University of Hertfordshire UH

PROJECT REPORT

MSc

Comparison of Predictive and Prognostic modelling approaches in Analysis of COVID-19

Name: Ward Ali Dib

Supervisor: Oleg Blyuss

Table of Contents

1) Introduction

COVID-19’s first confirmed case was reported on 31 December 2019 in Wuhan City, China. It was reported as a case of pneumonia with unknown causes. On 7 January 2020, Chinese authorities identified a new virus, which was temporarily named “2019-nCoV”. As the number of cases outside China increased, the WHO characterised the outbreak as a pandemic. Shortly after, the virus had crossed many borders, with 200 million confirmed cases and 4.2 million deaths worldwide. [1]

The disease can range from mild to severe, and the symptoms include but are not limited to; fever, continuous cough, and a loss or change in taste or smell. Studies also showed that COVID-19 may cause long-term health problems, with different symptoms that may overlap or fluctuate over time [2]. Studies estimated that 1 in 10 patients continue to experience symptoms even after the initial 12 week period. 55 different long-term effects were highlighted, the most common including fatigue, headaches, cough, cognitive impairment, pulmonary fibrosis, and organ damage [3].

As well as symptoms varying from person to person, the mortality rate of COVID-19 also varies, depending on factors such as age, sex, and existing comorbidities. As such, techniques such as predictive and prognostic modelling are invaluable for assessing risks factors, and highlighting the most common demographic characteristics that affect survival.

This report aims to present and compare different predictive and prognostic modelling methods, and apply those techniques to a dataset containing a cohort of patients consecutively admitted to the Sechenov University Hospital Network in Moscow with suspected COVID-19. The data was extracted from medical records of adult patients, at least 18 years of age, admitted between 8 April and 28 May 2020 [4].

We will compare various predictive modelling techniques by assessing their performance on the dataset, to determine mortality rate in relation to different diagnostic measures. We will also perform prognostic modelling, to predict the risk of future clinical outcomes in individual patients within a specific timeframe.

The dataset consists of records of 2237 patients, with 9 variables of the medical predictor type (age, sex, days between admission and result, and various comorbidities) and three of the two objective outcomes type; whether the patient admitted has COVID-19 or not, whether they’ve been admitted to the ICU or not, and whether they have recovered or passed away.

Gathered data often contains anomalies, impossible data combinations, missing values, etc. Analysing any dataset before it has been screened for such errors can produce misleading results. Thus, this dataset has been pre-processed and cleaned to contain no missing data, in preparation for further analysis.

2) Predictive modelling

Prediction modelling is mathematical processes designed to predict outcomes by identifying patterns within a given set of variables. It has grown important in medical research, as accurate predictive models can provide crucial insight to both patients and doctors. This insight can be used to monitor the course of an illness, or identify the risk of developing a certain an illness. This helps in providing personalised patient care, and aids in creating effective treatment plans [5].

Predictive modelling has also become a popular way to assess and manage high healthcare costs as well. It can be used to identify patients who are more prone to certain illnesses (such as diabetes or breast cancer) and thus facilitate implementation of preventative measures before the disease can progress to advanced stages [6].

2.1 Types of predictive modelling

Prediction models in medicine traditionally use a Bayesian framework approach, which applies Bayes' theorem to update a hypothesis's probability whenever new information becomes available. Although, a variety of different techniques for developing predictive algorithms exist. Statistical and data mining techniques are being used more often. Some examples include logistic regression, neural networks, and random forest [5]. The analysis can be conducted using programmes such as "RStudio" or "Spyder".

Machine learning algorithms can have many advantages over statistical models. This includes the lack of a predefined hypothesis. When many predictors are available, and there are interactions between them, it may be effective to approach prediction problem without any hypothesises. This reduces the likelihood of overlooking unexpected predictor variables. This is common with electronic health records. Thus, this could help professionals recognise clinically important risk factors in patients who have several marginal risk factors, all of which may have not been identified otherwise [5].

In this section, we will look at four different types of predictive modelling techniques:

- Logistic regression
- Neural Networks
- Support vector Machines
- Random Forest

All of which have their optimal applications and are widely used in different industries such as retail, entertainment, and healthcare. Then, we will briefly discuss how to develop, validate, and assess a predictive model.

2.1.1 Logistic regression

Logistic regression is a statistical prediction model. It's commonly used because it allows for multivariate modelling of a binary dependent variable. It works by analysing the association between one (or many) independent variables, and the chosen binary dependent variable. Binary variables are categorical variables that can take only two different values. For example, "yes vs. no" or "dead vs. alive". Thus, it can model the probability (or risk) of an outcome based on individual characteristics [7].

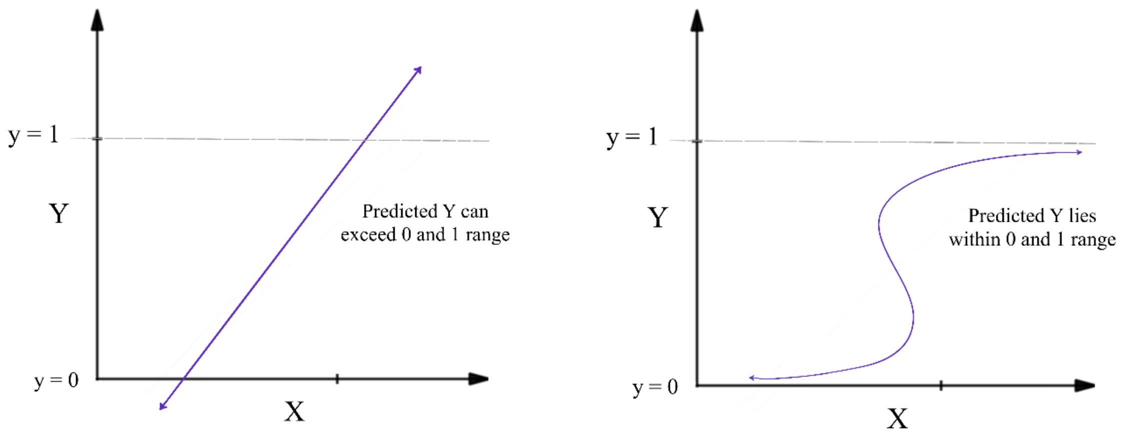

y = 1
Y
Predicted Y can exceed 0 and 1 range
y = 0
X
y = 1
Y
Predicted Y lies within 0 and 1 range
y = 0
X

Figure 1 – linear vs. Logistic Regression

As we can see in Figure 1, modelling the relationship between an independent variable (X) and the probability cannot happen linearly, as that would result in predictions that fall outside of the natural range of probabilities (0 to 1). This is where Logistic Regression comes in, to act as an extension of Linear Regression.

As such, it exhibits a linear relationship with the logit of the outcome. The regression coefficients represent the intercept (β_0) and slope (β_1) of this line. A simple example is shown in Figure 1. When solving for the probability (p), the relationship with the independent variable is sigmoidal, and as we can see, the estimated probabilities become constrained between 0 and 1 [8].

The basic equation for logistic regression with multiple independent variables is:

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_n x_n$$

Where:

- p indicates the probability of an event.
- β_1 is the intercept, and is a constant.
- $\beta_1 x_1 + \beta_2 x_2 + \cdots \beta_n x_n$ is the value of each independent variable (x_n) weighted by its respective beta coefficient (β).

Beta coefficients give the slope of the regression line. The slope represents the increase in outcome as the value of the independent variable increases by 1-unit. A larger beta coefficient means the corresponding independent variable is contributing more heavily to the outcome [9].

Application of logistic regression relies on its assumptions being met, which include:

- Independence of errors: Lack of duplicates responses or correlated outcomes [9].

- Linearity in the logit for continuous variables: the relationship between the variables and their respective logit-transformed outcomes must be linear. This assumption can be checked by creating a statistical term that represents the interaction between each continuous independent variable and its natural logarithm. Then, if a term is statistically significant, it violates the assumption [10].

- Absence of multicollinearity amongst independent variables: For example, variables like weight and body mass index are heavily correlated. So only one of them can be included in the same model. If the logistic regression model has highly correlated independent variables, the estimated beta coefficients of these variables will have large standard errors. To solve this, we can eliminate any redundant variables from the model [10].

- Lack of strongly influential outliers: For example, the actual outcome being very different from a patient's predicted outcome. If too many influential outliers exist, the accuracy of the model will decrease. After outlier detection, we would compare the overall model fit with and without the outliers. Outliers whose effect is not strong can be kept, while outliers with a strong influence will be removed [9].

- An adequate number of events per independent variable: We must select the smallest number of independent variables that provide the best explanations for the outcome. This is to prevent overfitting. If we selects 10 patients for a study, and include 10 independent variables in the logistic regression model, we will have an unstable model that overfits and does not provide adequate analysis. Overfitting will give estimates for the beta coefficients that are too large, and give higher errors. There is no universal rule for the number of outcomes per variable, but some common rules of thumb like 10 outcomes per 1 variable are used, as well as statistical tests that can be performed to check for overfitting, such as cross-validation [9].

Logistic regression has great advantages that makes it favoured among medical researchers in comparison to similar approaches. It is a simple machine learning algorithm that is easy to implement, yet provides great efficiency. It allows models to be updated easily with new data inputs. And most importantly, we can interpret the exponentiated logistic regression slope coefficient as an odds ratio. The odds ratio is a numerical way to represent the strength of association between the outcome and risk factor, or how much the odds of a particular outcome change per 1-unit increase of the independent variable. If the odds ratio is <1, odds are decreased for a given outcome, while an odds ratio>1 means the odds are increased [7].

However, logistic regression has limitations, such as assuming linearity between the dependent variable and the independent variables. Linearly separable data is rare in real-world scenarios. In addition, it is difficult to get complex relationships using logistic regression, and it can be outperformed by other models [9].

2.1.2 Artificial Neural Networks

Artificial Neural Networks (ANN) is a complex algorithm that is used for predictive analysis. It is inspired by the biological structure of the human brain. Although, in comparison to the brain it is rather simple, but it works well enough for the intended purposes.

Neural networks are used for data classification. They process current and past data, to estimate future outcomes. They can also discover any complex correlations or patterns hidden in the data, and produce an output free of noise. They can be used to make predictions on time series data, such as climate data [11].

The structure of an ANN consists of interconnected simple elements – called neurons. The core concept of such a structure is that neurons can be modelled with simple automata, and all the complexity, flexibility, and other pivotal qualities of a real brain can be mimicked by the connections between neurons. Each neuron takes a set of input values; each has an associated weight, and a numerical value called bias. This is used to match the input to the output [12].

The neuron operates in two stages. During the first stage, each predictor variable is multiplied by its weight. Then, a score is generated by adding the values together. In the second stage, the "activation function” transforms the score, so it lies within a fixed range (often 0, +1 or –1, +1). The score is the neuron’s output [11].

Several neurons are connected together to create a neural network model. The outputs from one layer are the inputs to the next layer. The structure of a neural-network has three layers:

- <u>The input layer:</u> feeds previous values into the next layer. It represents the predictor variables with one input each, and does not contain neurons.

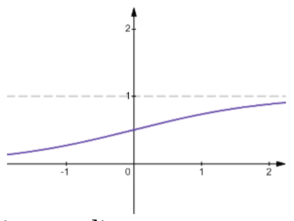
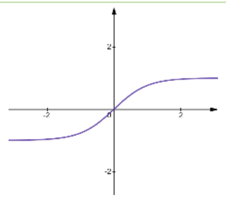
- <u>The hidden layer:</u> contains functions that generate predictors. The mathematical functions that modify the input data inside the hidden layers are the neurons. Their number depends on the problem [11].

- <u>The output layer:</u> collects predictions from the hidden layer and produces a result. This is the model’s prediction that produced by the output layer’s only neuron. This neuron combines the scores from the hidden layer’s neurons using a linear model. Finally, the output layer neuron transforms the score via its activation function. This can either work for a classification problem or regression problem [12].

Now, we can discuss how to build an Artificial Neural Network. There are four key elements to creating an ANN: activation function, weights, cost function, and bias.

The activation function defines how the input’s weighted sum is transformed into an output from a node (or nodes) in a layer. It can either be linear or nonlinear. Non-linear activation functions are more commonly used, as they give the model the ability to deal with various parameters and perform complex tasks. Most activation functions are differentiable, in order to facilitate *backpropagation* – the method most commonly used to train ANNs – which requires the derivative of prediction error [13].

There are many choices of activation functions, different functions may be used in different parts of the model. Hidden layers usually use the same activation function, while the output layer will use a different one. The choices are all dependent on the problem and type of prediction required. Table 1 shows a few common activation functions.

Activation Function	**Equation**	**Notes**
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	- The Sigmoid function exists in the range 0 to 1. Thus, it is especially useful for binary classification problems, or the output is a probability. It’s used in feedforward neural networks. - This function is differentiable, but also prone to problems like the vanishing gradient or saturation of later layers [12]. 
Tanh	$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	- Mainly used in classification between two classes, also in feedforward networks. The range is from -1 to 1. - It is differentiable, and converges quicker than the Sigmoid function [14]. 

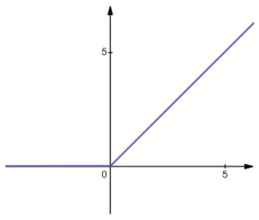
RELU (Rectified Linear Unit)	$f(x) = \begin{cases} 0 \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases}$	- It’s the most used activation function. It’s fast, easy to train, and achieves better performance. It’s also differentiable, which facilitates backpropagation. It overcomes the vanishing gradient problem. - The pitfall of this function is that all negative inputs are mapped to zero immediately, which decreases the model’s ability to fit and train, and the graphs do not map negative inputs appropriately [14]. 

Table 1 – Activation Functions

A neural network learns through a process called “forward propagation”. This is the process of feeding the input values into the neural network, to get a predicted value i.e. output. When the input is fed into the first layer, there is no operations taking place. The second layer takes values from first layer and applies the activation function to these values. Then, the resulting value is passed onto the next layer. This process is repeated until we achieve an output in the last layer [15]. The process of a neural network with two hidden layers is as follows:

$x = a_1 \ \ input\ layer$

$z_2 = \ W_1 x + \ b_1 \ \ is\ the\ neuron\ value\ at\ first\ hidden\ layer$

$a_2 = f(z_2) \ \ activation\ value\ at\ first\ hidden\ layer$

$z_3 = W_2 a_2 + b_2 \ \ neuron\ value\ at\ second\ hidden\ layer$

$a_3 = f(z_3) \ \ activation\ value\ at\ second\ hidden\ layer$

$s = W_3 a_3 \ \ output\ layer$

The weight is a value that indicates the “strength” of the connection. The larger the weight, the higher the influence of the neuron on the input side of that weight [12].

The weights associated with each neuron also have to be updated during the process of forward propagation. This is called “backpropagation” and works by repeatedly adjusting the weights in order to minimise the measure of the difference between the real output vector and the desired output vector. This measure of difference is *the cost function* – i.e. backpropagation minimises the cost function by adjusting the weights and biases [15].

As for the cost function, it is denoted as:

$$C = cost(s, y)$$

It evaluates the predicted output s against the expected output y. $cost$ can be many functions. Some are mentioned in Table 2 below.

Cost function	Used	Equation	Definition
Mean Squared Error (MSE)	Regression problems	$MSE = \frac{1}{n}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$	It is the sum of the squared differences between true value and prediction. The output is a single number, and it represents the cost [16].

Cross-Entropy	Classification problems	$H = -\sum_{i=1}^{\substack{output\\size}} y_i \cdot \log(\hat{y}_i)$	It's a measure of the difference between two probability distributions for a set of events [17].

Table 2 – Cost Functions

First, the cost function is used to calculate the error value. Then, the derivative of the error of each weight is calculated using chain rule. This means that for one weight $\left(w_{jk}\right)^l$, the gradient is as follows [18]:

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} \quad chain\ rule$$

$$z_j^l = \sum_{k=1}^{m} w_{jk}^l a_k^{l-1} + b_j^l \quad m\ is\ the\ number\ of\ neurons\ in\ layer\ l-1; by\ definition$$

$$\frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1} \quad by\ differentiation$$

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} a_k^{l-1} \quad is\ the\ final\ value$$

This process is repeated until a gradient for every weight is found. Then, the gradient value is subtracted from the weight value. This is to reduce the error value. As such, we keep getting closer to the minimum loss, i.e. the local minima of the cost function. This process is known as Gradient Descent [15].

The bias is a constant input, and is always 1. It has its own connection weight. This is to insure the availability of an activation in the neuron, even if all the other inputs are 0.

Adjusting the bias using backpropagation follows the similar process:

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} \quad by\ chain\ rule$$

$$\frac{\partial z_j^l}{\partial b_j^l} = 1 \quad by\ differentiation$$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \quad is\ the\ final\ result$$

The common part is determined using chain rule and called the "local gradient". It's denoted as $\delta_j^l = \frac{\partial C}{\partial z_j^l}$, and it allows the optimisation of model parameters [18], as such:

```
$while\ (termination\ condition\ not\ met)$
    $w := w - \epsilon \frac{\partial C}{\partial w}$
    $b := b - \epsilon \frac{\partial C}{\partial b}$
$end$
```

1) Chose the initial values of w and b randomly.
2) Epsilon (ϵ) is the learning rate. It controls the gradient's influence each time the model is updated.
3) w and b are the weights and biases.
4) Termination condition is met once the cost function is minimised [18].

An advantage of ANNs is that they can solve problems with unknown regularities, whilst giving more accurate predictions than other models. The model can also be conveniently modified when new data is obtained. Moreover, they work well with time series with small observational intervals. Thus, it can be used in areas where we are wanting observations daily or weekly, etc, such as the spread of a disease or development of certain symptoms across a period of time [12].

The main disadvantage of ANN models is that they require a lot of time and resources to build. But despite that, in practical applications, they can be successfully used to predict the development of certain illnesses, the geographic spread of outbreaks in real time, or even predict patients' weekly average expenditures on medications [19].

2.1.3 Support Vector Machines

Support Vector Machines, abbreviated as SVMs, are widely used for classification purposes, but their principles can be easily extended to time series prediction and regression. They share similarities with ANNs, but they were developed in the framework of Statistical Learning Theory [20].

SVMs are a non-linear method. The end goal is to separate two types of behaviour. They work to find a hyperplane in an N-dimensional space, to distinctly classify data points. N is the number of features – i.e. if there are 2 input features, the hyperplane is a line. If there are 3, then the hyperplane is a two-dimensional plane, and so on. This gives many possible hyperplanes we can chose from to separate two classes of data. The goal is finding a plane with the maximum distance between data points of the two classes. This is referred to as "maximising the margin". Maximising the margin distance reinforces the model, so future data points can be classified more easily [21].

Mathematically, SVMs work to find the equation of the line that maximises the margin. This becomes the model. Data points on either side of the line can be aligned with different classes. The score of the model is then a measure of how likely a data point is to lie on either side of this line [20].

Drawing a straight line does not always separate the data easily. This is when non-linear SVMs come in. In those, Kernel functions are used to transform a non-linear space, into a linear one, so it can be classified. As such, we'll split this section to discuss linear and non-linear SVMs [22].

- Linear SVMs:

To build a SVM model, two elements are needed: the hypothesis spaces, and the loss functions.

The traditional view of SVMs is that they find a solution in the form of an "optimal" hyperplane. The simplest formulation of SVMs is linear. This means the hyperplane lies on the space of the input data x. In this case, the hypothesis space is a subset of all hyperplanes, and is a linear function of the form $f(x) = wx + b$, which is always used to solve regression problems [20].

The best line is defined as that line which minimises the following cost function (Q):

$$Q = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} L^{\varepsilon}(x_i, y_i, f)$$

$$Subject\ to \begin{cases} y_i - wx_i - b \leq \varepsilon + \xi_i \\ wx_i + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i,\ \xi_i^* \geq 0 \end{cases}$$

There are three parts to this equation.

1- First is the cost function, which is a *weight decay*. It is used to keep weight sizes regular and penalizes large weights. Due to it, weights converge to smaller values. This function is important because large weights deteriorate the generalization ability of the model, as they often cause excessive variance. Similar approaches can be observed in neural networks [23].
2- Second is the penalty function. It penalizes any error larger than $\pm\varepsilon$ using a "ε-insensitive" loss function L^{ε} for each of the N training points. C is a positive constant that determines how much deviation from ε is tolerated. Errors larger than $\pm\varepsilon$ are denoted with the slack variables ξ (above ε) and ξ^* (below ε), respectively [23].
3- Third are constraints. These are set to the errors between the regression predictions $(wx_i + b)$ and the true values (y_i). The values of both ε and C have to be chosen manually – the best values are usually dependant on the data and the problem [23].

In order to maximise the margin, we thus need to minimize $||w||$. The minimisation of w is an example of minimisation with constraints in optimisation theory. Thus, it can be solved by applying Lagrangian theory, and then, it can be derived that the weight vector, w, is the linear combination of the training data:

$$w = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) x_i$$

α_i and α_i^* are the Lagrange multipliers. They are associated with a specific training point. The asterisks, denote which multiplier is above or below the regression line [22].

Using this formula into the equation of a linear function, the following solution is obtained for an unknown data point x:

$$f(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot \langle x_i, x \rangle + b$$

Data points closer to the hyperplane are the support vectors. They influence the position and orientation of the hyperplane. Using these vectors, we can achieve the goal of maximising the margin of the classifier [22].

- <u>Nonlinear SVMs:</u>

Sometimes, the datapoints are not easily separable by a straight line. To solve this, we can gain linear separation by mapping the data to a higher dimensional space. This is done by replacing x_i by a mapping into feature space, $\varphi(x_i)$, which linearises the relation between x_i and y_i, as shown in fig. In the feature space, the previous approach can be used in finding the regression solution [23]. When using a mapping function, the solution can be changed into:

$$f(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot K(x_i, x) + b \text{ where } K(x_i,\ x) = \langle \varphi(x_i),\ \phi(x) \rangle$$

Where K is the *kernel function*. It's proven to simplify the use of mapping, which can become difficult because the specific mapping for x's dimensions has to be known.

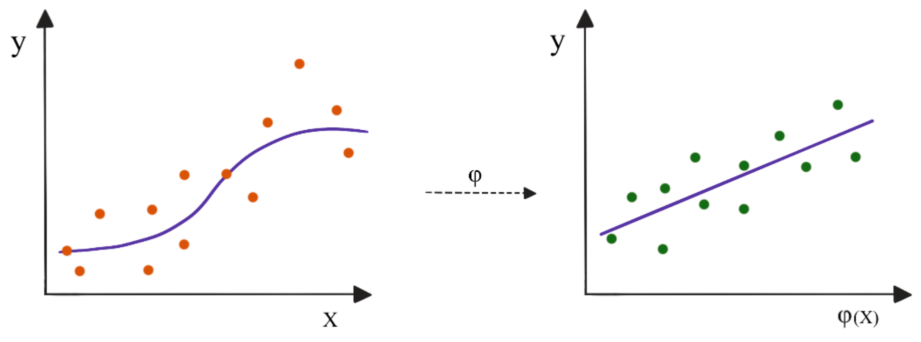


Figure 2 – mapping using kernel function

Using the kernel function, the data can be mapped into a feature space without full knowledge of φ, thus it is very efficient. Representing the mapping by using a kernel reduces the problem to finding kernels that identify families of regression formulas. The most used kernel functions are [22]:

- The Gaussian RBF. With a width of σ:

$$K(x_i, x) = \exp\left(\frac{-0.5\|x_i - x\|^2}{\sigma^2}\right)$$

  If the value of σ is set to a very large value, the RBF kernel approximates the use of a linear kernel, which is a polynomial with an order of 1.
- The polynomial kernel. With an order of d and constants a_1 and a_2:

$$K(x_i, x) = \left(a_1 x_i^T x \; + \; a_2\right)d$$

The choice of a kernel, its specific parameters, ε, and C, do not follow from the optimisation problem and have to be tuned manually. Except for the choice of the kernel function, the other parameters can be optimised in many ways, including Vapnik –Chervonenkis bounds, cross-validation, an independent optimisation set, or Bayesian learning [23].

Like all methods, SVMs have advantages and disadvantages. The choice of model always depends on the data. Some advantages of SVMs include their ability to be used for both regression and classification problems, and their effectiveness even with high dimensional data. They perform well in data where the number of features is higher than the number of rows of data, which makes them very useful in medical analysis and studies with small sample sizes [11].

The main disadvantages of SVMs come from non-linearly separable data, where selecting an optimal kernel may be difficult, more time is required to train, and overlapping classes may exist.

SVM is often used for classification tasks, such as categorisation, detecting spam emails, and image classification. The first time SVMs were successfully used to support medical decisions was to diagnose Tuberculosis from photomicrographs of Sputum smears [11].

2.1.4 Random Forest

A “Decision Tree” is a tool that uses a tree-like model of decisions and each of their possible consequences. It has been discovered that one of the best ways to improve the performance of Decision Tree based algorithms is to use ensembles of trees [24]. In this section, we will discuss one of these ensemble methods, known as Random Forest.

Random Forest is machine learning technique used for classification and regression purposes. It improves the performance of the Decision Tree while retaining most of its appealing properties, and it is among the top performers in prediction accuracy [24].

It consists of a large number of individual Decision Trees operating as an ensemble. The ensemble of B trees $\{T_1(X), \dots, T_B(X)\}$, where $X = \{x_1, \dots, x_p\}$ is a p-dimensional vector of properties, associated with an entry. The ensemble produces B outputs $\{\hat{Y}_1 = T_1(X), \dots, \hat{Y}_B = T_B(X)\}$ where $\hat{Y}_b, b = 1, \dots, B$, where B is the prediction for an entry by the b^{th} tree. Outputs of all trees are aggregated to produce one final prediction, $\hat{Y}$ [24].

For classification problems, $\hat{Y}$ is the class predicted by the majority of trees. For regression, it is the average of the individual tree predictions.

Given a dataset of N-entries for training, $D = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$, where $X_i, i = 1, \dots, n$, is a vector of descriptors. Y_i is either the corresponding class label (e.g., active/inactive) or activity of interest, the training algorithm is as follows:

1) From the training data of N-entries, draw a *bootstrap* sample. This means, randomly sampling from the dataset with replacement, resulting in different trees. This process is called *bagging*. Note that bagging does not split the data into smaller subsets and train them individually. For example, if the training data was [1, 2, 3, 4, 5, 6], bagging might give one of the trees the list [2, 3, 3, 1, 1, 6].
2) For each bootstrap sample, grow a tree with the following modification: at each node, choose the best split among a randomly selected subset of m_{try} (rather than all) descriptors. Here m_{try} is the only tuning parameter in the algorithm. The tree is grown to the maximum size, i.e. until no further splits are possible, and not trimmed back.
3) Repeat the above steps until (a sufficiently large number) B such trees are grown [24].

The approach is illustrated in Figure 3 below.

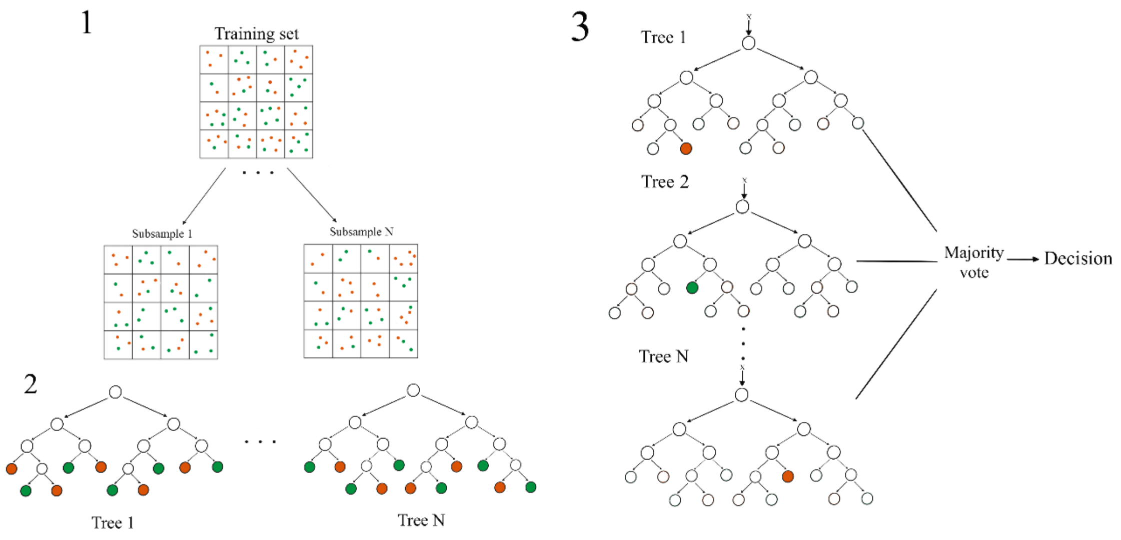


Figure 3 – The three steps of a Random Forest algorithm

There are two key conditions needed for the Random Forest approach to work.

1) Descriptor importance: Random Forest inherits the ability to select important descriptors and discard irrelevant ones from Decision Trees. However, these descriptors need to have at least some predictive significance in order to produce a good model [24].

2) Low correlation between models: uncorrelated models can produce more accurate ensemble predictions than any of the individual predictions. This is because low correlation allows the trees to protect themselves from individual errors. Meaning, if some trees are wrong, others are right, thus, the model is able to move in the right direction [24].

It is one of the most favoured methods in classification, as it combines many desirable features. It handles high-dimensional data well, can ignore irrelevant descriptors, handles multiple mechanisms of action, and is amenable to model interpretation. The Random Forest algorithm is very efficient with a large number of descriptors. Although, it is unsuitable for very linear datasets, sparce datasets, and datasets where one variable is much more important than all the others. In such cases, other methods are better [24].

2.2 Developing a predictive model

The different approaches to developing a predictive model depend on the kind of problem at hand. The first step, when using traditional regression, is *feature selection*. As the name suggests, it selects relevant candidate predictor variables for inclusion in the model. However, there is no consensus for the best strategy to do so [5].

A backward-elimination approach starts with the candidate variables, and performs hypothesis testing to find out which variables are better removed from the final model. On the other hand, a full model approach has all candidate variables. This is to avoid selection bias and potential overfitting. Significant predictor variables should be included in the final model regardless of their statistical significance, however, the number of variables is limited by the size of the dataset [5].

Poor variable selection results in poor model performance. This is less of a problem for machine learning techniques, as they are not just based on predefined hypotheses.

Other issues should be considered when developing a predictive model. These include handling missing data, and transforming variables. Missing data can either be removed or imputed to the mean, depending on the dataset. Variables of different types can be transformed into other types to aid in analysis, e.g. transforming categorical variables into numerical, normalising variables of different units, etc [5].

2.3 Validating a predictive model

For a prediction model to be valuable, it must have both the predictive ability in the derivation subset, and also perform well in a validation subset. A model's performance may differ greatly between these two subsets, depending on reasons such as overfitting, failure to include important variables in the model, measurement errors, etc. Thus, a model performance may be "too good" or overly optimistic, which does not guarantee that it will perform similarly on new data [5].

Validation can be either internal or external validation. Internal validation approach splits the dataset into two portions – a "training set" and a "validation set". Sometimes, splitting the data is not possible in limited sample sizes. For such, other internal validation methods exist such as bootstrapping or k-fold cross-validation.

In k-fold cross-validation, the sample is divided into k subsets, called "folds" (most commonly 5 or 10) and one subset is chosen to be the validation subset, while the remaining $k - 1$ subsets are used for training the model. Then, the validation subset is used to calculate the model's prediction error. After that is completed, the process restarts, with a different fold as the validation subset. This is repeated k times, and the model's performance is the average of the errors calculated in each step [25].

The advantages of this method are many, as it’s simple, easy to implement, and each observation is only used once for validation, which reduces bias.

Machine learning algorithms follow a different approach. In Random Forest specifically, a method called “in-bag” and “out-of-bag” sampling is used. This is where the initial data is divided into two groups of samples: “in-bag” and “out-of-bag”. The in-bag sample is created using random sampling with replacement from the initial dataset. This creates a subset equivalent in size to the initial set. On the other hand, out-of-bag samples consist of the unsampled data from the initial dataset, and typically include about a third of it. The “out-of-bag” subset can serve as an internal validation subset for the model derived using the “in-bag” sample [5].

Unfortunately, internal validation often yields optimistic results given that both the derivation and validation data sets are very similar. Although, external validation is a lot more difficult to accomplish, as it requires similar data collected in a different setting, but it is still always preferred to internal validation [5].

2.4 Assessing the performance of predictive models

Predictive models are judged based on their ability to make accurate predictions. The methods used for judging depend on the dataset and the model used. There are several tests to assess the performance of a predictive model, which assess overall performance, calibration, discrimination, and reclassification [26]. Some methods are shown and summarised in Table 3 below.

Measure	Description
Brier score	Average square distances from the predicted and the observed outcomes. It is used as a way to verify the accuracy of a probability forecast. Smaller scores (closer to zero) indicate better forecasts. It’s given by: $BS = \frac{1}{N}\sum_{t=1}^{N}(f_t - o_t)^2$ Where: N = the number of variables, f_t is the forecast probability, o_t is the outcome (1 if it happened, 0 if it didn’t) [25].
Receiver Operating Characteristic (ROC) Curve	Overall measure of how effectively the model differentiates between events and non-events. It can summarise a model’s performance by evaluating the trade-offs between true positive rate (sensitivity) and false positive rate ($1 -$ specificity) [27]. Sensitivity and specificity are inversely proportional, meaning that as the sensitivity increases, the specificity decreases, and vice versa. The area under curve (AUC), referred to as index of accuracy (A), is a perfect performance metric for ROC curves. Higher AUC means better prediction power of the model [27].
Confusion matrix	It is a tabular representation of Actual vs. Predicted values.

		Predicted	
		No	Yes
Actual	No	True Positive (D)	False Negative (C)
	Yes	False Positive (B)	True Negative (A)

	The accuracy of the model can be calculated as such: $accuracy = \frac{D+A}{D+A+B+C}$ Specificity and sensitivity can also be derived as such [27]: $$sum\ to\ 1 \begin{cases} True\ Negative\ Rate\ (TNR), specificity = \frac{A}{A+B} \\ False\ Positive\ Rate\ (FPR), 1 - specificity = \frac{B}{A+B} \end{cases}$$ $$sum\ to\ 1 \begin{cases} True\ Positive\ Rate\ (TPR), sensitivity = \frac{D}{C+D} \\ False\ Negative\ Rate\ (TNR) = \frac{C}{C+D} \end{cases}$$

Table 3 – Methods to access model performance

3) <u>Prognostic modelling</u>

Prognostic models are statistical tools that are used to assist in predicting outcomes within a specific period. A prognostic model uses a combination of predictor values and converts them to an estimate of the risk (or probability) of a specific outcome within a specific period.

The ability to predict outcomes of diseases or treatment in invaluable in the medical field. E.g., it allows for prediction of the chance or duration of survival of a patient, progression of a disease within the system, and prediction of clinical events related to treatment responses. As such, prognostic models can open pathways leading to improvements in treatment, because it aids clinicians in making informed choices tailored to an individual's profile of prognostic factors [28].

3.1 Kaplan-Meier (K–M) curves

The Kaplan-Meier estimator was described by Edward Kaplan and Paul Meier in the Journal of the American Statistical Association in 1958, as part of a paper on how to deal with incomplete observations. Since then, it has become a way of dealing with differing time-to-event data, such as survival times. An example of when time-to-event may be an important end-point variable include COVID-19 survival times. Although the event is often discussed in the context of "survival", it may also be any event of interest. Hence, Kaplan-Meier analyses can be used in non-medical analysis [29].

The Kaplan-Meier method is a non-parametric method used to estimate the survival probability from observed survival times. Non-parametric methods do not rely on the assumption of an underlying probability distribution, and that makes K-M estimators powerful, since survival data has a skewed distribution [30].

The survival probability at time t_i, $S(t_i)$, is calculated as follow [31]:

$$S(t_i) = S(t_i - 1)\left(1 - \frac{d_i}{n_i}\right)$$

Where:

- $S(t_i - 1)$ = the probability of survival at $t_i - 1$
- n_i = the number of patients who survived just before t_i
- d_i = the number of events at t_i
- $t_0 = 0,\ S(0) = 1$

This statistic gives the probability that an individual patient will survive past a particular time t. At $t = 0$, the Kaplan-Meier estimator $S = 1$. And as $t \to \infty,\ S \to 0$. The estimated probability $\big(S(t)\big)$ is a step function – i.e. it changes value only at the time of each event. We can also compute confidence intervals for the survival probability [29].

Expanding on that, the K-M estimator is based on the assumption that the probability of surviving past a certain time point t equals the product of the observed survival rates until the time point t [31]. In other words, the survival probability at time t is given by:

$$S(t) = p_1 * p_2 * \dots * p_t$$

Where p_1 is the proportion of all patients surviving past the first time point, p_2 is the proportion of patients surviving past the second time point, and so on until t is reached. It is important to note that from p_2 onwards, we only account for the patients who have survived past the previous time point. This makes p_2 to p_t conditional on the previous proportions [32].

In summary, the model begins when the treatment begins, and ends when either the event of interest is reached, or the subject does not survive. This duration is called *serial time*. Unlike calendar time, this describes the clinical course time. In most clinical trials, individuals begin the study at zero time, without regard to when they entered the study. Thus, they may reach the end-point at vastly differing points along the trial calendar [29].

The Kaplan-Meier survival curve is a plot of the survival probability against time. This gives useful summaries about the data, and it can be used to estimate measures such as median survival time.

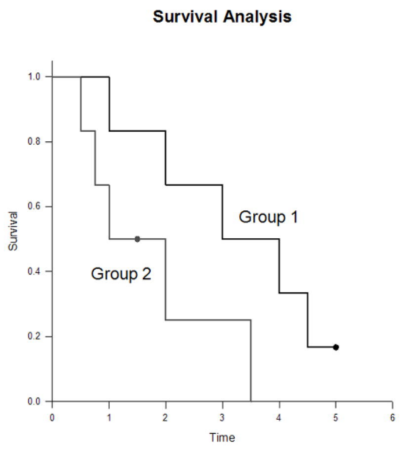


Figure 4 - a Kaplan-Meier curve generated by SigmaPlot [29].

Figure 4 shows an example of a K-M curve. One member of Group 1 survived until the end of the study. On the other hand, there were no survivors in Group 2 after 3.5 years [29].

In such cases of comparison, the log-rank test is used to compare survival curves. The log-rank test is a statistical hypothesis test that tests the null hypothesis that survival curves of two populations do not differ. Chi-squared testing is used to derive a $p\ value$ [32].

Usually, $p < 0.05$ is significant, and $p < 0.05$ is insignificant, i.e. it indicates that the two treatment groups are significantly different in terms of survival.

3.2) Cox Regression

Cox regression models are often used in medical research for investigating the association between the survival time of patients, and one or more predictor variables [33].

The goal of Cox Regression is to evaluate the effect of several risk factors at the same time. So, we would have to examine how specific factors influence the rate of an event happening at a point in time (e.g., survival, death). This is known as Hazard Rate. Risk factors also known as “covariates” in clinical analysis [33].

Cox regression is split into two types: proportional and non-proportional hazard models.

The non-proportional model is used when the proportional hazard assumption is not satisfied for a covariate. Namely, the assumption that the hazard ratio needs to be constant over time, and the effect of the given covariate on the hazard level is the same at all timepoints. For the purposes of this paper, we will only discuss the Proportional Hazard models [34].

The Cox Proportional Hazard model is expressed by the "hazard function" $h(t)$. The hazard function can be interpreted as the risk of the event happening at time t [32]. It can be estimated by:

$$h(t) = h_0(t)e^{(\beta_1 x_1 + \beta_2 x_2 + \dots + \beta_i x_i)}$$

where:

- t is survival time
- $h(t)$ is the hazard function determined by the set of i covariates $(x_1, x_2, \dots, x_i)$
- $\beta_1, \beta_2, \dots, \beta_i$ are the coefficients measuring the impact of covariates.
- h_0 is *baseline hazard*. It is the value of hazard if all covariates are equal to 0. It is also relevant to time t.

This can be written as a multiple linear regression of the logarithm of the hazard on the variables x_i. This would look like:

$$\ln\left(\frac{h_i(t)}{h_0(t)}\right) = \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_i x_i$$

The quantities e^{β_i} are the Hazard Ratios. If $\beta_i > 0$, then $e^{\beta_i} > 1$. Thus, as the value of the i^{th} covariate increases, the hazard increases, in other words, the length of survival decreases [32].

In summary:

- HR = 1: No effect
- HR < 1: Hazard reduction
- HR > 1: Hazard increase

As in the section above, the p-value is used to test the significance of the model. The null hypothesis would be that all the coefficients $\beta = 0$ [32].

The Cox Regression model is a powerful tool in clinical analysis, as it can be applied with both quantitative predictor variables, and categorical variables. Unlike K-M estimator, this method is not univariate. So it can extend survival analysis methods to simultaneously assess how several risk factors effect survival time. In clinical investigations, where many factors need to be considered in prognosis, this is very important [35].

4) Application to real data: COVID-19 patients

In this section, we will apply the predictive and prognostic modelling methods discussed previously on a dataset of patients admitted to the Sechenov University Hospital Network in Moscow City for suspected Coronavirus Disease 2019 (COVID-19) infection. This data was extracted from the medical records of adult patients admitted for suspected COVID-19 infection in Moscow between 8 April and 28 May 2020 [4].

For Logistic Regression, Random Forest, and Support Vector Machines, we used R programming language in RStudio. As for the Neural Network, we used Python in Spyder.

The dataset contains 12 columns. 6 of them being comorbidities with binary values. The sample size comprises of 2237 patients admitted to the hospital, of whom 93 have died. The coding for these variables is 1 = Yes, 0 = No. The variables are:

- ID: patient ID.
- chroniccard_mhyn: chronic cardiovascular disease.
- hypertension_mhyn: hypertension.
- chronicpul_mhyn: chronic pulmonary disease.
- renal_mhyn: chronic kidney disease.
- chronicneu_mhyn: chronic neurological disorder.
- diabetes_mhyn_2: diabetes.
- Age: Age in years.
- ICU: the fact of admission to ICU.
- Days: number of days between the admission and the outcome.
- COVID: results of the PCR test (0 for negative, 1 for positive).
- Sex2: female or male.
- Death: the main outcome - discharge (0) or death (1).

The raw data looks like this.

ID	chroniccard_mhyn	hypertension_mhyn	chronicpul_mhyn	renal_mhyn	chronicneu_mhyn	diabetes_mhyn_2	Age	ICU	Days	COVID	Sex2	Death
1	0	0	0	0	0	0	38	0	12.12736111	0	Female	0
2	0	0	0	0	0	0	48	0	8.702222222	0	Male	0
3	0	1	0	0	0	0	54	0	11.77461806	0	Male	0
4	0	0	0	0	0	0	58	0	12.670625	0	Female	0
5	0	0	1	0	0	0	31	0	11.77896991	1	Female	0
6	1	1	0	0	0	1	65	0	8.418738426	0	Male	0

4.1 Predictive Modelling

First, we split the dataset into two equal halves of training and test set. Then, we applied different models to it and produced the corresponding ROC curves, and Table 4 with the confidence intervals and AUCs to help us assess the models' performance.

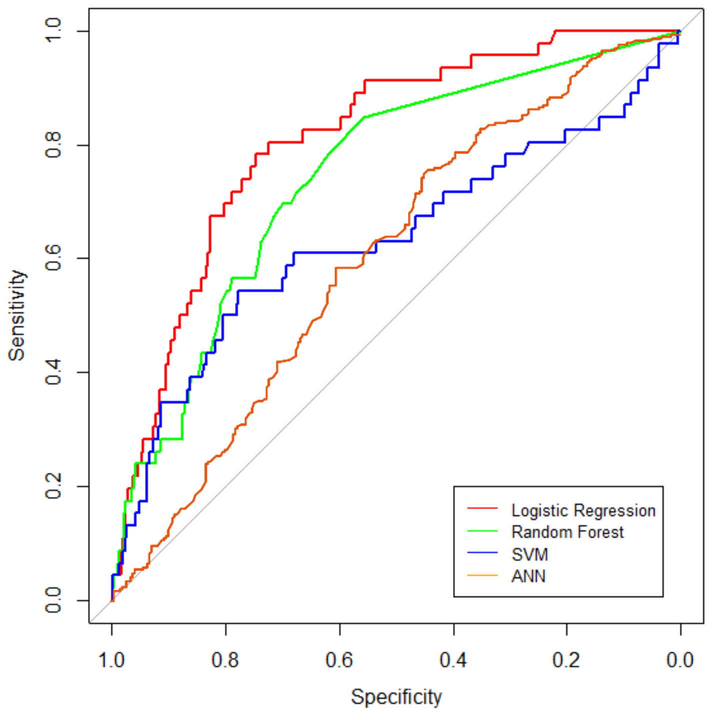


Figure 5 – ROC curves

As discussed previously, True Positive Rate (Sensitivity) is plotted against the False Positive Rate (1 – Specificity) [36]. In medical analysis, the TPR is the rate of correct identification of positive tests for the disease – in this case COVID-19.

A random classifier is expected to give points lying on the diagonal, where specificity = sensitivity. The closer the curve comes to the diagonal, the less accurate the model. On the other hand, models with ROCs closer to the top-left corner indicate a better performance [37]. From Figure 5 we can see this clearly with the Logistic Regression model, indicating it is performing the best out of the four tested.

Another testing method discussed is the area under the ROC (AUC). It's a good method as it summarises the performance of each model into a numerical measure. The minimum AUC should be considered a chance level i.e. AUC = 0.5. AUC of 0.7 to 0.8 is acceptable, and above 0.8 up to 0.9 is excellent [37]. However, if the AUC is too high it might indicate overfitting, class bias, or other problems in the model. This further confirms that the Logistic Regression model has the better prediction, with the highest AUC of 0.8, as seen in Table 4.

Model	AUC	Confidence Interval of AUC	Sensitivity at a Fixed 90% Specificity
Logistic Regression	0.801	0.752 – 0.868	0.413
Random Forest	0.744	0.675 – 0.813	0.283
SVM	0.632	0.532 – 0.732	0.348
ANN	0.612	0.598 – 0.677	0.316

Table 4 – Model performance assessments

The confidence interval is used in statistics to quantify the uncertainty of an estimate. In predictive modelling, it is used to assess the performance or error of a model. Often, larger samples from which the estimate is drawn, give more precise estimates, and a smaller (better) confidence interval. Whereas larger confidence intervals give less precise estimates [38].

Sensitivity is the ability of a model to predict a positive test result to an individual with the disease that is tested for. A highly sensitive model means there are few false negative diagnosis; as such, fewer cases of the disease will be missed. Specificity on the other hand, is the ability of a model to designate patients who do not have the disease as negative [39].

In the last column, we fixed specificity at 90%, and calculated the sensitivity at that point. If a model has 90% specificity, it means it can correctly predict 90% of patients that do not have COVID-19, but it will miss 10% of them. This results in a non-symmetric distribution, which is why we used the median, which gives the expected value for performance [39]. Table 4 shows the Logistic Regression model has the highest median.

The accuracy of these models above can be improved in many ways. For Logistic Regression the best methods would be normalisation, feature selection, looking for class imbalance, and exploring more classifiers. Random Forest can be improved by increasing the number of trees and lowering the number of estimators. SVM can be improved by adding more data. This is not always possible in limited sample sizes of patients, but the other widely used approaches such as treating outliers and feature selection can also be applied here. Improving a Neural Network can be tackled through many approaches. Hyperparameter tuning is usually the first line of action, i.e. changing the activation or cost functions, number of hidden layers, batch size, number of epochs to train on. The model used on this data had an AUC of 0.43 with 200 epochs, but when changing the epochs to 2000, the AUC improved to 0.61 as seen in Table 4.

Overall, Logistic Regression and Random Forest are the best models to apply on this data. Logistic Regression is commonly used in medical analysis, so this result was expected. Random Forest handled the number of predictors well too. SVMs and ANNs would fare better with a larger dataset.

Keeping in mind these models can be fine-tuned in order to achieve a higher accuracy, any model can be used, but it's better to use the most suitable model, in this case Logistic Regression.

4.2 Prognostic modelling

In RStudio, we first generated Kaplan-Meier curves for every variable alongside a p-value. This gives us an estimate of survival probability depending on time, with respect to a specific comorbidity.

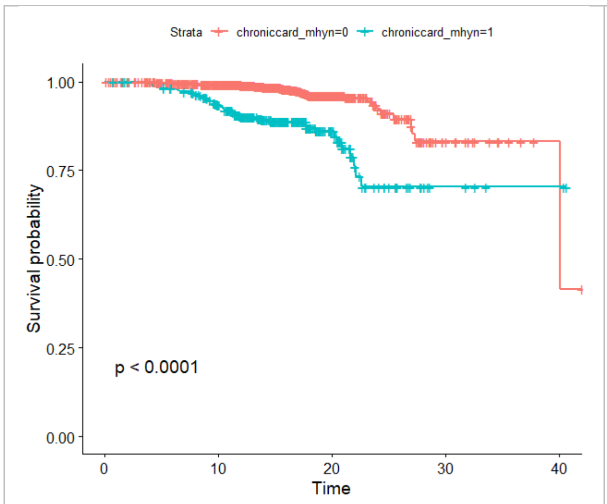


Figure 6

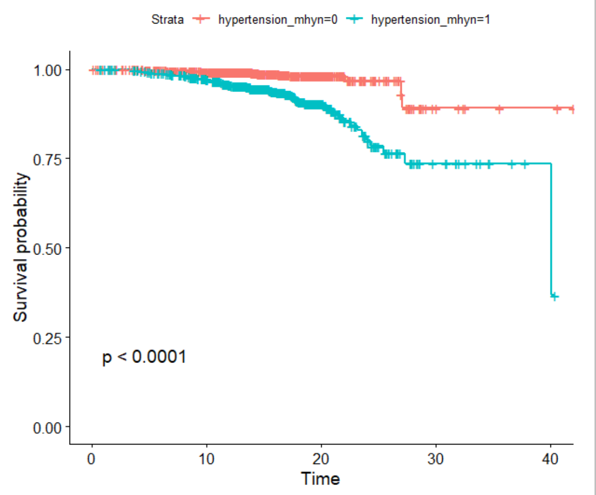


Figure 7

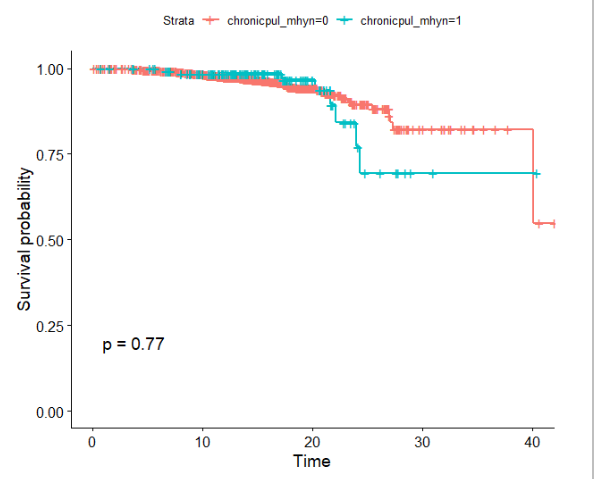


Figure 8

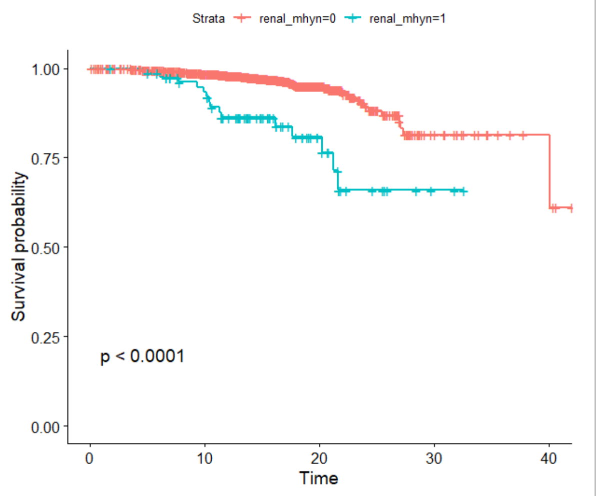


Figure 9

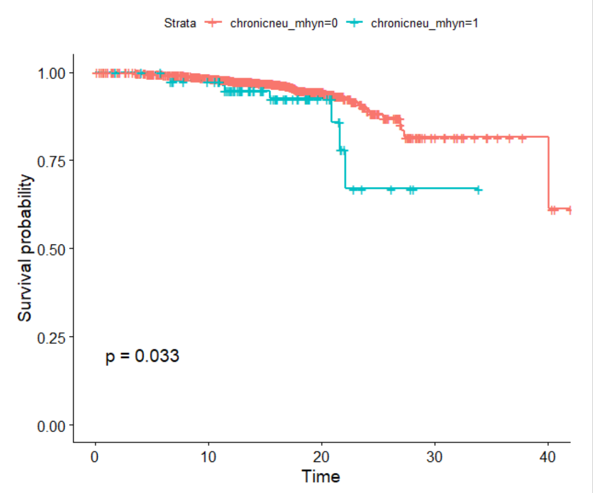


Figure 10

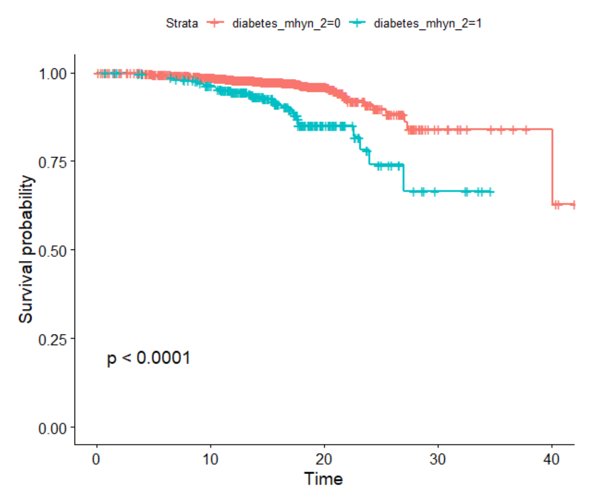
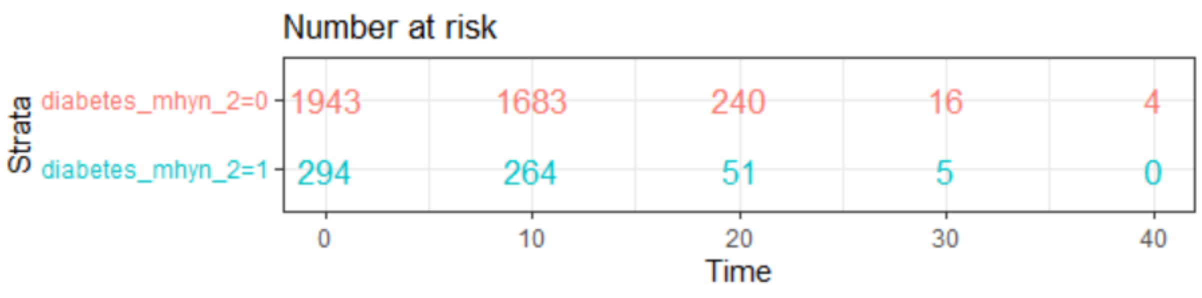


Figure 11

For comorbidities, it can be seen from Figures 6 to 11, that all of them contribute in some way to an increased risk of death. However, some comorbidities were not statistically significant, such as chronic pulmonary disease, and others were moderately significant such as neurological disorders. Comorbidities with $p < 0.0001$ have a significant impact on survival probability. For example, in Figure 11, after 30 days, no patients with diabetes had survived, whereas some patients without diabetes did.

This can be quantified by generating a summary Table 5. So 4 patients without diabetes survived by day 40, while none of the patients with diabetes did.



Number at risk

Strata	0	10	20	30	40
diabetes_mhyn_2=0	1943	1683	240	16	4
diabetes_mhyn_2=1	294	264	51	5	0

Time

Table 5 – Summary table

Other variables included are sex and age. We created a histogram to determine the most common age among the patients admitted.

From this, we can see the most common age is 50 years old. Thus, we divided the patients into two age categories of above and under 50 years old. The KM curve and summary table were as follows:

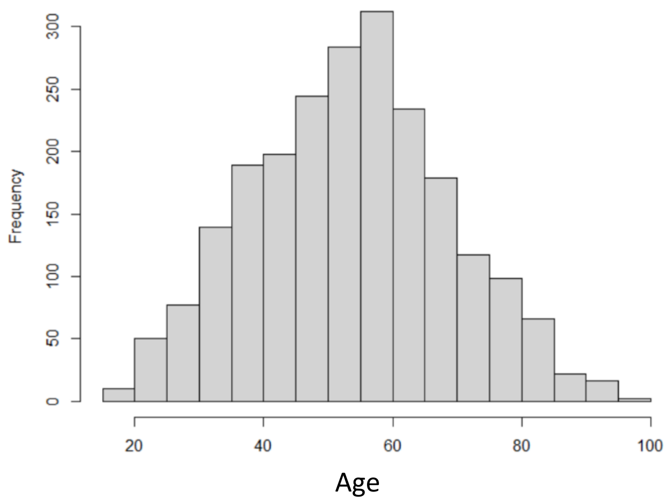
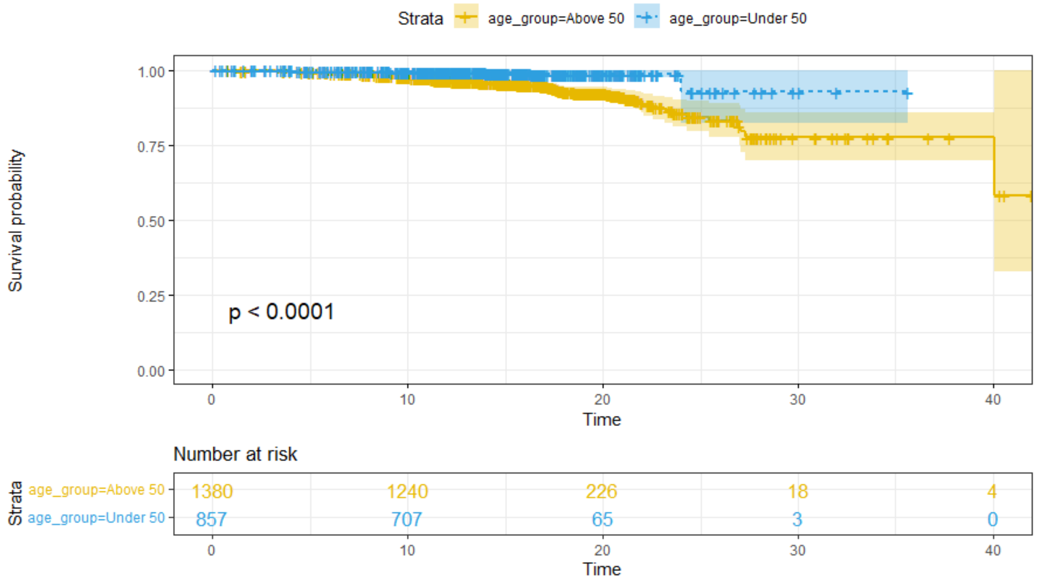


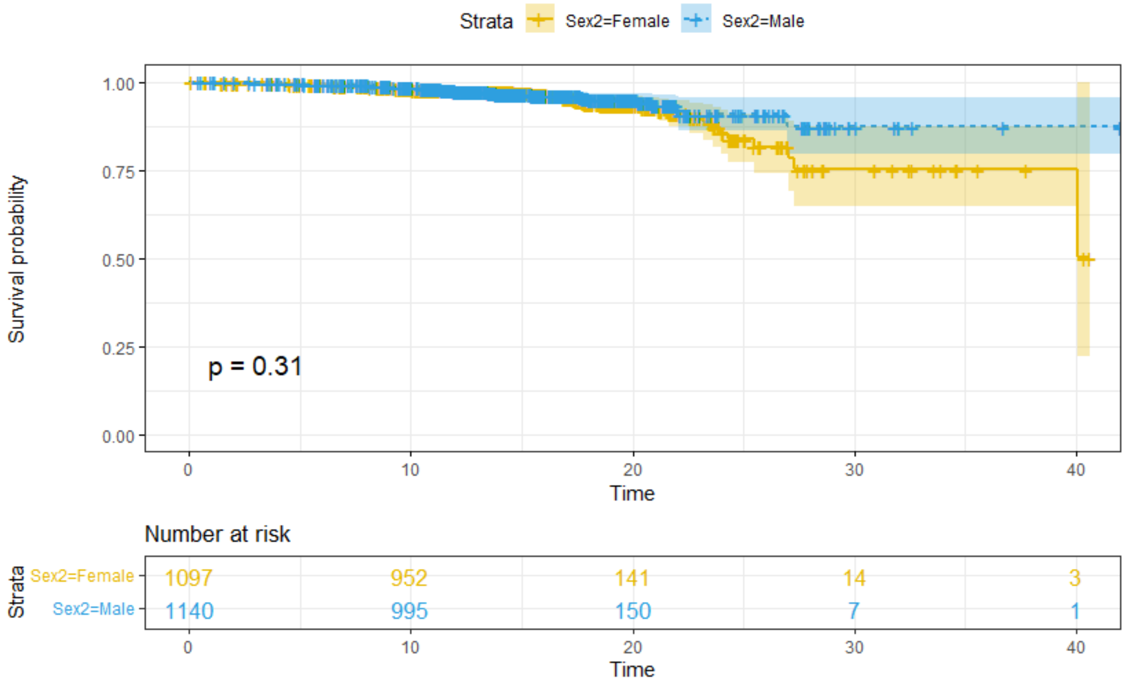
Figure 12 – histogram of age distribution



Strata	0	10	20	30	40
age_group=Above 50	1380	1240	226	18	4
age_group=Under 50	857	707	65	3	0

Figure 13 – KM curve and summary table for age groups

From the $p < 0.0001$, we can tell age is a statistically significant factor. More patients who are under 50 years old survived than those over 50. As for sex, more female patients survived than male patients, but the result is not statistically significant with a high p-value of 0.31, as shown in Figure 14.



Strata	0	10	20	30	40
Sex2=Female	1097	952	141	14	3
Sex2=Male	1140	995	150	7	1

Figure 14 – KM curve and summary table for sex

Now we can move onto Cox Regression. The variables we will use in this analysis are chronic cardiovascular disease, hypertension, chronic kidney disease, and diabetes.

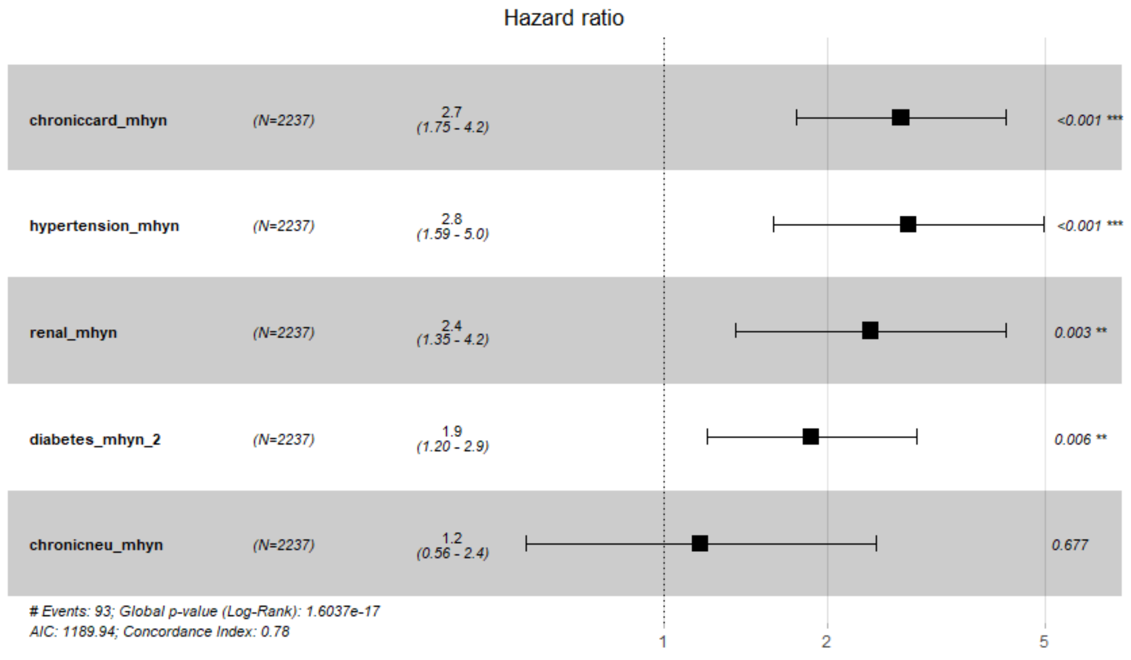


Figure 15 – Hazard Ratios of different risk factors

Interpreting the Cox model shown in Figure 15 is simple. Hazard ratios closer to 1 do not affect survival. Hazard ratios less than 1 are linked to improved survival, and those greater than 1 mean the predictor is associated with decreased survival (i.e. increased risk of death). From above, we can see that the hazard ratios for all the comorbidities are above 1, meaning an increased risk of death, with the "deadliest" comorbidities being chronic heart disease and hypertension, closely followed by chronic kidney disease and then diabetes. As for the neurological disorders, they seem to be less hazardous. This is further proven by their moderate significance with a p-value of 0.033.

5) Conclusion

This report looked at a sample of 2237 COVID-19 patients with various comorbidities. 93 of the patients have died. We applied four predictive modelling techniques to the data and compared their performance using various measures such as ROC, AUC, Sensitivity and Specificity. The Logistic Regression model performed quite well for the purpose of this analysis, closely followed by the Random Forest model. Both Artificial Neural Networks and Support Vector Machines underperformed in comparison to the other two.

Next, we generated Kaplan-Meier curves for different predictor variables and used the p-value to gauge their statistical significance. Out of the 6 comorbidities in the dataset, the 4 most influential to survival were chronic heart disease, hypertension, chronic kidney disease, and diabetes. Another risk factor was age – as patients over 50 years old were more likely to pass away from the virus. Cox Proportional Hazard model was applied to these risk factors to generate hazard ratios, which matched the results from the Kaplan-Meier curves.

Overall, age and chronic comorbidities were risk factors for mortality from COVID-19 infection. This result is in line with other international cohorts [4], including a UK ISARIC study [40] which used similar data.

Bibliography

1. WHO. About the virus [Internet]. www.euro.who.int. 2021. Available from: https://www.euro.who.int/en/health-topics/health-emergencies/coronavirus-covid-19/novel-coronavirus-2019-ncov

2. £18.5 million to tackle long COVID through research [Internet]. GOV.UK. 2021. Available from: https://www.gov.uk/government/news/185-million-to-tackle-long-covid-through-research

3. Lopez-Leon S, Wegman-Ostrosky T, Perelman C, Sepulveda R, Rebolledo PA, Cuapio A, et al. More than 50 long-term effects of COVID-19: a systematic review and meta-analysis. Scientific Reports [Internet]. 2021 Aug 9 [cited 2021 Aug 16];11(1):16144. Available from: https://www.nature.com/articles/s41598-021-95565-8

4. Munblit D, Nekliudov NA, Bugaeva P, Blyuss O, Kislova M, Listovskaya E, et al. Stop COVID Cohort: An Observational Study of 3480 Patients Admitted to the Sechenov University Hospital Network in Moscow City for Suspected Coronavirus Disease 2019 (COVID-19) Infection. Clinical Infectious Diseases. 2020 Oct 9;73(1):1–11.

5. Waljee AK, Higgins PDR, Singal AG. A Primer on Predictive Models. Clinical and Translational Gastroenterology. 2014 Jan;5(1):e44.

6. Vogenberg FR. Predictive and Prognostic Models: Implications for Healthcare Decision-Making in a Modern Recession. American Health & Drug Benefits [Internet]. 2009;2(6):218–22. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4106488/

7. Sperandei S. Understanding logistic regression analysis. Biochemia Medica. 2014;24(1):12–8.

8. Schober P, Vetter TR. Logistic Regression in Medical Research. Anesthesia and Analgesia [Internet]. 2021 Feb 1;132(2):365–6. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7785709/

9. Stoltzfus JC. Logistic Regression: A Brief Primer. Academic Emergency Medicine. 2011 Oct;18(10):1099–104.

10. W D, Lemeshow S, Sturdivant RX. Applied logistic regression [Internet]. New York, Etc.: John Wiley And Sons, Cop; 2013 [cited 2019 Jun 8]. Available from: https://www.wiley.com/en-us/Applied+Logistic+Regression%2C+3rd+Edition-p-9780470582473

11. Finlay S. Predictive analytics, data mining and big data : myths, misconceptions and methods. Basingstoke, Hampshire ; New York, Ny: Palgrave Macmillan; 2014.

12. Tokarev KE, Orlova YA, Rogachev AF, Rudenko AY, Protsyuk MP. Neural network application for predictive modeling. IOP Conference Series: Materials Science and Engineering. 2020 Aug 26;905:012069.

13. Jason Brownlee. A Gentle Introduction to the Rectified Linear Unit (ReLU) for Deep Learning Neural Networks [Internet]. Machine Learning Mastery. 2019. Available from: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

14. Labs M. Secret Sauce behind the beauty of Deep Learning: Beginners guide to Activation Functions [Internet]. Medium. 2019. Available from: https://towardsdatascience.com/secret-sauce-behind-the-beauty-of-deep-learning-beginners-guide-to-activation-functions-a8e23a57d046

15. Simeon Kostadinov. Understanding Backpropagation Algorithm [Internet]. Medium. Towards Data Science; 2019. Available from: https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

16. Pishro-Nik H. Mean Squared Error (MSE) [Internet]. www.probabilitycourse.com. [cited 2021 Sep 3]. Available from: https://www.probabilitycourse.com/chapter9/9_1_5_mean_squared_error_MSE.php

17. Brownlee J. A Gentle Introduction to Cross-Entropy for Machine Learning [Internet]. Machine Learning Mastery. 2019. Available from: https://machinelearningmastery.com/cross-entropy-for-machine-learning/

18. Nielsen MA. Neural Networks and Deep Learning [Internet]. Neuralnetworksanddeeplearning.com. Determination Press; 2019. Available from: http://neuralnetworksanddeeplearning.com/chap2.html

19. Akhtar M, Kraemer MUG, Gardner LM. A dynamic neural network model for predicting risk of Zika in real time. BMC Medicine. 2019 Sep 2;17(1).

20. Georgios Paliouras, Vangelis Karkaletsis, Spyropoulos CD. Machine learning and its applications : advanced lectures. Berlin ; New York: Springer; 2001.

21. Gandhi R. Support Vector Machine — Introduction to Machine Learning Algorithms [Internet]. Towards Data Science. Towards Data Science; 2018. Available from: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

22. Veropoulos K, Cristianini N, Campbell C. The Application of Support Vector Machines to Medical Decision Support: A Case Study. ;

23. Thissen U, van Brakel R, de Weijer AP, Melssen WJ, Buydens LMC. Using support vector machines for time series prediction. Chemometrics and Intelligent Laboratory Systems. 2003 Nov;69(1-2):35–49.

24. Svetnik V, Liaw A, Tong C, Culberson JC, Sheridan RP, Feuston BP. Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling. Journal of Chemical Information and Computer Sciences. 2003 Nov;43(6):1947–58.

25. Shipe ME, Deppen SA, Farjah F, Grogan EL. Developing prediction models for clinical use using logistic regression: an overview. Journal of Thoracic Disease. 2019 Mar;11(S4):S574–84.

26. Pencina MJ, D’Agostino RB, Demler OV. Novel metrics for evaluating improvement in discrimination: net reclassification and integrated discrimination improvement for normal variables and nested models. Statistics in Medicine. 2011 Dec 7;31(2):101–13.

27. Logistic Regression R | Introduction to Logistic Regression [Internet]. Analytics Vidhya. 2015 [cited 2021 Sep 3]. Available from: https://www.analyticsvidhya.com/blog/2015/11/beginners-guide-on-logistic-regression-in-r/#:~:text=log

28. Steyerberg EW, Moons KGM, van der Windt DA, Hayden JA, Perel P, Schroter S, et al. Prognosis Research Strategy (PROGRESS) 3: Prognostic Model Research. PLoS Medicine. 2013 Feb 5;10(2):e1001381.

29. Rich JT, Neely JG, Paniello RC, Voelker CCJ, Nussenbaum B, Wang EW. A practical guide to understanding Kaplan-Meier curves. Otolaryngology–Head and Neck Surgery. 2010 Sep;143(3):331–6.

30. Kaplan EL, Meier P. Nonparametric Estimation from Incomplete Observations. Journal of the American Statistical Association. 1958 Jun;53(282):457–81.

31. Kassambara A. Survival Analysis Basics - Easy Guides - Wiki - STHDA [Internet]. www.sthda.com. Available from: http://www.sthda.com/english/wiki/survival-analysis-basics#kaplan-meier-survival-estimate

32. Schütte D. Survival Analysis in R For Beginners [Internet]. DataCamp. 2019. Available from: https://www.datacamp.com/community/tutorials/survival-analysis-R

33. Kassambara A. Cox Proportional-Hazards Model - Easy Guides - Wiki - STHDA [Internet]. Sthda.com. 2019. Available from: http://www.sthda.com/english/wiki/cox-proportional-hazards-model

34. Borucka J. Extensions of Cox model for non-proportional hazards purpose. Ekonometria. 2014;(3(45)).

35. Cox DR. Regression Models and Life-Tables. Journal of the Royal Statistical Society Series B (Methodological) [Internet]. 1972 [cited 2021 Sep 3];34(2):187–220. Available from: https://www.jstor.org/stable/2985181

36. Karen Steward PhD. Sensitivity vs Specificity [Internet]. Analysis & Separations from Technology Networks. Technology Networks; 2019. Available from: https://www.technologynetworks.com/analysis/articles/sensitivity-vs-specificity-318222

37. Steen D. Understanding the ROC Curve and AUC [Internet]. Medium. 2020. Available from: https://towardsdatascience.com/understanding-the-roc-curve-and-auc-dd4f9a192ecb

38. Hajian-Tilaki K. Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation. Caspian journal of internal medicine [Internet]. 2013;4(2):627–35. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3755824/

39. Brownlee J. Confidence Intervals for Machine Learning [Internet]. Machine Learning Mastery. 2018. Available from: https://machinelearningmastery.com/confidence-intervals-for-machine-learning/

40. Swift A, Heale R, Twycross A. What are sensitivity and specificity? Evidence Based Nursing. 2019 Nov 12;23(1):ebnurs-2019-103225.

41. Docherty AB, Harrison EM, Green CA, Hardwick HE, Pius R, Norman L, et al. Features of 20 133 UK patients in hospital with covid-19 using the ISARIC WHO Clinical Characterisation Protocol: prospective observational cohort study. BMJ [Internet]. 2020 May 22;369. Available from: https://www.bmj.com/content/369/bmj.m1985

Appendices

Appendix 1: R code.

```
# Import all the needed libraries.#
library(pROC)
library(randomForest)
library(e1071)
library(caret)
library(survival)
library(survminer)
library(dplyr)


# Create new dataframes with desired inputs only.
C19A <- C19
C19B <- C19

# Remove C19 unwanted columns.
C19A$ID <- c()
C19A$Sex2 <- c()
C19A$Days <- c()
C19A$ICU <- c()

C19B$ID <- c()

# Splitting the data for predictive modelling.
set.seed(143)
ind <- sample(2, nrow(C19A), prob = c(0.5, 0.5), replace = TRUE)
train.data <- C19A[ind == 1,]
test.data <- C19A[ind == 2,]

#------- Logestic Regression -------

# Logistic Regression for training set.
training_logit <- glm(Death~., data=train.data, family="binomial")
summary(training_logit)

# Odds Ratios.
exp(training_logit$coefficients)
# Confidence intervals around ORs.
exp(confint(training_logit))

# Applying the Logistic Regression models to the test set.
LRmodel <- predict(training_logit, newdata=test.data,
type="response")

#------- Random Forest -------

# Random Forest for training set.
```

```r
RFmodel <- randomForest(factor(Death)~., data=train.data,
importance=TRUE, ntree=1000, mtry=3, replace=TRUE)
importance(RFmodel)[,c(3,4)]
rf_prediction <- predict(RFmodel, test.data, type = "prob")

#------- Support Vector Machines -------

SVMmodel <- svm(Death ~ ., data =train.data)
svm_prediction <- predict(SVMmodel, test.data)

# ------ ROC and AUC value to evaluate performance ------

# ROC
lrROC <- roc(test.data$Death, LRmodel)
plot(lrROC, col="red", main="ROC curves")

rfROC <- roc(test.data$Death, rf_prediction[,2])
plot(rfROC, add=TRUE, col = "green")

svmROC <- roc(test.data$Death, svm_prediction)
plot(svmROC, add=TRUE, col = "blue")

legend(0.4, 0.2, legend=c("Logistic Regression", "Random Forest",
"SVM", "ANN"),
       col=c("red", "green", "blue", "orange"), lty=1:1, cex=0.8)

# AUC
auc(lrROC)
auc(rfROC)
auc(svmROC)

# Confidence Intervals for each model.
ci.auc(lrROC)
ci.auc(rfROC)
ci.auc(svmROC)

# Sensitivities at fixed specificity.
ci.se(lrROC, specificities=c(0.9))
ci.se(rfROC, specificities=c(0.9))
ci.se(svmROC, specificities=c(0.9))


#------- Prognostic Analysis -------

# Histogram to split data by age.
hist(C19B$Age)
C19B <- C19B %>% mutate(age_group = ifelse(C19B$Age >=50, "Above
50", "Under 50"))
C19B$age_group <- factor(C19B$age_group)
```

```
# Fit survival data using the Kaplan-Meier method.
surv_object <- Surv(time = C19B$Days, event = C19B$Death)
surv_object

fit1 <- survfit(surv_object ~ age_group, data = C19B)
ggsurvplot(fit1, data = C19B, pval = TRUE)

fit2 <- survfit(surv_object ~ chroniccard_mhyn, data = C19B)
ggsurvplot(fit2, data = C19B, pval = TRUE)

fit3 <- survfit(surv_object ~ hypertension_mhyn, data = C19B)
ggsurvplot(fit3, data = C19B, pval = TRUE)

fit4 <- survfit(surv_object ~ chronicpul_mhyn, data = C19B)
ggsurvplot(fit4, data = C19B, pval = TRUE)

fit5 <- survfit(surv_object ~ renal_mhyn, data = C19B)
ggsurvplot(fit5, data = C19B, pval = TRUE)

fit6 <- survfit(surv_object ~ chronicneu_mhyn, data = C19B)
ggsurvplot(fit6, data = C19B, pval = TRUE)

fit7 <- survfit(surv_object ~ diabetes_mhyn_2, data = C19B)
ggsurvplot(fit7, data = C19B, pval = TRUE)

fit8 <- survfit(surv_object ~ Sex2, data = C19B)
ggsurvplot(fit8, data = C19B, pval = TRUE)


fit <- survfit(Surv(time = C19B$Days, event = C19B$Death) ~ Sex2,
data = C19B)
ggsurvplot(fit,
           pval = TRUE, conf.int = TRUE,
           risk.table = TRUE, # Add risk table
           risk.table.col = "strata", # Change risk table color by
groups
           linetype = "strata", # Change line type by groups
           ggtheme = theme_bw(), # Change ggplot2 theme
           palette = c("#E7B800", "#2E9FDF"))

#------- Cox Regression -------

fit.coxph <- coxph(surv_object ~ chroniccard_mhyn +
                     hypertension_mhyn + renal_mhyn +
diabetes_mhyn_2 + chronicneu_mhyn, data = C19B)
ggforest(fit.coxph, data = C19B)


#--------- End ---------
# Import all the needed libraries.#
library(pROC)
library(randomForest)
library(e1071)
library(caret)
library(survival)
library(survminer)
library(dplyr)


# Create new dataframes with desired inputs only.
C19A <- C19
C19B <- C19

# Remove C19 unwanted columns.
C19A$ID <- c()
C19A$Sex2 <- c()
C19A$Days <- c()
C19A$ICU <- c()

C19B$ID <- c()

# Splitting the data for predictive modelling.
set.seed(143)
ind <- sample(2, nrow(C19A), prob = c(0.5, 0.5), replace = TRUE)
train.data <- C19A[ind == 1,]
test.data <- C19A[ind == 2,]

#------- Logestic Regression -------

# Logistic Regression for training set.
training_logit <- glm(Death~., data=train.data, family="binomial")
summary(training_logit)

# Odds Ratios.
exp(training_logit$coefficients)
# Confidence intervals around ORs.
exp(confint(training_logit))

# Applying the Logistic Regression models to the test set.
LRmodel <- predict(training_logit, newdata=test.data,
type="response")

#------- Random Forest -------

# Random Forest for training set.
RFmodel <- randomForest(factor(Death)~., data=train.data,
importance=TRUE, ntree=1000, mtry=3, replace=TRUE)
importance(RFmodel)[,c(3,4)]
rf_prediction <- predict(RFmodel, test.data, type = "prob")

#------- Support Vector Machines -------
```

```
SVMmodel <- svm(Death ~ ., data =train.data)
svm_prediction <- predict(SVMmodel, test.data)

# ------ ROC and AUC value to evaluate performance ------

# ROC
lrROC <- roc(test.data$Death, LRmodel)
plot(lrROC, col="red", main="ROC curves")

rfROC <- roc(test.data$Death, rf_prediction[,2])
plot(rfROC, add=TRUE, col = "green")

svmROC <- roc(test.data$Death, svm_prediction)
plot(svmROC, add=TRUE, col = "blue")

legend(0.4, 0.2, legend=c("Logistic Regression", "Random Forest",
"SVM", "ANN"),
       col=c("red", "green", "blue", "orange"), lty=1:1, cex=0.8)

# AUC
auc(lrROC)
auc(rfROC)
auc(svmROC)

# Confidence Intervals for each model.
ci.auc(lrROC)
ci.auc(rfROC)
ci.auc(svmROC)

# Sensitivities at fixed specificity.
ci.se(lrROC, specificities=c(0.9))
ci.se(rfROC, specificities=c(0.9))
ci.se(svmROC, specificities=c(0.9))


#------- Prognostic Analysis -------

# Histogram to split data by age.
hist(C19B$Age)
C19B <- C19B %>% mutate(age_group = ifelse(C19B$Age >=50, "Above
50", "Under 50"))
C19B$age_group <- factor(C19B$age_group)


# Fit survival data using the Kaplan-Meier method.
surv_object <- Surv(time = C19B$Days, event = C19B$Death)
surv_object

fit1 <- survfit(surv_object ~ age_group, data = C19B)
ggsurvplot(fit1, data = C19B, pval = TRUE)
```

```
fit2 <- survfit(surv_object ~ chroniccard_mhyn, data = C19B)
ggsurvplot(fit2, data = C19B, pval = TRUE)

fit3 <- survfit(surv_object ~ hypertension_mhyn, data = C19B)
ggsurvplot(fit3, data = C19B, pval = TRUE)

fit4 <- survfit(surv_object ~ chronicpul_mhyn, data = C19B)
ggsurvplot(fit4, data = C19B, pval = TRUE)

fit5 <- survfit(surv_object ~ renal_mhyn, data = C19B)
ggsurvplot(fit5, data = C19B, pval = TRUE)

fit6 <- survfit(surv_object ~ chronicneu_mhyn, data = C19B)
ggsurvplot(fit6, data = C19B, pval = TRUE)

fit7 <- survfit(surv_object ~ diabetes_mhyn_2, data = C19B)
ggsurvplot(fit7, data = C19B, pval = TRUE)

fit8 <- survfit(surv_object ~ Sex2, data = C19B)
ggsurvplot(fit8, data = C19B, pval = TRUE)


fit <- survfit(Surv(time = C19B$Days, event = C19B$Death) ~ Sex2,
data = C19B)
ggsurvplot(fit,
           pval = TRUE, conf.int = TRUE,
           risk.table = TRUE, # Add risk table
           risk.table.col = "strata", # Change risk table color by
groups
           linetype = "strata", # Change line type by groups
           ggtheme = theme_bw(), # Change ggplot2 theme
           palette = c("#E7B800", "#2E9FDF"))

#------- Cox Regression -------

fit.coxph <- coxph(surv_object ~ chroniccard_mhyn +
                     hypertension_mhyn + renal_mhyn +
diabetes_mhyn_2 + chronicneu_mhyn, data = C19B)
ggforest(fit.coxph, data = C19B)


#--------- End ---------
```

Appendix 2: Python code

```
import os
import sys
import pandas as pd
import numpy as np
from keras.callbacks import ModelCheckpoint
from keras.models import  Sequential
from keras.layers import  Dense, Activation
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import Perceptron
from sklearn import preprocessing
import matplotlib.pyplot as plt
from scipy.stats import sem
from sklearn.metrics import roc_auc_score
import statistics
df =
pd.read_excel(r'C:\Users\warda\Desktop\wrd\Uni\MSc\Project\C19.xlsx'
)
df.head()
df.drop(['Sex2'], axis = 1, inplace = True)
df.drop(['ID'], axis = 1, inplace = True)
df.drop(['ICU'], axis = 1, inplace = True)
df.drop(['Days'], axis = 1, inplace = True)
df.head()
dataset = df.values
dataset
X = dataset[:,0:8]
Y = dataset[:,8]
min_max_scaler = preprocessing.MinMaxScaler()
X_scale = min_max_scaler.fit_transform(X)
X_scale
X_train, X_val_and_test, Y_train, Y_val_and_test =
train_test_split(X_scale, Y, test_size=0.5)
X_val, X_test, Y_val, Y_test = train_test_split(X_val_and_test,
Y_val_and_test, test_size=0.5)
print(X_train.shape, X_val.shape, X_test.shape, Y_train.shape,
Y_val.shape, Y_test.shape)
model = Sequential([
    Dense(32, activation='relu', input_shape=(8,)),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid'),
])
model.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])
hist = model.fit(X_train, Y_train,
          batch_size=32, epochs=2000,
          validation_data=(X_val, Y_val))
model.evaluate(X_test, Y_test)[1]
Y_pred = model.predict(X_test).ravel()
fpr, tpr, threshold = roc_curve(Y_test, Y_pred)
roc_auc = auc(fpr, tpr)
plt.rcParams["figure.figsize"] = (10.66, 10.91)
#plt.plot([-1, 0], [0, 1], color='grey', linestyle='-', alpha=0.3)
plt.plot(fpr-1, tpr, color="#e6550d",linewidth=2.2, label='Neural
Network' % roc_auc)
#plt.savefig("ANNROC.png", transparent=True, dpi=300)
roc_auc
n_bootstraps = 2000
rng_seed = 142  # control reproducibility
bootstrapped_scores = []

rng = np.random.RandomState(rng_seed)
for i in range(n_bootstraps):
    indices = rng.randint(0, len(Y_pred), len(Y_pred))
    if len(np.unique(Y_test[indices])) < 2:
        continue
    score = roc_auc_score(Y_test[indices], Y_pred[indices])
    bootstrapped_scores.append(score)
    print("Bootstrap #{} ROC area: {:0.3f}".format(i + 1, score))
    import matplotlib.pyplot as plt
plt.hist(bootstrapped_scores, bins=50)
plt.title('Histogram of the bootstrapped ROC AUC scores')
plt.show()
sorted_scores = np.array(bootstrapped_scores)
sorted_scores.sort()

confidence_lower = sorted_scores[int(0.05 * len(sorted_scores))]
confidence_upper = sorted_scores[int(0.95 * len(sorted_scores))]
print("Confidence interval : [{:0.3f} - {:0.3}]".format(
    confidence_lower, confidence_upper))
```