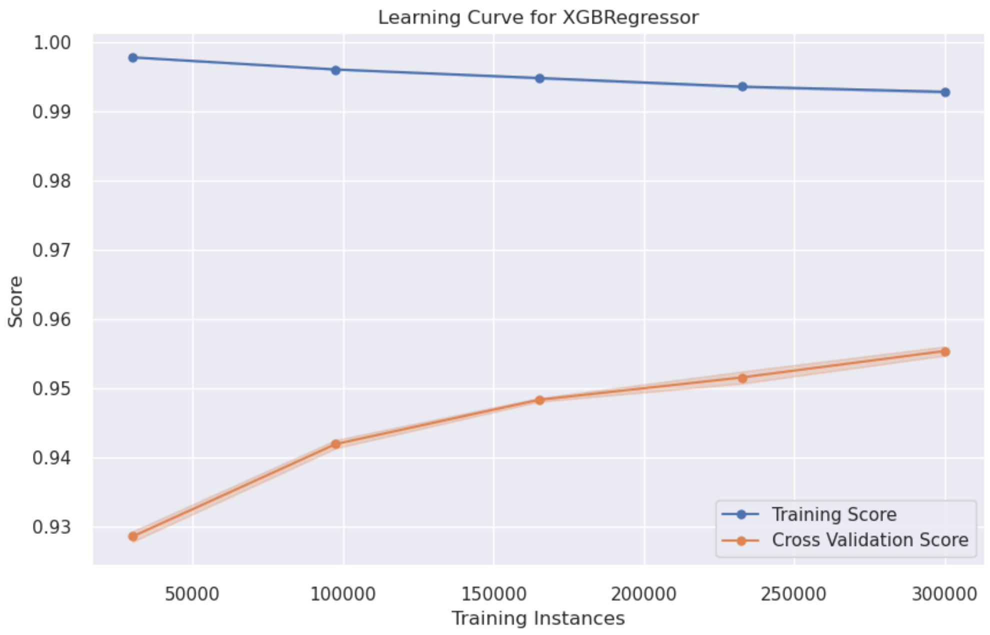


Figure 11: Learning curve of XGBoost.

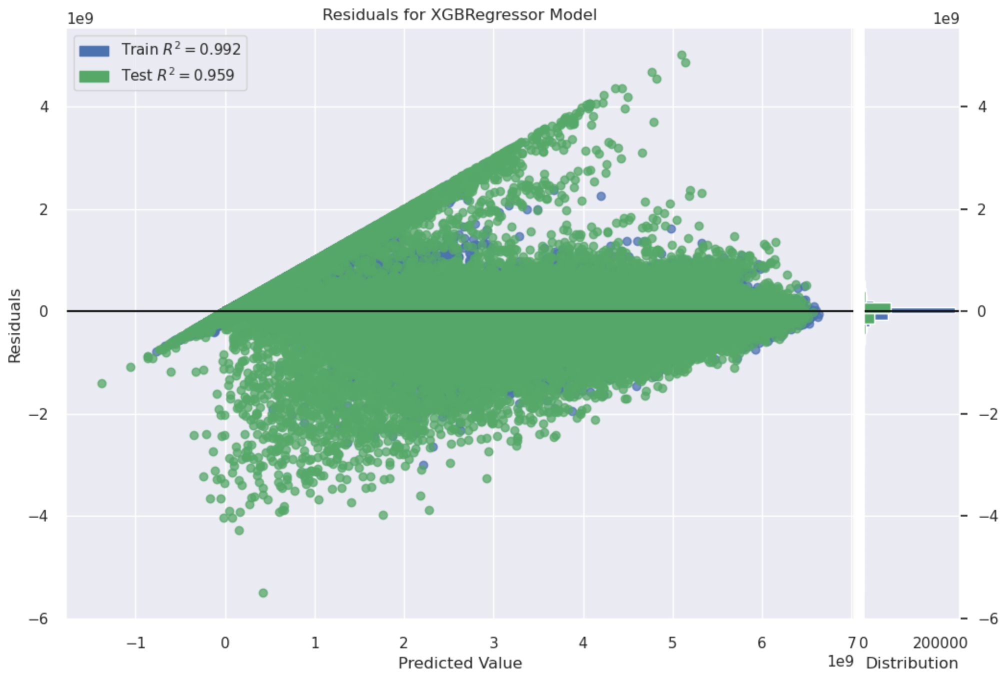


Figure 12: XGBoost residuals. Test set is a validation set.

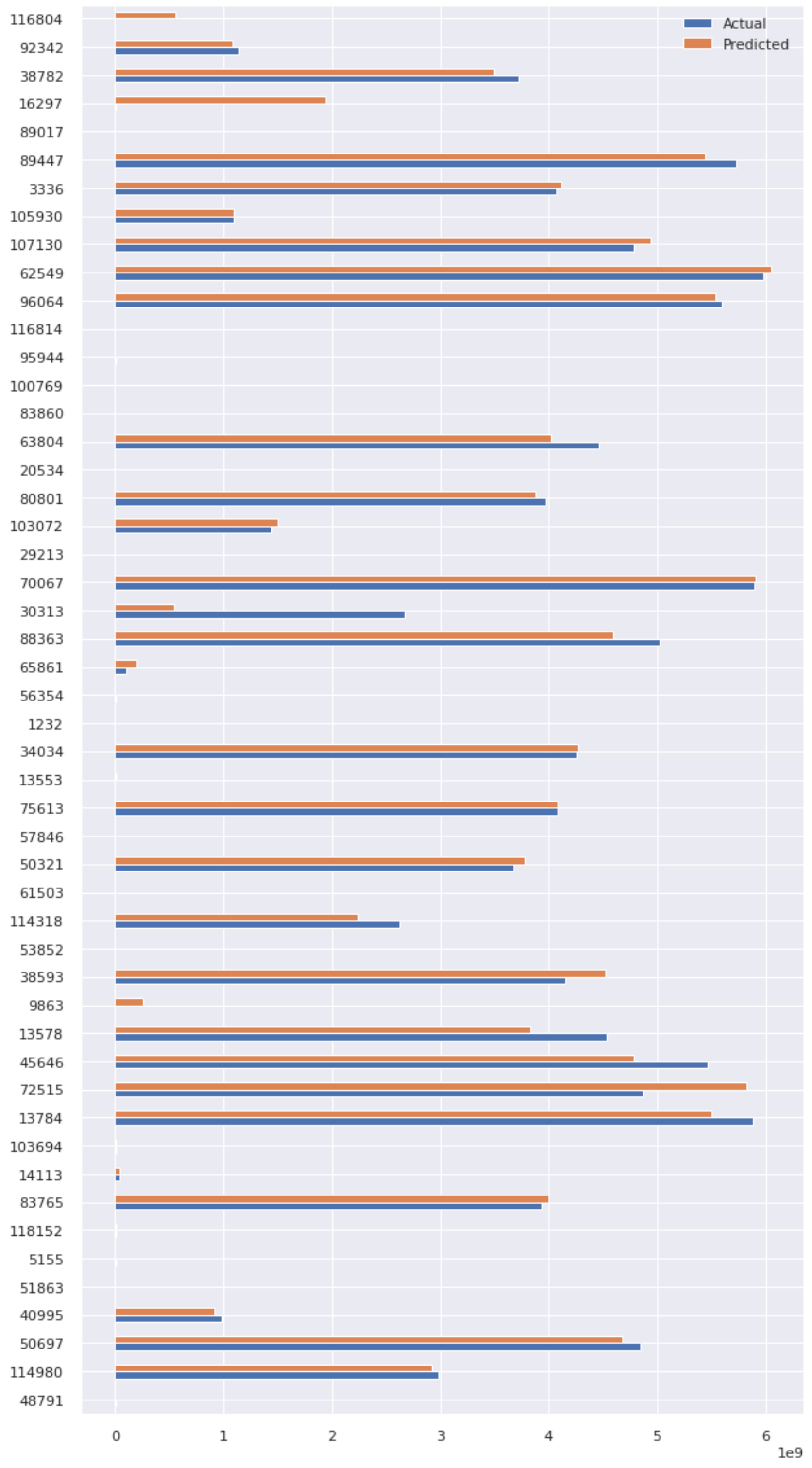
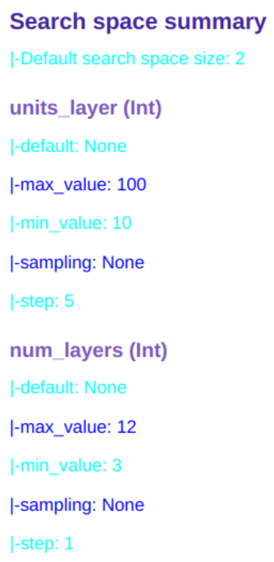
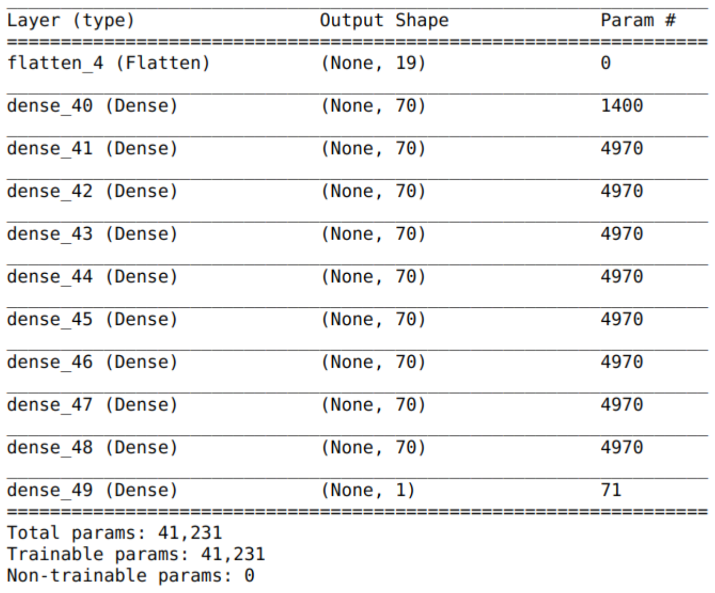


Figure 13: Predictions sampled from test set, XGBoost.



```
Search space summary
|-Default search space size: 2
units_layer (Int)
|-default: None
|-max_value: 100
|-min_value: 10
|-sampling: None
|-step: 5
num_layers (Int)
|-default: None
|-max_value: 12
|-min_value: 3
|-sampling: None
|-step: 1
```

Figure 14: Search space, neural network hyperparameters.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_4 (Flatten)          (None, 19)                0
_________________________________________________________________
dense_40 (Dense)             (None, 70)                1400
_________________________________________________________________
dense_41 (Dense)             (None, 70)                4970
_________________________________________________________________
dense_42 (Dense)             (None, 70)                4970
_________________________________________________________________
dense_43 (Dense)             (None, 70)                4970
_________________________________________________________________
dense_44 (Dense)             (None, 70)                4970
_________________________________________________________________
dense_45 (Dense)             (None, 70)                4970
_________________________________________________________________
dense_46 (Dense)             (None, 70)                4970
_________________________________________________________________
dense_47 (Dense)             (None, 70)                4970
_________________________________________________________________
dense_48 (Dense)             (None, 70)                4970
_________________________________________________________________
dense_49 (Dense)             (None, 1)                 71
=================================================================
Total params: 41,231
Trainable params: 41,231
Non-trainable params: 0
```

Figure 15: Best neural network architecture.

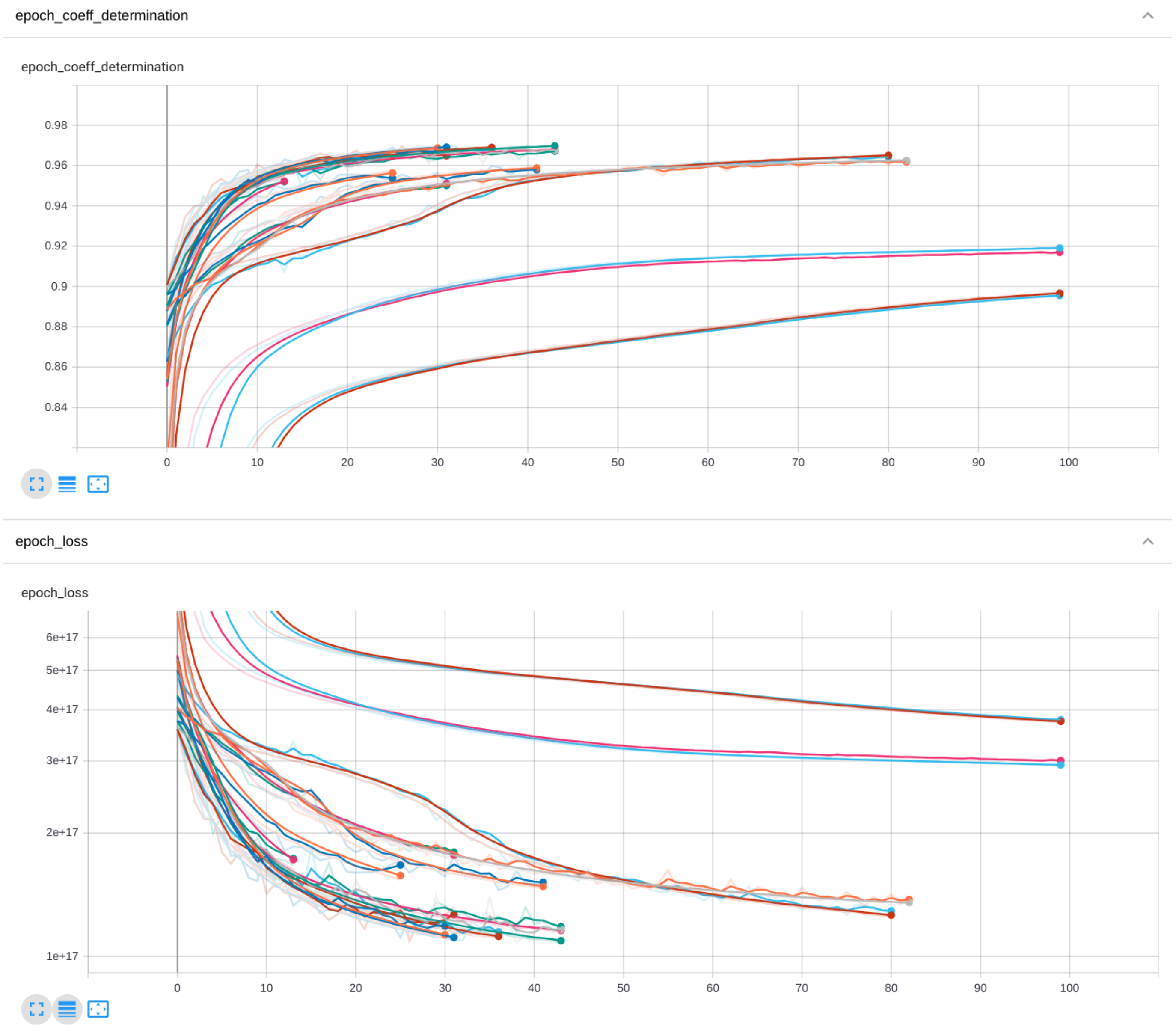


Figure 16: Coefficient of determination and loss (MSE) of different neural networks architectures.

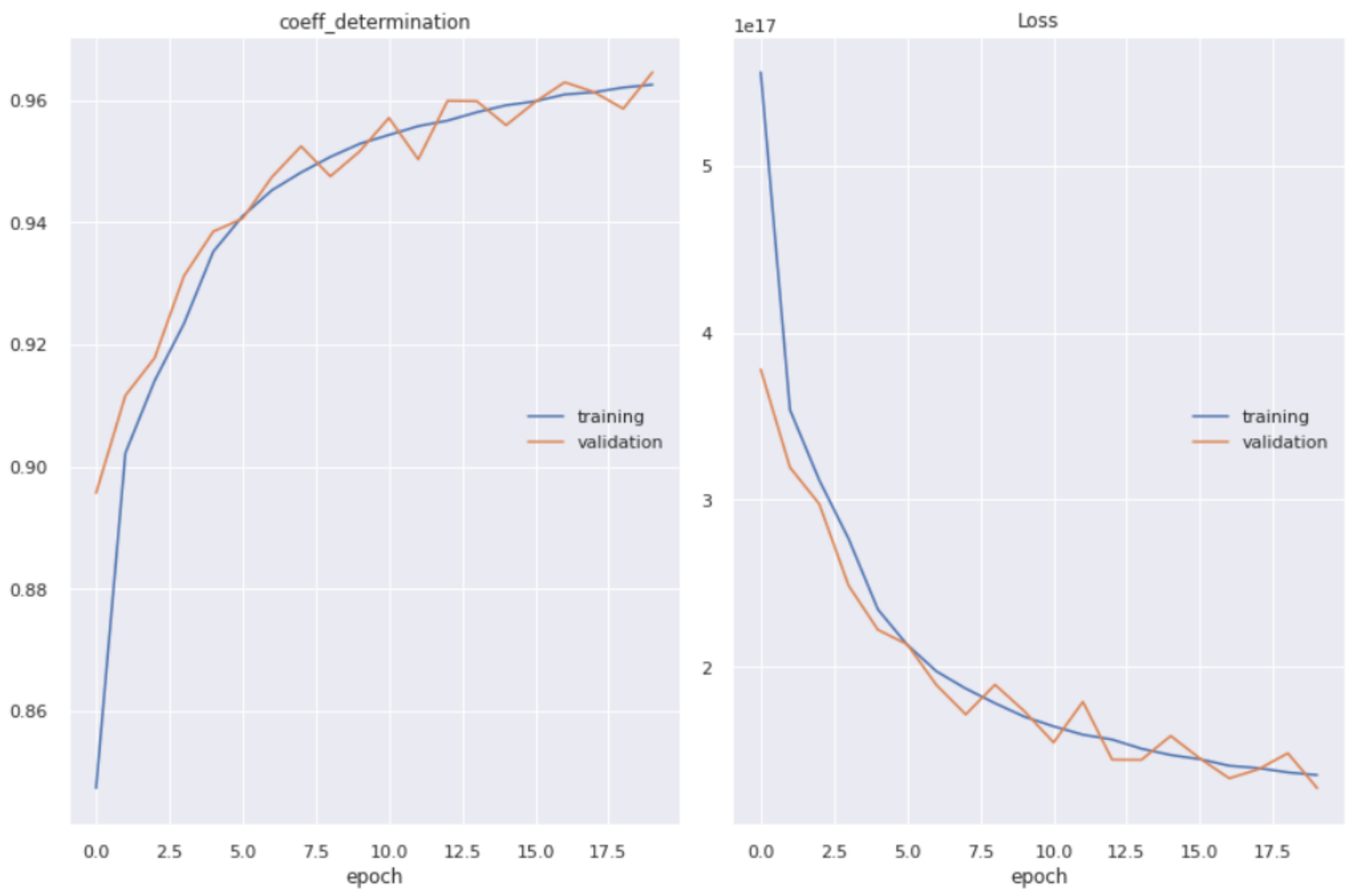


Figure 17: Coefficient of determination and loss (MSE) of the best neural network architecture.

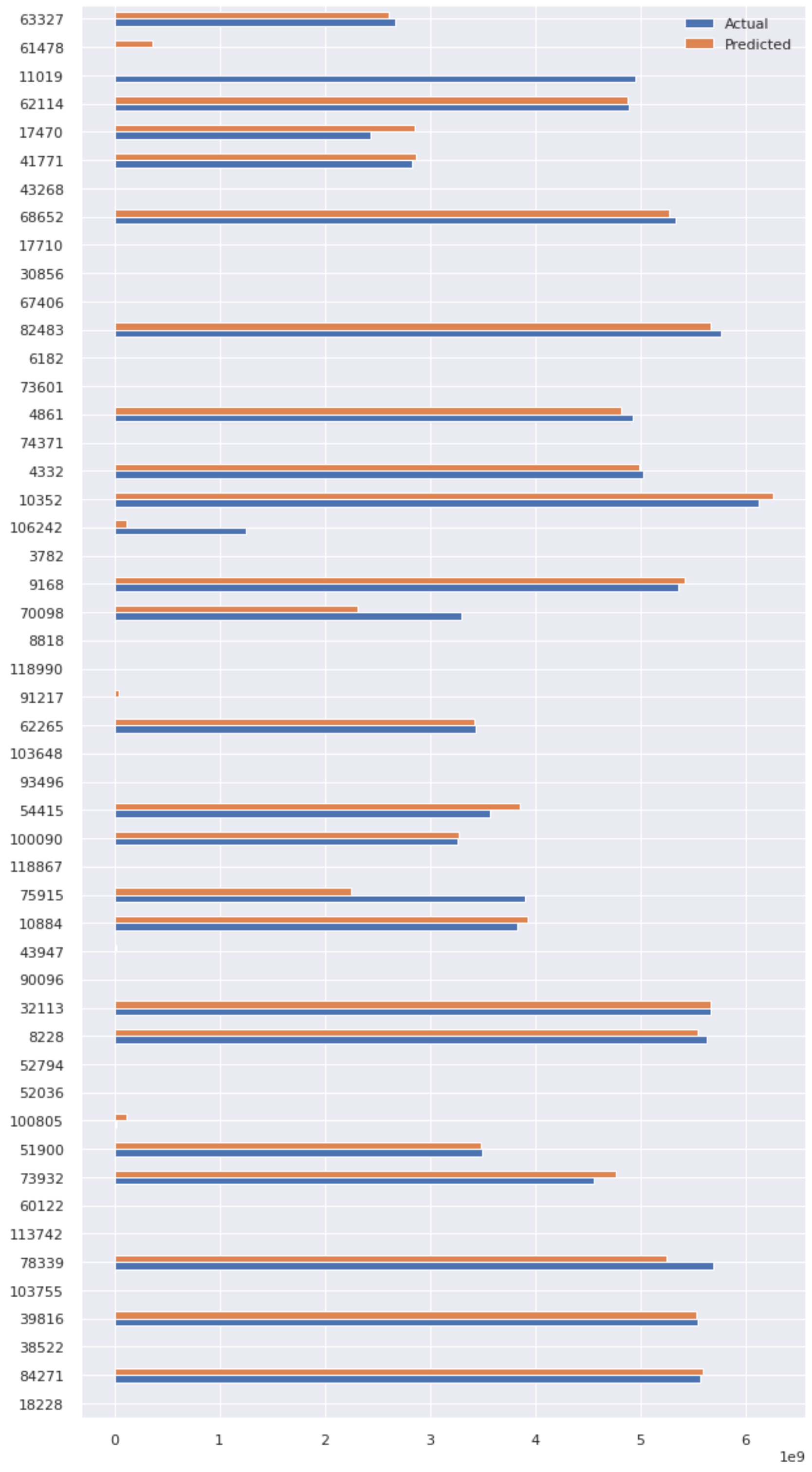


Figure 18: Predictions sampled from test set, neural network.

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)
```