ACTL3142 Assignment Part 2

Tadhg Xu-Glassop - z5480859

2024T2

Contents

1 Executive Summary

This report utilises multiple facets of modelling through the consideration of statistical and machine learning methodologies to effectively assess the mortality and health risk of the Brazilian private health insurance portfolio. Findings will be focused on all-known characteristics to gain a deeper insight into such characteristics that impact the mortality of patients, as well as restricting ourselves to only data known upon admission, to gain an understanding and predict patients that face higher risk. Key methods, results and findings are described within the main report, and more technical details and results can be found within the technical appendix A.

2 Modelling Patient Characteristics and COVID-19 deaths

Given the numerous predictors in the dataset, we investigate the differing relationships present between predictors. This involves medical traits and symptoms patients may possess and the likelihood of them dying to COVID, and also the relationships that may exist between such predictors. In this section, we combine statistical and machine learning methodologies with domain knowledge of the dataset to create a model with the goal of *statistical inference* in mind, which will provide valuable insights into the relationship between patient characteristics and COVID deaths.

2.1 Constructing the Models

The nature of an inference problem allows several differing models to be considered viable options, with little foresight on what the best model may be. Thus, 3 models selected for their inferential properties were constructed using 80% of the dataset for training, with the remaining 20% held out for testing in section 2.2. This number of models was chosen as it allows a reasonably comprehensive analysis of how well different models model the data, without bearing the risk of *overfitting in model selection* with an excess of models[4]. Each model is briefly described here, and further details are provided in the appendix. Feature engineering was also performed prior to any modelling, adding the variable `daysInHosp` which is the number of days a patient spends in the hospital before dying or being discharged. Leverage points were discovered through this, with some patients having extremely long stays[1]. These observations were removed, so the models could better reflect the vast majority of patients.

Model 1: A logistic regression model with initially all predictors and interaction terms between `oxygensat` and `respdistress`, `dyspnoea` and `respdistress` and `diarrhea` and `vomit`. Such interaction terms were found by considering correlation coefficients between predictors in tandem with domain knowledge, then a comparison between models with different interaction terms with 10-fold CV was done[2]. This member of the GLM family was chosen as it is a classification model, and the coefficients of the predictors including the interaction terms allow for easy interpretation. LASSO regularisation with λ_{1SE} was also applied, as the *selective property* of LASSO regularisation and the extra penalty of this tuning parameter can provide insights regarding noise parameters[3].

Model 2: A Generalised Additive Model (GAM) with natural spline terms for `age`, `dateHosp` and `daysInHosp`, with degrees of freedom 2, 3 and 4 respectively. These parameters were chosen by performing 10-fold CV with the different degree spline terms for each numerical predictor, selecting the best one with consideration of the *bias-variance tradeoff*[4]. All other predictors were initially included as linear terms and the interaction terms from Model 1. A natural spline was used over a standard regression spline, as a smooth visual representation will be easier to interpret. To eliminate parameters, *analysis of deviance* was used to remove `cardio` and the `dyspnoea * oxygensat` interaction term. Unlike Model 1, LASSO was not used, as LASSO can behave cause problems when there are correlated variables, as is the case with spline predictors[6]. Other forms of regularisation, such as ridge, were not considered, as these do not have selective properties and we seek knowledge of the noise variables in the dataset.

Model 3: A single classification tree derived with Cost-Complexity Pruning on a full tree to simplify the full tree down into a more interpretable model without sacrificing much accuracy[5]. A single tree was chosen over ensemble methods including bagging and boosting as these methods sacrifice interpretability for predictive performance and will thus be deferred to section 3.

2.2 Comparing the Models

With 3 unique models that are all suitable for the purpose of inference, we select the best one by considering direct methods of comparison.

First, 10-fold cross validation with classification error and F1 score was used as a direct measurement of predictive performance for all 3 models. A threshold of 0.5 was used for error, while a threshold of 0.3 was used for F1 score for Models 1 and 2.[6] The results are summarised in table 1. We see that Model 2 has the lowest error and the highest F1 score, showing better performance than the other models.

Now, using the the held out 20% of the data mentioned in section 2.1, we can test the performance of models 1 and 2 at different thresholds with a Receiver Operating Characteristic (ROC) curve and compare the Area Under Curve (AUC). We find that the AUC of model 1 is 0.796, while the AUC of

[1]See A.1 for details and visualisations.

[2]See A.2 for details on interaction terms.

[3]See A.4 for details on model 1 construction.

[4]See A.5 for specific details on model construction and parameter selection for model 2

[5]See A.6 for a visualisation of this model, as well as construction details.

[6]This value was chosen to maximise F1 score for these models. See A.3 for more details.

Model	CV Classification Error	CV F1 Score
1	0.2678345	0.6880765
2	0.2588910	0.6974229
3	0.2653728	0.6538808

Table 1: 10-Fold Cross Validation Classification Error on the 3 Proposed Models, using the same folds for each test.

model 2 is 0.81, meaning model 2 performs better than model 1 on unseen data across different thresholds[7]. Further, table 2 shows the test set error for each model, where a threshold of 0.5 was taken for models 1 and 2. Clearly, model 2 has the best result, showing it performs the best on unseen data.

Model	Test Set Error
1	0.2680894
2	0.2594383
3	0.2620782

Table 2: Test Set Error with the unseen 20% of data.

Thus, out of the three candidate inferential models, we have strong evidence that model 2 - a GAM with natural spline predictors and some interaction terms, outperforms a logistic regression with LASSO regularisation and a tree model on the training set, and performs well on unseen data. With this in mind, we ensure we can safely make inferences with this model by checking its assumptions are satisfied.

2.3 Assumptions of the Best Model

Now, before we make inferences with this model, we first address the assumptions we must make to use this model. Failure to address false assumptions can lead to biased and false inferences, and so for our purpose it is essential we do this.

Of course, we assumed that the predictors `age`, `dateHosp` and `daysInHosp` have a relationship to the log-odds of `covidDeath` that can be modelled by a spline. To justify this assumption, we plot the residuals of fitted and observed values of the spline predictors to visualise the goodness of fit for the splines, as seen in figure 1. A fitted smooth curve (blue) has been added, with a red dashed-line representing $y = 0$, which would indicate a perfect fit. We see that for each of the spline predictors, the line of best fit slightly deviates from $y = 0$, and the residuals are distributed in a way that has a predictable trend, which has an approximate average of 0. Hence, we have that each of the spline predictors is approximately systematically unbiased, so this assumption is justified.

A key assumption of the GAM in general is that all observations must be independent of each other. Intuitively, this is not the case for COVID cases and deaths, as cases can be dependent on each other especially if there is a widespread outbreak at the time and hospital capacity may be at a limit,

[7]AUC was not calculated for model 3 as such a statistic is unrelated for a tree model. See A.7 for more details.

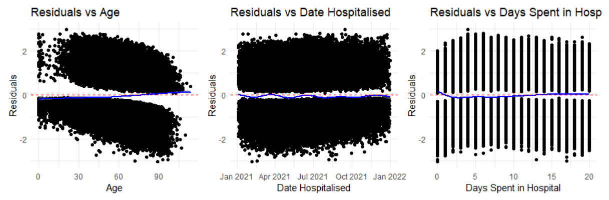

Figure 1: Residuals of Fitted v.s. Observed for (from left to right) `age`, `dateHosp` and `daysInHosp`.

impacting the likelihood of someone surviving. To address this, we have included a flexible predictor for `dateHosp`, which was shown to be statistically significant and fit the data well above. With this predictor, our model addresses the temporal trends, and helps justify this assumption. This assumption also implies that there cannot be any repeated measurements. This is satisfied, as our dataset contains a unique `Patient_id` for each observation and so we know we each observation is unique.

Further, we also must assume there is no multicollinearity amongst the independent variables, that is, each predictor provides unique information about `covidDeath`. This is especially important for our model, as interaction terms are present. We can check this assumption is satisfied by calculating the Variance Inflation Factor (VIF) of each predictor[8] This shows that all but 2 predictors have a VIF between 1 and 2, with the two highest VIF predictors `respdistress` and its interaction term with `oxygensat` being 2.26 and 2.63 respectively. Taking the common *Rule of 10* for VIF tolerance[15][2], all our VIF values are tolerable, and thus we can assume there is no major multicollinearity amongst our predictors, satisfying this assumption.

2.4 Insights from the Best Model

After proving model 2 is a better fit than other inferential models and its assumptions are satisfied, we can use this model to gain insights into the relationship between patient characteristics and COVID deaths.

First, we investigate the main feature of this model - the spline terms. Interpreting the coefficients is difficult, so we plot the splines and the contribution of log-odds it gives for different predictor values. Plotting the spline modelling `dateHosp`, seen in the left panel of figure 2, we see an increase rapidly from the base date to a peak around May 2021. This then decreases to a value lower than the intercept at October 2021. Interpreting this in tandem with the number of daily admissions and deaths, visualised in the right panel, this sharp increase represents the major outbreak, where Brazil recorded the highest number of COVID cases in the world at the time[19]. The increase in the log-odds of death up to 0.6 for patients admitted at this time quantifies the fact that hospitals were overcrowded, with all ICU units between 90% to 100% capacity, and effective care could not be given to all patients[18]. Then, the following decrease in log-odds of death

[8]See A.9 for the VIF of each predictor.

aligns with the decrease of admissions, and the inference that patients admitted in Jan 2022 had 0.1 lower log-odds of dying than patients in Jan 2021 quantifies the phenomenon of *herd immunity*, wherein the Brazilian population was collectively more immune to COVID as a result of vaccinations and immunity from previous infection[12][21].

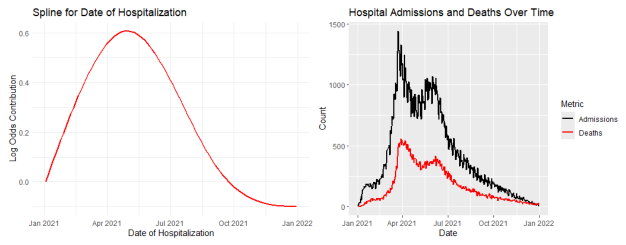

Figure 2: Left: Natural spline describing log-odds of `covidDeath` for date of hospitalisation. Right: Daily hospital admissions and deaths throughout the dataset.

For the remaining two spline predictors `age` and `daysInHosp`, we similarly interpret them by plotting the value of the spline at different values of the predictor. Figure 3 shows the two resulting splines. In the `daysInHosp` spline, we notice a minimum at 11 days, which corresponds to the mean hospitalisation tenure of survivors. Thus, stays shorter or longer than this are associated with higher log odds of death. This is can be explained by realising that very short hospitalization periods of non-survivors is a result of patients being admitted too late, when their condition is already critical by the time they receive medical attention. In addition, longer hospital stays are associated with patients with underlying health issues that impact their COVID mortality, explaining the increase in log-odds after 11 day hospital stays[1]. Now, the `age` spline conveys the well-known fact that those most vulnerable to COVID are elder generations[11]. We see a non-linear increasing relationship between log-odds and age, with the gradient becoming steeper for higher ages, representing the higher risk for elder generations.

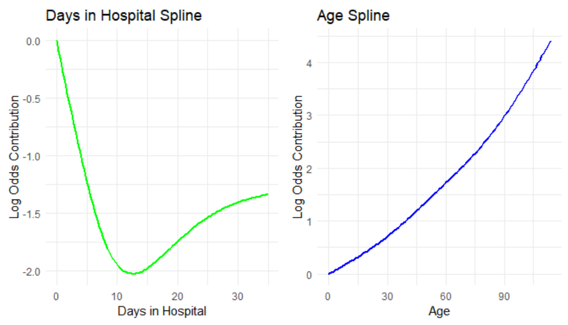

Figure 3: Left: Natural spline describing the added log-odds of `covidDeath` for each day spent in hospital. Right: Similar, but for integer age.

Turning our attention to the two interaction terms that were included in the model, we infer some relationships between predictors and how they impact COVID deaths[9]. The coefficient of `oxygensat * respdistress` being 0.22218, in tandem with the coefficients of `oxygensat` and `respdistress` being 0.14845 and 0.12814 respectively, shows a severe interaction between the two characteristics. That is, a patient with only a respiratory distress syndrome or low blood oxygen levels has a 0.5370445[10] or 0.5319912 higher probability of dying than a patient with neither respectively, not considering other predictors. However, a patient with both has a significant 0.6221702 higher probability of dying relative to someone with neither, quantifying the interaction these two predictors have with relation to `covidDeath`. In addition, the coefficients of `diarrhea` and `vomit` are -0.06096 and -0.01676 respectively, with the coefficient of the interaction term `diarrhea * vomit` being 0.06683. This means a patient with both diarrhea and vomiting symptoms has higher probability of dying than a patient with only one of those symptoms. This is likely a result of cases of COVID where only one of these symptoms are present are less serious than cases where both of these symptoms are present, providing more insights into the relationships and interactions between predictors.

Onto the significance of individual symptoms, we recall that analysis of deviance showed `cardio` was insignificant in the presence of other predictors, and was hence removed. Interpreting this, `cardio` is a *noise predictor*, wherein it provides no valuable information about whether a patient will have a higher chance of dying to COVID in tandem with the other predictors, that is, patients with cardiovascular disease do not appear to have higher mortality than an identical patient without such a disease. Now, after retraining our model on scaled data and comparing the coefficients[11], we gain an understanding of which predictors have the most impact. The highest predictor is `icuTRUE`, with a coefficient of 2.093058655, which represents a 0.31 increase in the probability of dying for a patient admitted to the ICU, ignoring other predictors and considering the intercept of -2.6. In comparison, the second most significant predictor is `immunoTRUE`, with a coefficient of 0.711779747 representing a 0.063 increase in probability of death. On the other hand, the lowest predictor coefficient is `vaccineTRUE`, which is -0.391988764. This represents a 0.0218 decrease in the probability of death for a vaccinated patient relative to an unvaccinated patient, without consideration of other predictors. Thus, we have found patients admitted to the ICU have a significantly higher likelihood of dying, while vaccinated patients have a slightly lower probability. Further, we have found that cardiovascular disease does not impact patient mortality.

[9]See A.8 for model summary and coefficient estimates.

[10]Coefficients represent log-odds as implied by using the canonical link for logistic regression. So, change in probabilities are found by applying the inverse logit function on the coefficients with consideration of the intercept.

[11]See A.10 for the process used and the output.

3 Predicting at High-risk Individuals

So far, we have been modelling `covidDeath` by making use of all available predictors. From here, we will restrict ourselves to just the variables known upon admission, and strive to make a predictive model. This will aid medical practitioners by identifying high-risk individuals that require urgent care. In this section, the F-score is the predominant cost function in comparing models. This is because the metric balances precision and recall, which is particularly useful when there is less `TRUE` cases than `FALSE`, as is the case with the dataset, and if we care more about correctly identifying `TRUE` cases[16]. This aligns with our goal of identifying high-risk patients, and thus is an appropriate metric for our problem.

3.1 Developing Predictive Models

Similar to section 2, we consider multiple predictive models from different model classes and machine learning methodologies. Then, we compare them on the test set, and take the best performing one.

Model 4: GAM with Elastic Net Regularisation. A modified version of Model 2 (the best model from section 2), using only known data upon admission. Interaction terms were kept from the original model, with `cardio` added back, and then elastic net regularisation was performed to shrink down parameters and reduce the impact of noise parameters. The tuning parameters α (balance of ℓ_1 and ℓ_2 penalties) and λ (weight of overall penalty) were found by performing 10-fold CV with F-score cost function on different values of λ for different values of α. Then, the parameter combination that maximised F-score, being $\alpha = 0.008$ and $\lambda = 0.004367$ were picked, giving Model 4[12]. This regularisation method was chosen as it allows hyper-tuning of parameters to maxmise F-score. Although this level of λ and α tends not to have selective properties, it is fine for our purpose of maximising predictive performance.

Model 5: GAM with Non-Parametric Predictors, Elastic Net. We construct another GAM with non-parametric methods applied to the continuous predictors. Taking in the findings from section 2.4, we seek to capture the temporal nature of the `dateHosp` predictor with LOESS (local regression) for increased predictive performance. The common tricube function was used as the weight function, and the optimal span s was found by performing 10-fold CV with F-score on identical GAM's using all known predictors except for differing span parameters for the local regression on `dateHosp`. We find that the the span that maximises F-score is $s = 0.2$. We then apply a smoothing spline to `age`, applying an identical method to find the smoothing parameter λ_{smooth}, yielding a value of $\lambda_{\text{smooth}} = 0.5$ which maximises F-score[13]. Then, all predictors and interaction terms from model 1 were added, and elastic net regularisation in a manner identical to Model 4 was applied, yielding parameters $\alpha = 0.001$ and $\lambda_{\text{net}} = 0.01161$[14].

Model 6: Gradient-Boosted Tree Model. To improve upon the performance of Model 3 from section 2.1, we will use a modified boosting ensemble method known as *gradient boosting*, where rather than sequentially training on residuals like normal boosting, we train on pseudo-residuals by using a differentiable cost function[15]. Such methods have extensive parameters, so hyper-tuning such a model can be very tedious. To address this, we use a racing method, where we specify a grid of potential parameter values[16] and randomly create n different models[17]. Then, we perform 10-fold CV with F-score cost penalty, and eliminate models that are significantly under-performing with respect to other models after successive folds, hence why it is called a racing method. Then, we select the final model that has not been eliminated. The benefit of this methodology is that we can cross-validate the performance of hundreds of models simultaneously and find an optimal model without having to re-tune the model repeatedly. Doing this process with $n = 1000$, we reach Model 6. The race visualisation and final parameters are detailed in A.16.

Model 7: Random Forest. Another way of improving Model 3 is to implement a random forest, an ensemble method that involves implementing multiple trees training on bootstrapped samples with m predictors, where $m < p$. For the parameters, we take the common $m = \sqrt{p}$, and for the number of trees and minimum node size, we again implement a racing method with 10-fold CV, an F-score cost penalty and $n = 100$ models[17]. We find the optimal number of trees is 813 and the minimum node size is 9, which formulates Model 7.

3.2 Comparing Predictive Models.

With a test set available, we train the models on all available data and perform a test, measuring the F-score of each model. The results are summarised in table 3. Clearly, the tree methods are outclassed by the GAM's. We also see that Model 5 has a higher test score than Model 4, likely due to the fact that it uses non-parametric methods to fit the data better.

Model	Test F1-Score
4	0.60667
5	0.61341
6	0.58119
7	0.50117

Table 3: Test F1-Score for all candidate predictive models.

From this, we select Model 4 to be our best predictive model, and thus its predictions to be the most reliable.

[12]See A.11 for details on parameter tuning.
[13]See A.12 for the results of these CV's.

[14]See A.13 for visualisation and details.
[15]See A.14 for a technical overview of this methodology.
[16]See A.15 for details on this method.
[17]See A.17 for race results and details.

3.3 Insights into High-Risk Patients

We now take advantage of Model 4's predictive power and analyse its predictions to gain insights of the characteristics known upon admission of the patients that are more likely to result in death from COVID.

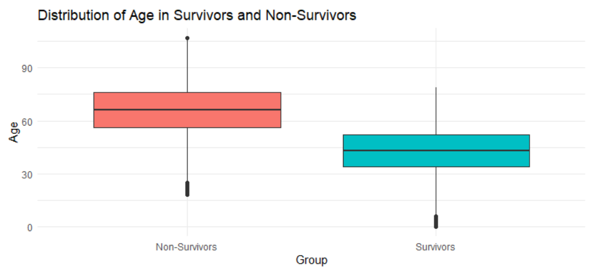

Figure 4: Box Plot for `age` of survivors and non-survivors.

Figure 4 illustrates the distribution of the age of survivors and non-survivors. Clearly, non-survivors are associated with higher age, with the 1st and 3rd quartiles of the age of non-survivors being 56 and 76 respectively, while these statistics are 34 and 52 for survivors. This shows that the majority of the survivors and non-survivors can be partitioned into disjoint age groups based on these quartiles, illustrating which specific age demographics are at the highest risk.

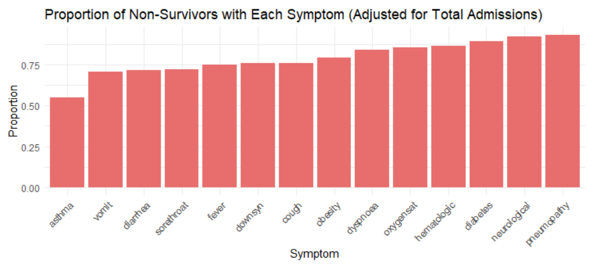

Figure 5: Ordered adjusted proportions of symptoms for non-survivors.

By adjusting the proportions of non-survivors with each symptom through considering the total proportion of patients with each symptom, we can quantify the severity of each symptom. Figure 5 shows these adjusted proportions in order. Interpreting this, pneumopathy is the most significant predictor, with 93.4% of patients admitted with the condition predicted to die. This is contrasted by the least signiciant predictor asthma, with 53.2% of asthmatic patients expected to die. Recalling results from section 2.1 and A.2, `pneumopathy` is highly correlated to the `oxygensat` predictor, which is not known upon admission but from our findings in section 2.4, can significantly increase a patient's mortality risk. Further, patients with neurological diseases such as Parkinson's disease, and diabetes are at very high risk, with 92% and 89.1% of such patients predicted to die respectively, which coincides with numerous studies on COVID mortality[14][20].

We can investigate the predicted monthly mortality rate, visualised in figure 6. The change in monthly mortality rate in our model's predictions from a peak in March 2021 to a low in 2022 corresponds with our inferences in section 2.4, wherein couldn't receive necessary care as a result of the influx of admissions due to the pandemic, and also the impact the vaccine had on the mortality of the population.

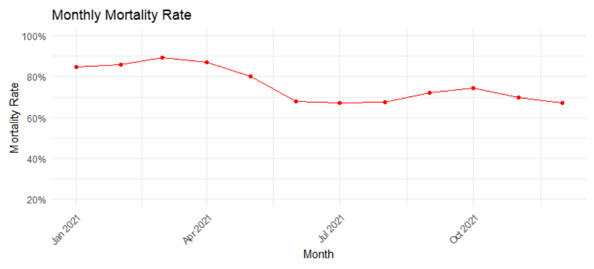

Figure 6: Predicted monthly mortality rate.

Further investigating the effect of the vaccine, we find that 41.4% of survivors were vaccinated, while 39.3% of non-survivors were vaccinated. Performing a Two Proportion Z-Test on these statistics, we get a p value of 0.0019, providing very strong evidence that the vaccine is effective in reducing COVID mortality for admitted patients[18]. In addition, considering that throughout the timeframe of the dataset, approximately 75% of the Brazilian proportion was vaccinated[13], in tandem with the fact that only 39.8% of admitted patients in the test set were vaccinated, provides very strong evidence that the vaccine is effective in preventing people being admitted to the hospital in the first place. Thus, we have found that not only is the vaccine significant in that it reduces the mortality of admitted patients, but it also reduces the hospitalization rate significantly.

4 Conclusion

Throughout this report, we have explored the `CovidHospDataBrasil.csv` dataset with inferential and predictive models to uncover powerful insights into the relationship between patient characteristics and COVID-19 deaths. From our findings, we discovered that the most vulnerable patients are those with pneumopathy diseases, as not only do they increase risk significantly, but are correlated with further significant characteristics that are hard to detect upon admission. Additional significant characteristics include age, especially those in the $56 - 76$ demographic, and those with neurological conditions. Further, we have found strong evidence for the effectiveness of the vaccine in preventing hospitalizations and reducing the mortality of patients, and the significance of *when* a patient is admitted and how it impacts their mortality. These insights can be used to identify and provide care for high-risk patients purely off their profile upon admission, and also illustrates the importance of increasing hospital resources and vaccinations to reduce mortality.

[18]See A.18 for details on hypothesis test used.

A Technical Appendix and AI Usage

A.1 Feature Engineering and Removing Outliers

The predictor `daysInHosp` was added to gain further information about patients. It was added as it is easy to interpret, and it does not have major collinearity with other predictors, as seen in A.9. However, it does reveal interesting information about the hospital stay of patients. Figure 7 shows the distribution of *all* patients. We see that the majority of patients have stays less than 20 days, but a relatively small number of patients have stays for up to 300 days, which massively influences our data.

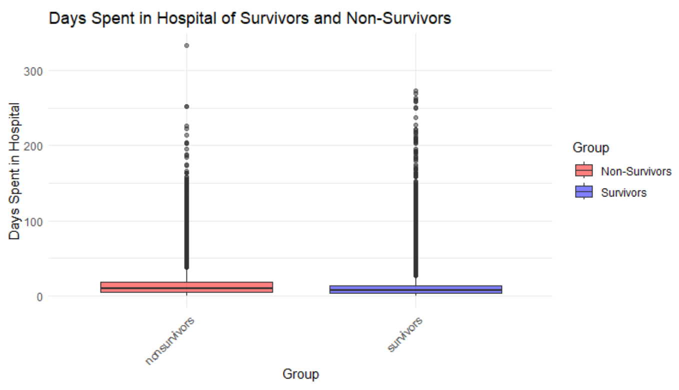

Figure 7: Box Plot for `daysInHosp` for survivors and non-survivors.

We remove the majority of these outliers, specifically all patient stays longer than 40 days. Figure 8 shows the same graph, but just with this data. It is now much more interpretable, and so we base our model on this, as it reflects the majority of the patient stays in the dataset.

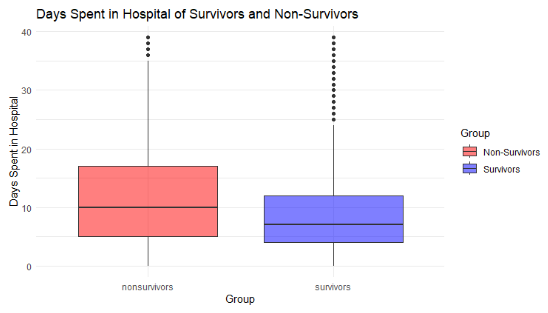

Figure 8: Box plot for `daysInHosp < 40` for survivors and non-survivors.

A.2 Finding Interaction Terms

Interaction terms were sought for to capture the compounding effects symptoms are known to have each other, and relax the assumption that the contribution of each predictor is independent for better modelling[7]. To do this, possible interactions were first postulated by considering the correlation matrix of all predictors in the dataset, visualised in figure 9.

We find that the most significant correlations are between `dyspnoea` and `respdistress`, `oxygensat` and `respdistress`, `dyspnoea` and `oxygensat`, and `diarrhea` and `vomit`. Considering our domain knowledge, these correlations make sense and are easily explained; studies show that respiratory diseases, represented by `respdistress` and signified by the `dyspnoea` symptom, are signals for dangerously low blood-oxygen levels[9]. Further, patients who exhibit symptoms of diarrhea and vomit are usually patients experiencing more extreme cases of COVID, and such symptoms are a result of a bodily reaction to not only the disease. As these predictors seem rational for interaction terms, we seek to directly assess the influence they can have on a model.

To do this, we construct several models experimenting with interaction terms with the most significant terms, `respdistress`, `oxygensat` and `dyspnoea`. We also create a control model, with no interaction terms, to see if the interaction terms pose an improvement. The following code block details each model.

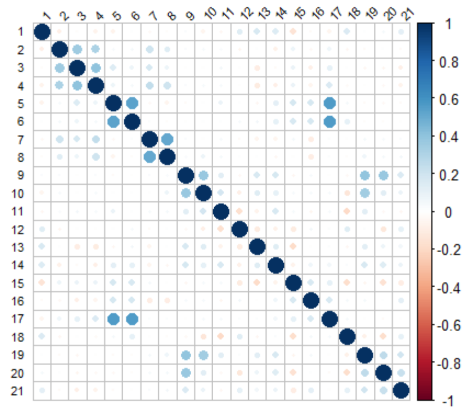

Figure 9: Correlation matrix of all medical predictors.

```
# control model with no interactions
control_model <- glm(
  covidDeath ~ dateBirth + sex + vaccine + dateHosp + fever + cough +
    sorethroat + dyspnoea + oxygensat + diarrhea + vomit + hematologic +
    downsyn + asthma + diabetes + neurological + pneumopathy + obesity +
    icu + respdistress + cardio + hepatic + immuno + renal + daysInHosp,
  data = train,
  family = binomial
)

# interaction between dyspnoea * oxygensat
inter_model1 <- glm(
  covidDeath ~ dateBirth + sex + vaccine + dateHosp + fever + cough +
    sorethroat + dyspnoea + oxygensat + diarrhea + vomit + hematologic +
    downsyn + asthma + diabetes + neurological + pneumopathy + obesity +
    icu + respdistress + cardio + hepatic + immuno + renal + daysInHosp +
    dyspnoea * oxygensat,
  data = train,
  family = binomial
)


# interaction between dyspnoea * respdistress, oxygensat * respdistress,
# dyspnoea * oxygensat
inter_model2 <- glm(
  covidDeath ~ dateBirth + sex + vaccine + dateHosp + fever + cough +
    sorethroat + dyspnoea + oxygensat + diarrhea + vomit + hematologic +
    downsyn + asthma + diabetes + neurological + pneumopathy + obesity +
    icu + respdistress + cardio + hepatic + immuno + renal + daysInHosp +
    dyspnoea * respdistress + oxygensat * respdistress + dyspnoea * oxygensat,
  data = train,
  family = binomial
)

# interaction between oxygensat * respdistress, dyspnoea * respdistress
inter_model3 <- glm(
  covidDeath ~ dateBirth + sex + vaccine + dateHosp + fever + cough +
    sorethroat + dyspnoea + oxygensat + diarrhea + vomit + hematologic +
    downsyn + asthma + diabetes + neurological + pneumopathy + obesity +
    icu + respdistress + cardio + hepatic + immuno + renal + daysInHosp +
    oxygensat * respdistress + dyspnoea * respdistress,
  data = train,
  family = binomial
)

# interaction between oxygensat * respdistress
inter_model4 <- glm(
  covidDeath ~ dateBirth + sex + vaccine + dateHosp + fever + cough +
    sorethroat + dyspnoea + oxygensat + diarrhea + vomit + hematologic +
    downsyn + asthma + diabetes + neurological + pneumopathy + obesity +
    icu + respdistress + cardio + hepatic + immuno + renal + daysInHosp +
    oxygensat * respdistress,
  data = train,
  family = binomial
)

# interaction between dyspnoea * respdistress
inter_model5 <- glm(
  covidDeath ~ dateBirth + sex + vaccine + dateHosp + fever + cough +
    sorethroat + dyspnoea + oxygensat + diarrhea + vomit + hematologic +
    downsyn + asthma + diabetes + neurological + pneumopathy + obesity +
    icu + respdistress + cardio + hepatic + immuno + renal + daysInHosp +
    dyspnoea * respdistress,
  data = train,
  family = binomial
)
```

Now, performing 10-fold cross validation with classification error on each model, we can compare the performance they offer. Table 4 summarises the results; model 0 is the control model with no interaction terms, and each subsequent model detailed in the above code block. We see model 3, which has an interaction term between `oxygensat` and `respdistress` as well as `dyspnoea` and `respdistress` has a decent improvement over the control model and the best classification error overall, so we select it moving forward.

Model	10-Fold CV Classification Error
0	0.2659977
1	0.2658307
2	0.2657592
3	0.2654888
4	0.2658228
5	0.2659580

Table 4: CV Error for all candidate interaction term models (1).

Further, we consider another interaction term between `vomit` and `diarrhea`. We repeat this process, reshuffling the train and test data to prevent overfitting, and we consider the control model, model 3 from above, model 3 with the added interaction term, and a model with just the interaction term. The following code block again details each model compared.

```
# control model with no interactions
control_model <- glm(
  covidDeath ~ dateBirth + sex + vaccine + dateHosp + fever + cough +
    sorethroat + dyspnoea + oxygensat + diarrhea + vomit + hematologic +
    downsyn + asthma + diabetes + neurological + pneumopathy + obesity +
    icu + respdistress + cardio + hepatic + immuno + renal + daysInHosp,
  data = train,
  family = binomial
)

# interaction between oxygensat * respdistress, dyspnoea * respdistress
inter_model1 <- glm(
  covidDeath ~ dateBirth + sex + vaccine + dateHosp + fever + cough +
    sorethroat + dyspnoea + oxygensat + diarrhea + vomit + hematologic +
    downsyn + asthma + diabetes + neurological + pneumopathy + obesity +
    icu + respdistress + cardio + hepatic + immuno + renal + daysInHosp +
    oxygensat * respdistress + dyspnoea * respdistress,
  data = train,
  family = binomial
)

# interaction between oxygensat * respdistress, dyspnoea * respdistress
# diarrhea * vomit
inter_model2 <- glm(
  covidDeath ~ dateBirth + sex + vaccine + dateHosp + fever + cough +
    sorethroat + dyspnoea + oxygensat + diarrhea + vomit + hematologic +
    downsyn + asthma + diabetes + neurological + pneumopathy + obesity +
    icu + respdistress + cardio + hepatic + immuno + renal + daysInHosp +
    oxygensat * respdistress + dyspnoea * respdistress + diarrhea * vomit,
  data = train,
  family = binomial
)

# interaction between diarrhea * vomit
inter_model3 <- glm(
  covidDeath ~ dateBirth + sex + vaccine + dateHosp + fever + cough +
    sorethroat + dyspnoea + oxygensat + diarrhea + vomit + hematologic +
    downsyn + asthma + diabetes + neurological + pneumopathy + obesity +
    icu + respdistress + cardio + hepatic + immuno + renal + daysInHosp +
    diarrhea * vomit,
  data = train,
  family = binomial
)
```

The results of another 10-fold CV with classification error on these new models is detailed in table 5; again model 0 is the control model. We see that model 3 from the previous CV experiment outperforms the control model again, and that adding an interaction term between `diarrhea` and `vomit` results in a lesser CV error.

Model	10-Fold CV Classification Error
0	0.2663078
1	0.2662817
2	0.2662442
3	0.2662601

Table 5: CV Error for all candidate interaction term models (2).

Hence, taking all of the above into account, we consider the interaction terms between `oxygensat` and `respdistress`, `dyspnoea` and `respdistress` as well as `diarrhea` and `vomit` valuable in providing further information about the morality of patients, and thus effective as added interaction terms in models.

A.3 Optimal F1-Score Threshold

Throughout the report, Models 1, 2, 4 and 5 had an altered threshold, unique to the common 0.5 for binary predictions. This parameter was altered when testing for F-score as the F-score aims to maximise accuracy and recall, so having a higher false positive rate is more desirable than a higher false negative rate. For this reason, we expect the threshold to be lower. To find the optimal threshold, we, train the model on the training set, and then make predictions on the test set. Then, we find the F1-score associated for different thresholds. Figure 10 visualises the results for this process for Model 2. We see that it maximises around 0.31. Repeating this process for other models, we get similar results that the threshold that maximises F1 score is around 0.3. So, for simplicity, we take a threshold of 0.3 in section 2.2 and 3.2.

A.4 Constructing Model 1

Model 1 is a logistic regression model, using all the significant interaction terms found in A.2 and all available predictors, with LASSO regularisation applied. As mentioned in section 2.1, LASSO regularisation was applied over other forms of regularisation as its selective properties make for easier interpretation of noise-parameters, which suits the goal of model 1 as an inferential model. The following code is what was used to construct model 1.

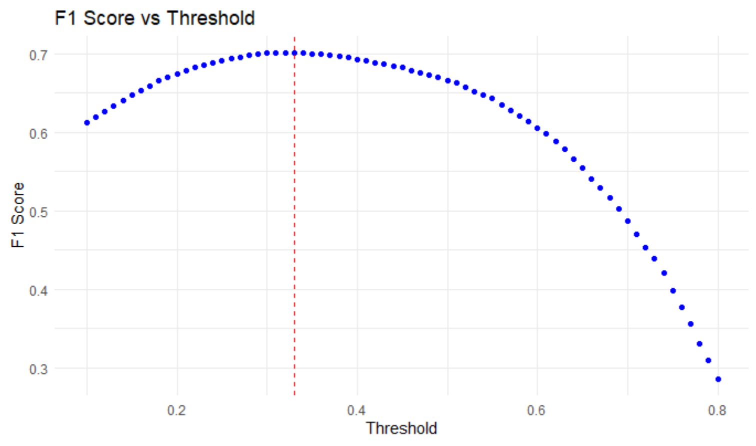

Figure 10: F1-score on the test set for different thresholds for Model 2.

```
data_1 <- model.matrix(
  covidDeath ~ dateBirth + sex + vaccine + dateHosp + fever + cough +
    sorethroat + dyspnoea + oxygensat + diarrhea + vomit + hematologic +
    downsyn + asthma + diabetes + neurological + pneumopathy + obesity +
    icu + respdistress + cardio + hepatic + immuno + renal + daysInHosp +
    oxygensat * respdistress + dyspnoea * respdistress + diarrhea * vomit,
  data = train,
  family = binomial()
)

x <- data.matrix(data_1)

y <- train$covidDeath

lambda_1 <- cv.glmnet(x, y, alpha = 1, family = binomial())

plot(lambda_1)

model1 <- glmnet(
  x, y, alpha = 1, lambda = lambda_1$lambda.1se, family = binomial
)
```

In addition, figure 11 shows the different CV GLM residuals for different values of lambda. λ_{min} is signified, as well as λ_{1SE}. As mentioned, λ_{1SE} was selected, as using a higher penalty will shrink more terms and thus eliminate more noise parameters, and from the plot this decision is justified as clearly we are not taking on a significant amount of deviance for this extra benefit.

A.5 Constructing Model 2

Model 2 assumes that the continuous predictors have a relationship to the log-odds of the predictor that can be modelled with a spline. Of course, this begs the question of what degrees of freedom to use for each spline. To find the optimal parameter, for each continuous predictor, we train several near identical models, considering all linear predictors, except varying the degrees of freedom for the natural spline of the continuous predictor in question. Figure 12 shows the CV classification error for each model. From the plots, we select degrees of freedom 2, 3, and 4 for `age`, `dateHosp` and `daysInHosp` respectively (note from the graphs, it is not obvious, but degree 2 offers good improvement for `age`, and such is the case as well for `dateHosp`). Such degrees were picked, as they get a small value of classification error without taking on too much flexibility. This justification is with the bias-variance tradeoff in mind, as we do not want to reduce the bias of our model so much that it causes significant variance.

After this, analysis of deviance was used to remove predictors, removing significant predictors one-by-one until all predictors were significant at the 5% level. First, all predictors and interaction terms from A.2 were used.

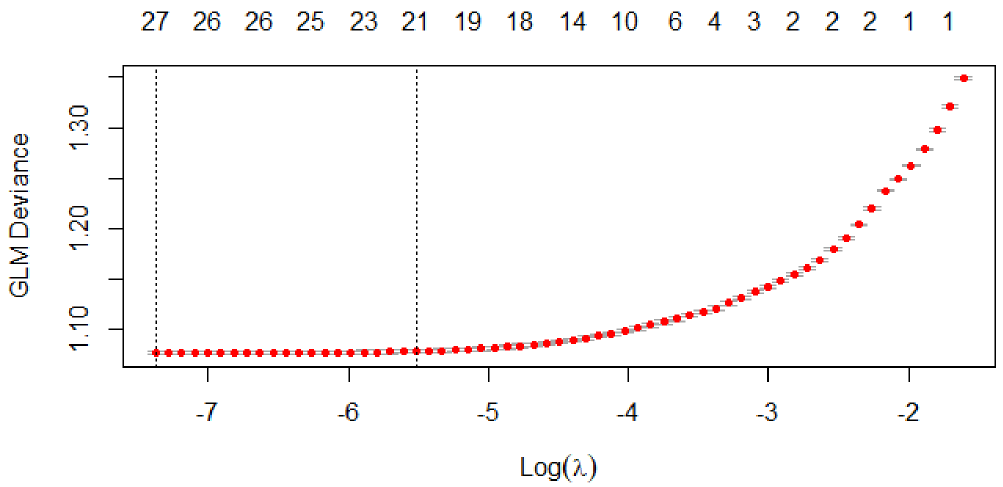

Figure 11: LASSO regularisation for Model 1 - CV deviance for different values of log lambda.

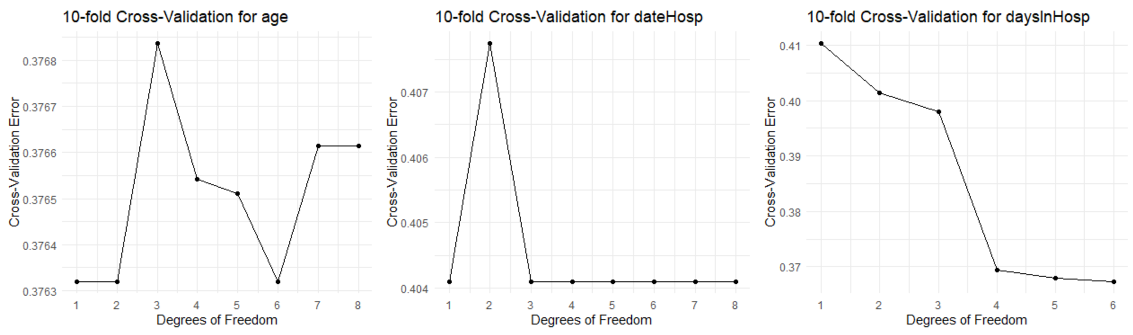

Figure 12: CV error results for differeing degrees of freedom for (from left to right) `age`, `dateHosp` and `daysInHosp`.

```
> print(anova(model2, test = "Chisq"))
Analysis of Deviance Table

Model: binomial, link: logit

Response: covidDeath

Terms added sequentially (first to last)


                 Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                          125767     169696
ns(age, df = 2)   2   7121.8  125765     162574 < 2.2e-16 ***

[lots of output]

cardio            1      0.8  125743     135580 0.35784
hepatic           1    101.4  125742     135478 < 2.2e-16 ***
immuno            1    323.6  125741     135155 < 2.2e-16 ***

[...]
```

From this, `cardio` was removed, and another test was done.

```
> print(anova(model2, test = "Chisq"))
Analysis of Deviance Table

Model: binomial, link: logit

Response: covidDeath

Terms added sequentially (first to last)


                  Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                           125767   169696
ns(age, df = 2)    2   7121.8  125765   162574 < 2.2e-16 ***

[...]

oxygensat:respdistress 1 51.0 125736 131747 9.190e-13 ***
dyspnoea:respdistress 1 0.4  125735   131747 0.51680
diarrhea:vomit     1     4.9  125734   131742 0.02759 *

[...]
```

Now, `dyspnoea * respdistress` was removed, and another test was done.

```
> print(anova(model2, test = "Chisq"))
Analysis of Deviance Table

Model: binomial, link: logit

Response: covidDeath

Terms added sequentially (first to last)


                  Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                           125767   169696
ns(age, df = 2)    2   7121.8  125765   162574 < 2.2e-16 ***
sex                1    199.2  125764   162375 < 2.2e-16 ***
vaccine            1   1279.9  125763   161095 < 2.2e-16 ***
ns(dateHosp, df = 3) 3 513.7 125760   160581 < 2.2e-16 ***
fever              1     58.2  125759   160523 2.392e-14 ***
cough              1    206.5  125758   160317 < 2.2e-16 ***
sorethroat         1     10.1  125757   160307 0.00151 **
dyspnoea           1   1377.9  125756   158929 < 2.2e-16 ***
oxygensat          1    818.5  125755   158110 < 2.2e-16 ***
diarrhea           1     57.8  125754   158052 2.885e-14 ***
vomit              1      4.7  125753   158048 0.03064 *
hematologic        1     30.4  125752   158017 3.534e-08 ***
downsyn            1     30.1  125751   157987 4.197e-08 ***
asthma             1     27.9  125750   157959 1.272e-07 ***
diabetes           1    247.9  125749   157712 < 2.2e-16 ***
neurological       1    198.4  125748   157513 < 2.2e-16 ***
pneumopathy        1    136.2  125747   157377 < 2.2e-16 ***
obesity            1    549.0  125746   156828 < 2.2e-16 ***
icu                1  20861.8  125745   135966 < 2.2e-16 ***
respdistress       1    385.4  125744   135581 < 2.2e-16 ***
hepatic            1    101.1  125743   135480 < 2.2e-16 ***
immuno             1    320.5  125742   135159 < 2.2e-16 ***
renal              1    358.9  125741   134800 < 2.2e-16 ***
ns(daysInHosp, df = 4) 4 3002.0 125737 131798 < 2.2e-16 ***
oxygensat:respdistress 1 51.0 125736 131747 9.190e-13 ***
```

```
diarrhea:vomit     1     4.9   125735   131742 0.02747 *
```

We see all predictors are significant at the 5% level. This completes model 2.

A.6 Constructing Model 3

We consider a single classification tree for Model 3. We do not apply ensemble methods, as mentioned in section 2.1, as this hinders interpretability of the model. First, we construct a full tree, using the `rpart` library. We set the cost penalty to be very small, 0.00001, so a very complex tree is made.

```
model3 <- rpart(
  covidDeath ~ age + sex + vaccine + dateHosp + fever + cough +
    sorethroat + dyspnoea + oxygensat + diarrhea + vomit + hematologic +
    downsyn + asthma + diabetes + neurological + pneumopathy + obesity +
    icu + respdistress + cardio + hepatic + immuno + renal + daysInHosp,
  data = data,
  method = "class",
  control = rpart.control(cp = 0.0001)
)
```

From this tree, we perform cost-complexity pruning, to simplify our tree to prevent overfitting and increase the interpretability of the model. Below is a summary of CV error for different values of `cp`, which dictate the complexity of the tree.

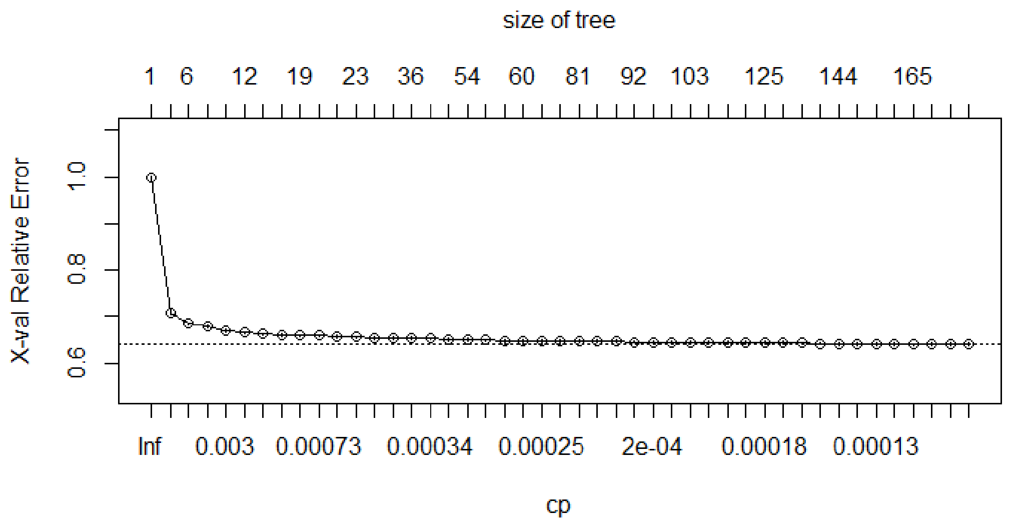

Figure 13: CV Residual Deviance for different complexity parameters of the tree

We see the error plateuas, so we consider the bias-variance tradeoff and select a complexity level that is as simple as possible without sacrificing much accuracy. This correspond to a `cp` value of 0.00078841, which gives the tree seen in figure 14.

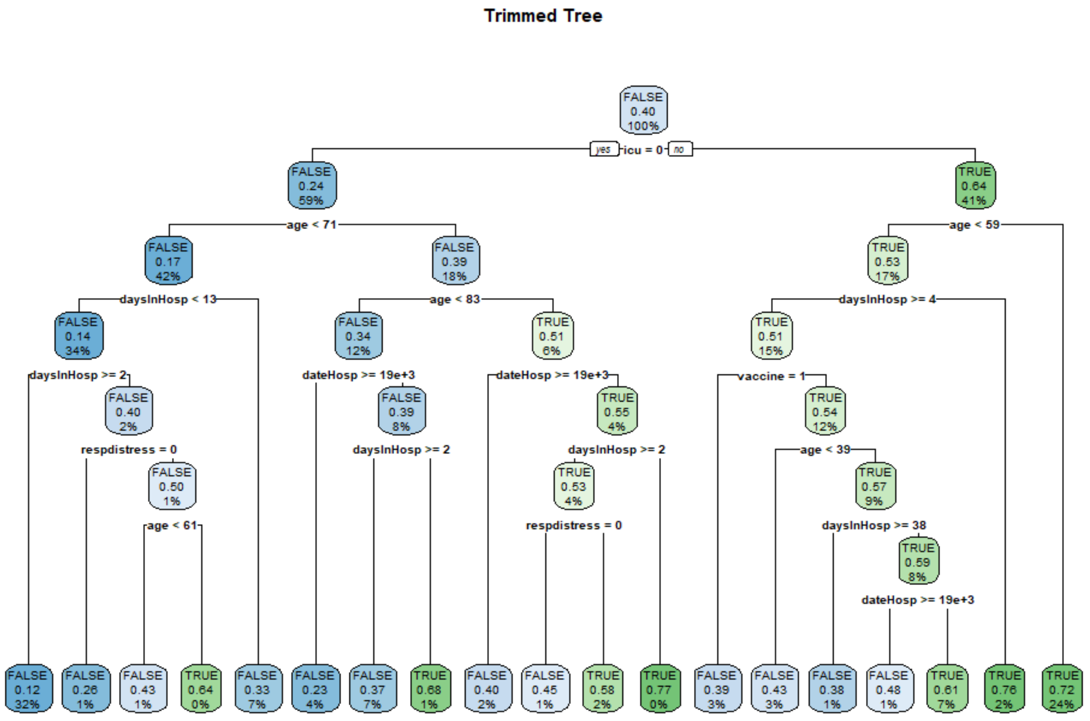

Figure 14: Visualisation of the tree for Model 3.

A.7 ROC Curve on the Test Set

As mentioned in section 2.2, the 20% test data was used to compare model 1 and 2 to assess their performance on unseen data across different thresholds. This was simply done for a comparison between the two models, as did not include model 3. This is because a classification tree uses a majority vote system to decide the class, based on the number of cases in the terminal node. Using a threshold does not make sense, as it would introduce a systematic bias to the model, and hence the AUC statistic is meaningless for such a model. Figure 15 shows the resulting ROC curve for Models 1 and 2.

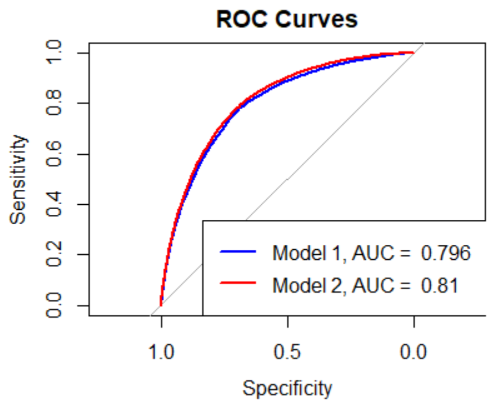

Figure 15: ROC Curve of Models 1 and 2 with AUC

A.8 Best Inferential Model Summary

```
> summary(best_model)
```

```
Call:
glm(formula = covidDeath ~ ns(age, df = 2) + sex + vaccine +
   ns(dateHosp, df = 3) + fever + cough + sorethroat + dyspnoea +
   oxygensat + diarrhea + vomit + hematologic + downsyn + asthma +
   diabetes + neurological + pneumopathy + obesity + icu + respdistress +
   hepatic + immuno + renal + ns(daysInHosp, df = 4) + oxygensat *
   respdistress + diarrhea * vomit, family = binomial(), data = filtered_data)

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)            -2.578000 0.085829 -30.036 < 2e-16 ***
ns(age, df = 2)1        4.360192 0.118985 36.645 < 2e-16 ***
ns(age, df = 2)2        4.455129 0.054633 81.547 < 2e-16 ***
sexM                    0.146892 0.013835 10.618 < 2e-16 ***
vaccineTRUE            -0.391989 0.016969 -23.100 < 2e-16 ***
ns(dateHosp, df = 3)1  -0.447281 0.035431 -12.624 < 2e-16 ***
ns(dateHosp, df = 3)2   0.861939 0.097250 8.863 < 2e-16 ***
ns(dateHosp, df = 3)3  -0.755951 0.054654 -13.831 < 2e-16 ***
feverTRUE              -0.040432 0.014331 -2.821 0.004781 **
coughTRUE              -0.163792 0.015681 -10.445 < 2e-16 ***
sorethroatTRUE         -0.032558 0.018645 -1.746 0.080769 .
dyspnoeaTRUE            0.223385 0.018587 12.018 < 2e-16 ***
oxygensatTRUE           0.200416 0.024409 8.211 < 2e-16 ***
diarrheaTRUE           -0.076610 0.022471 -3.409 0.000651 ***
vomitTRUE              -0.007176 0.031291 -0.229 0.818604
hematologicTRUE         0.109463 0.072759 1.504 0.132467
downsynTRUE             0.337888 0.096411 3.505 0.000457 ***
asthmaTRUE             -0.161333 0.036614 -4.406 1.05e-05 ***
diabetesTRUE            0.214722 0.014467 14.842 < 2e-16 ***
neurologicalTRUE        0.557109 0.030360 18.350 < 2e-16 ***
pneumopathyTRUE         0.293263 0.032449 9.038 < 2e-16 ***
obesityTRUE             0.283971 0.018401 15.432 < 2e-16 ***
icuTRUE                 2.093059 0.015004 139.500 < 2e-16 ***
respdistressTRUE        0.154001 0.032599 4.724 2.31e-06 ***
hepaticTRUE             0.601713 0.063510 9.474 < 2e-16 ***
immunoTRUE              0.711780 0.039701 17.928 < 2e-16 ***
renalTRUE               0.612269 0.031798 19.255 < 2e-16 ***
ns(daysInHosp, df = 4)1 -1.865145 0.034810 -53.581 < 2e-16 ***
ns(daysInHosp, df = 4)2 -0.795209 0.035423 -22.449 < 2e-16 ***
ns(daysInHosp, df = 4)3 -3.311453 0.076999 -43.007 < 2e-16 ***
ns(daysInHosp, df = 4)4 -0.040776 0.035257 -1.157 0.247467
oxygensatTRUE:respdistressTRUE 0.233653 0.036662 6.373 1.85e-10 ***
diarrheaTRUE:vomitTRUE  0.054908 0.050630 1.085 0.278142
---
Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

   Null deviance: 176824 on 133031 degrees of freedom
Residual deviance: 130871 on 132999 degrees of freedom
AIC: 130937

Number of Fisher Scoring iterations: 4
```

A.9 VIF of Predictors in Model

As mentioned in section 2.3, we check to see if there is significant multicollinearity amongst our predictors by calculating the VIF of each predictor. Each predictor's VIF is summarised below. The low values of each indicate that there is no significant multicollinearity present.

```
> car::vif(best_model, type = "predictor")
                         GVIF Df GVIF^(1/(2*Df))
ns(age, df = 2)   1.556804 2    1.117014
sex               1.024016 1    1.011937
vaccine           1.483340 1    1.217925
ns(dateHosp, df = 3) 1.348626 3 1.051111
fever             1.099078 1    1.048369
cough             1.091002 1    1.044511
sorethroat        1.080363 1    1.039405
dyspnoea          1.200240 1    1.095555
oxygensat         2.055559 1    1.433722
diarrhea          1.357530 1    1.165131
vomit             1.746819 1    1.321673
hematologic       1.014971 1    1.007458
downsyn           1.012425 1    1.006193
asthma            1.015823 1    1.007881
diabetes          1.045915 1    1.022700
neurological      1.036313 1    1.017995
pneumopathy       1.022126 1    1.011002
obesity           1.110160 1    1.053641
icu               1.176373 1    1.084607
respdistress      5.114026 1    2.261421
hepatic           1.012818 1    1.006389
immuno            1.019132 1    1.009521
renal             1.009711 1    1.004844
ns(daysInHosp, df = 4) 1.096600 4 1.011594
oxygensat:respdistress 6.961528 1 2.638471
diarrhea:vomit    2.140344 1    1.462992
```

A.10 Coefficients of Model Ordered by Magnitude

Before making comparisons between coefficients in our model, we scale our numerical data such that it has mean 0 and standard deviation 1. We do this with the tidyr package, with the following code.

```
data_scaled <- filtered_data %>%
  mutate_if(is.numeric, scale)
```

Now, we retrain our best model with this data, and order the coefficients. The output is below. Note in the analysis, we ignore spline coefficients, and just focus on binary predictors, as it is easier to make inferences from these.

```
> sort(scaled_model$coefficients)
     ns(daysInHosp, df = 4)3          (Intercept)    ns(daysInHosp, df = 4)1
             -3.311453110           -2.577999888           -1.865145460
     ns(daysInHosp, df = 4)2  ns(dateHosp, df = 3)3  ns(dateHosp, df = 3)1
             -0.795209018           -0.755950693           -0.447281165
              vaccineTRUE              coughTRUE              asthmaTRUE
             -0.391988764           -0.163792194           -0.161332785
             diarrheaTRUE   ns(daysInHosp, df = 4)4              feverTRUE
             -0.076609996           -0.040775714           -0.040432476
           sorethroatTRUE              vomitTRUE     diarrheaTRUE:vomitTRUE
             -0.032557794           -0.007176311            0.054907848
          hematologicTRUE                   sexM           respdistressTRUE
              0.109462515            0.146892168            0.154000574
            oxygensatTRUE           diabetesTRUE            dyspnoeaTRUE
              0.200416432            0.214722458            0.223385248
oxygensatTRUE:respdistressTRUE         obesityTRUE         pneumopathyTRUE
              0.233653415            0.283970975            0.293262854
              downsynTRUE        neurologicalTRUE             hepaticTRUE
              0.337887729            0.557109030            0.601713287
                renalTRUE             immunoTRUE    ns(dateHosp, df = 3)2
```

```
      0.612268838             0.711779747             0.861938694
         icuTRUE          ns(age, df = 2)1        ns(age, df = 2)2
      2.093058655             4.360192320             4.455128909
```

A.11 Elastic Net Regularisation for Model 4

Below is the cross validated deviance for different values of α, represented by different coloured lines, for a grid of λ values. λ_{min} was selected to get the best prediction accuracy.

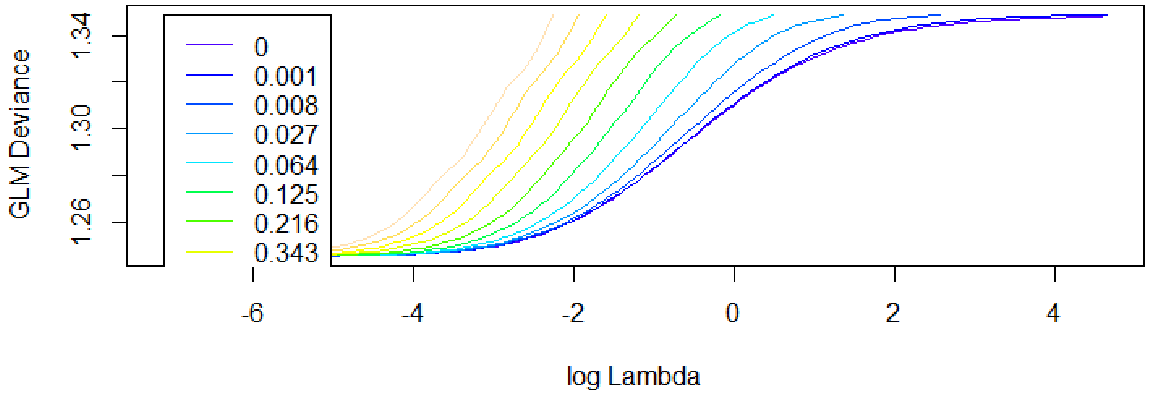

Figure 16: 10-fold CV Residual Deviance for α and λ for Model 4.

A.12 Optimising the Parameters for Model 5

To determine the span s for the local regression on `dateHosp`, and the smooth parameter λ for the smoothing spline on `age`, cross validation with F1-Score was used, training models with different tuning parameters and calculating F1-score. The results are visualised in figure 17. From the results, we select $\lambda = 0.5$ as it maximises F1 score noticeably, and we select $s = 0.2$, as using 0.1 may lead to over-fitting as a result of the bias-variance tradeoff.

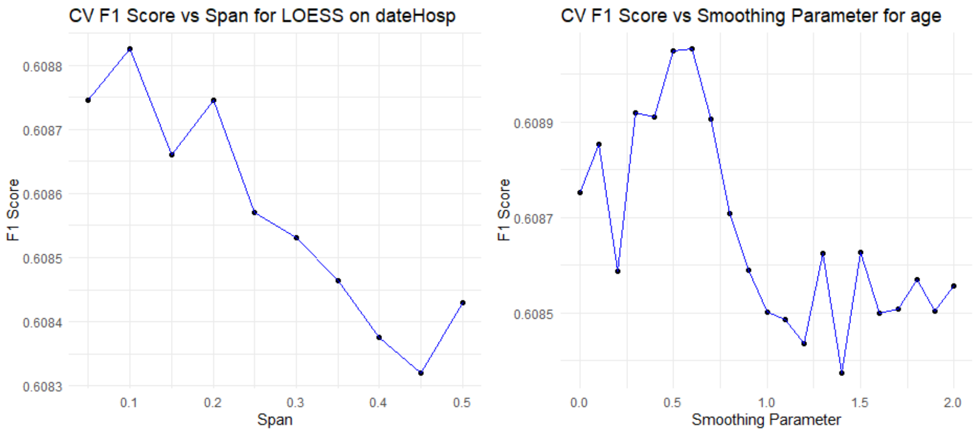

Figure 17: 10-fold CV F1-Score for LOESS on `dateHosp` and smoothing spline on `age`, with respective tuning parameters.

A.13 Elastic Net Regularisation for Model 5

After using the parameters found in A.12, we apply elastic net regularisation to shrink down the noise in the model. We apply an identical approach to Model 4, discussed in A.11, using 10-fold cross validation on deviance to tune α and λ. The results are visualised in figure 18, and we reach parameters $\alpha = 0.001$ and $\lambda = 0.01161$.

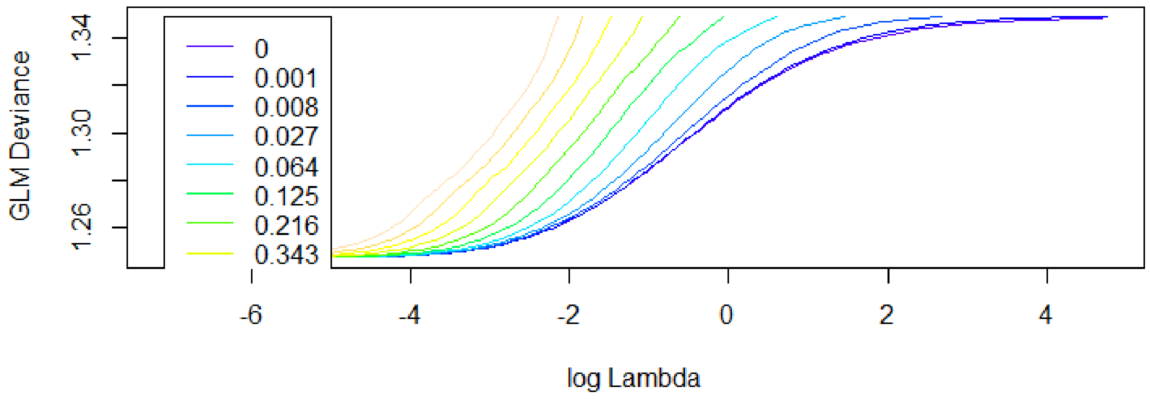

Figure 18: 10-fold CV Residual Deviance for α and λ for Model 5.

A.14 Gradient-Boosting Overview

Similar to the standard boosting ensemble method, gradient-boosting seeks achieve powerful prediction performance by combining several weak learners into strong learners. Boosting does this by sequentially training weak learners, identifying false classifications of previous learners and adding weight to them. Thus, future weak learners will focus more on examples that past learners misclassified. Gradient boosting, developed by Jerome Friedman in 1999[3], modifies this algorithm by training new weak learners with respect to the gradient of the loss function of the current ensemble (all previous weak learners combined).

Mathematically, denote some imperfect model F_m. Our algorithm must improve our model, say by adding an other weak learner h_m. So, we have

$$F_{m+1}(x_i) = F_m(x_i) + h_m(x_i) = y_i.$$

Rearranging gives us,

$$h_m(x_i) = y_i - F_m(x_i).$$

This is boosting, as we are fitting h_m to the residual $y_i - F_m(x_i)$. Now, consider the MSE loss function $\mathcal{L}_{\text{MSE}}$. We have that

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - F(x_i))^2$$

$$\Rightarrow \frac{\partial \mathcal{L}_{\text{MSE}}}{\partial F(x_i)} = -\frac{2}{n}(y_i - F(x_i)) = -\frac{2}{n}h_m(x_i).$$

We see that the gradient of $\mathcal{L}_{\text{MSE}}$ is proportional to $h_m(x_i)$. It can be shown this is the case for other loss functions, including those for classification such as F-Score[8]. This gives way to gradient boosting, where we use this fact to improve our model by training new weak learners on the gradient of the loss function.

This methodology has gained meteoric popularity in many fields for its relatively quick training time, extensive parameters for hyper-tuning, and most importantly, its high predictive accuracy.

A.15 Hyper-tuning the Gradient Boost Model (Model 6) with Racing Methods

The following racing method is credited to Julia Silge[17], who outlined the method on her blog applied to a different dataset. In Silge's case, less parameters and combinations were considered to reduce computation time given the fact it was done for a competition where contestants had very limited time to make predictions. In our case, we increased the number of tuned predictors and models to increase the likelihood of getting the most optimal model. `xgboost` was selected as the engine for the gradient boost, due to its computational speed. The following code outlines the parameters tuned, and the grid of values selected. For information on each parameter, see A.16. The grid of values was decided as they are common values for a dataset of this size[5].

```
library(xgboost)

xgb_spec <-
 boost_tree(
   trees = tune(),
   min_n = tune(),
   mtry = tune(),
   tree_depth = tune(),
   learn_rate = tune(),
   loss_reduction = tune(),
   sample_size = tune(),
   stop_iter = tune()
 ) %>%
 set_engine("xgboost") %>%
 set_mode("classification")

xgb_grid <- grid_latin_hypercube(
 trees(range = c(300, 500)),
 min_n(range = c(1, 20)),
 mtry(range = c(1, 10)),
 tree_depth(range = c(3, 10)),
 learn_rate(range = c(0.01, 0.3)),
 loss_reduction(range = c(0, 10)),
 sample_size = sample_prop(range = c(0.5, 1)),
 stop_iter(range = c(10, 50)),
 size = 1000
)
```

A.16 Gradient-Boosting (Model 6) Race Results

Using `tune_race_anova`, we can hypertune our model by using the `xgb_spec` and grid outlined in A.15. Note that parallel computing was used for this, otherwise the computation time would be extremely long. Figure 19 visualises the racing algorithm, where we begin with 1,000 models, and then progressively remove models that are significantly under-performing over different folds. Clearly, we see that the majority of the models selected have around 0.5 F-score, while the other models plateau around 0.575. Although it is possible there exists a perfect model which can outperform the GAM's discussed in section 3.1, it would require incredible computational power and time, which is largely unfeasible given that this ideal model may not even exist.

Now, after we have selected the best performing model from the race, we look to the parameters. Here, each parameter is given, with a brief summation from the `xgboost` documentation of how it impacts the model[10].

- `mtry = 5` : the number of randomly selected predictors for each weak learner.
- `trees = 408` : the number of weak learners, i.e. trees.
- `min_n = 18` : the minimum number of observations required for each terminal node.
- `max_depth = 8` : the maximum number of splits (branches) allowed for a tree to reach a terminal node.
- `loss_reduction = 3890` : a.k.a. gamma, a regularisation parameter that controls the complexity of the tree.
- `sample_size = 0.681` : the proportion of the population taken for bootstrapped samples.
- `stop_iter = 23` : specifies the number of rounds to continue training after the last improvement in the validation metric.

- `learning_rate` : a.k.a. eta, the step size at each iteration of model (addition of weak learners) while moving towards the minimum of the loss function.

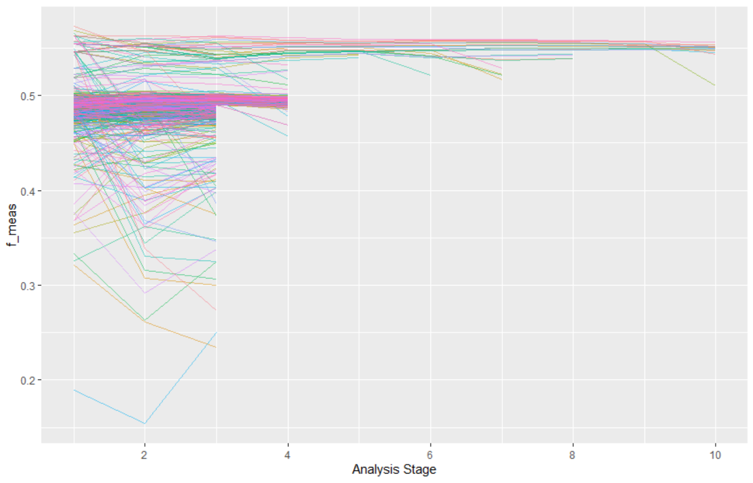

Figure 19: Visualisation of race tuning for Gradient-Boosted Tree.

A.17 Random Forest (Model 7) Race Algorithm

We employ an identical method to tune Model 7. Although there are less parameters, tuning a random forest is more computationally heavy than tuning a gradient boost model, although this largely thanks to the amazing optimisation of `xgboost`. For this reason, only 100 models were sampled (although the computation time was twice as long as the time it took to race 1000 `xgboost` models!) The the following code is what was used to tune the model, and the grid of values chosen for each parameter. Note that $m = \sqrt{p}$ was chosen, as it is almost always an optimal choice.

```
library(ranger)

rf_spec <- rand_forest(
   mtry = tune(),
   trees = tune(),
   min_n = tune()
  ) %>%
  set_engine("ranger") %>%
  set_mode("classification")

rf_grid <- grid_latin_hypercube(
  finalize(mtry(), train),
  trees(range = c(300, 1000)),
  min_n(range = c(2, 10)),
  size = 100
)
```

Now, we again use `tune_race_anova` to apply a racing algorithm, which is visualised in figure 20. The parameters of the best model are outlined in section 3.1.

A.18 Two Proportion Z-Test on Proportion of Vaccinated Survivors and Non-Survivors

In section 3.3, we seek to test whether the proportion of vaccinated survivors is less than the proportion of vaccinated non-survivors. So, consider the following hypothesis,

$$H_0 : p_{\text{vac \& survived}} = p_{\text{vac \& died}} \quad \text{v.s.} \quad p_{\text{vac \& survived}} < p_{\text{vac \& died}}.$$

We perform this test in `R` using the following code.

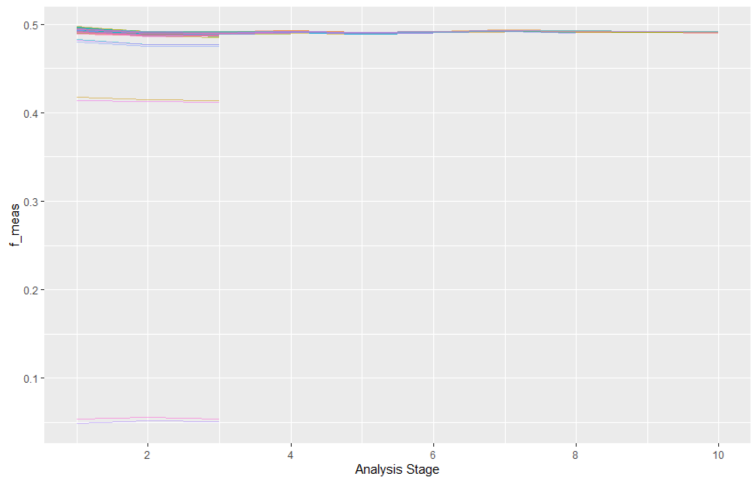

Figure 20: Visualisation of race tuning for Random Forest.

```
> prop.test(
+   x = c(vaccinated_survivors, vaccinated_non_survivors),
+   n = c(total_survivors, total_non_survivors),
+   alternative = "greater"
+ )

	2-sample test for equality of proportions with continuity correction

data: c(vaccinated_survivors, vaccinated_non_survivors) out of c(total_survivors,
    total_non_survivors)
X-squared = 8.4222, df = 1, p-value = 0.001853
alternative hypothesis: greater
95 percent confidence interval:
 0.00864178 1.00000000
sample estimates:
  prop 1  prop 2
0.4135599 0.3933934
```

Interpreting the outcome, we clearly see that the proportion of vaccinated survivors 0.4136 is greater than the proportion of vaccinated non-survivors 0.3934, with a p value of 0.001853. This provides strong evidence to our claim in section 3.3.

A.19 Generative AI Usage

ChatGPT was used to generate `R` code as well as validate and generate some ideas throughout the report. Below are prompts used for particular purposes.

- **Validate / Generate Ideas**
  - "a big drawback of logistic regression and thus gam's is we must assume each observation is independent to each other. intuitively, covid deaths is not very independent, as there can be correlated deaths from local outbreaks and thus there are temporal dependencies. by adding a predictor for dateHosp (date hospitalised), which I have shown is a good fit from residuals, does this address the limitation of independence?"
  - "like the main dataset, survivors and non-survivors has symptoms (binary). how can i get an understanding of the proportion of non-survivors had each symptom?"
  - "im finding it hard to use loess() on a date variable. is it a good idea to add a new column daysSinceStart which is the days since the date 1st January 2021 (when the data starts)?"
  - "for the random forest, what is a good size?"
  - *(pasted code)* "in this code, i tune trees, min_n and mtry... what else can i tune to improve predictive accuracy?"

- **Generate `R` code**
  - "when i compare coefficients, how can i normalise the coefficient values so i can make comparisons?"
  - "i have two datasets - survivors and non_survivors. can you give me code to make a stacked box plot of the distribution of age in both datasets?"
  - "with this dataset, there is dateHosp. how can i show the number of admissions on the same plot for both datasets?"
  - "can we rotate the symptom labels so they dont go over each other?"
  - "in my dataset test_data, which has covidDeath and vaccine, how can i perform a two proportion z test on the proportion of vaccinated survivors and non-survivors?"
  - "i want to illustrate the mortality rate for each month. i have two data sets, test_data, which contains all patients, and non-survivors, which contains all non-survivors. both of these datasets has a predictor dateHosp. i want to find the proportion of non-survivors out of all the people hospitalsed for each month, and then visualise this. how can i do this?"
  - "can you give me ggplot2 code to plot this dataframe glm_f1 ¡- data.frame(threshold = seq(0.1, 0.8, 0.01), f1 = f1_vec) just showing the points, and then a vertical line where the maximum is"
  - "now, in my glm, i have done some polynomial terms with the coefficients. how can i visualise these over the dataset?"
  - "how do i use glm.lo and lo() in R to perform LOESS predictions? refer to my recent covidDeath dataset"

References

[1] Ali SI Darwish E. Al Omair OA Essa A Elzorkany K Shehab-Eldeen S Alarfaj HM Alarfaj SM Alabdulqader F Aldoughan A Agha M. *Factors Affecting Hospitalization Length and in-Hospital Death Due to COVID-19 Infection in Saudi Arabia: A Single-Center Retrospective Analysis*. PubMed Central. Jan. 8, 2023. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10404051/.

[2] David A Belsley, Edwin Kuh, and Roy E Welsch. *Regression diagnostics: Identifying influential data and sources of collinearity*. John Wiley & Sons, 2005.

[3] Jerome H Friedman. *Greedy Function Approximation: A Gradient Boosting Machine*. Technical Report IMS-TR: 1999-15. Stanford University, 1999.

[4] Nicola Talbot Gavin Cawley. *On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation*. MIT. Oct. 2024. URL: https://jmlr.csail.mit.edu/papers/volume11/cawley10a/cawley10a.pdf.

[5] Aurelien Geron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, 2017. ISBN: 978-1491962299.

[6] Mohamed Hebiri Johannes C. Lederer. *How Correlations Influence Lasso Prediction*. Université Paris-Est. Sept. 7, 2021. URL: https://arxiv.org/pdf/1204.1605.

[7] Eryk Lewinson. *A Comprehensive Guide to Interaction Terms in Linear Regression*. Nvidia Technical Blog. URL: https://developer.nvidia.com/blog/a-comprehensive-guide-to-interaction-terms-in-linear-regression/.

[8] Cheng Li. *A Gentle Introduction to Gradient Boosting*. College of Computer and Information Science Northeastern University. 2020. URL: http://www.chengli.io/tutorials/gradient_boosting.pdf.

[9] Matthew Diamond1; Hector L. Peniston; Devang K. Sanghavi; Sidharth Mahapatra. *Acute Respiratory Distress Syndrome*. University of Nebraska Medical Center. Jan. 31, 2024. URL: https://www.ncbi.nlm.nih.gov/books/NBK436002/.

[10] Davis Vaughan Max Kuhn. *Boosted trees via xgboost*. Posit. URL: https://parsnip.tidymodels.org/reference/details_boost_tree_xgboost.html.

[11] Lindsey M. Duca Likang Xu Sandy F. Price Catherine A. McLeanD. *COVID-19 Incidence, by Age Group — United States*. CDC. Jan. 1, 2021. URL: https://www.cdc.gov/mmwr/volumes/69/wr/mm695152a8.htm.

[12] J. Mendoza. *COVID-19 vaccine immunization development in Brazil 2021-2023*. statista. Apr. 10, 2023. URL: https://www.statista.com/statistics/1288019/population-vaccinated-against-covid-brazil/.

[13] J. Mendoza. *COVID-19 vaccine immunization development in Brazil 2021-2023*. statista. Apr. 10, 2023. URL: https://www.statista.com/statistics/1288019/population-vaccinated-against-covid-brazil/.

[14] NIH. *COVID-19 and the Nervous System*. National Institute of Neurological Disorders and Stroke. URL: https://www.ninds.nih.gov/current-research/coronavirus-and-ninds/covid-19-and-nervous-system.

[15] Robert M O'Brien. *A Caution Regarding Rules of Thumb for Variance Inflation Factors*. Springer. URL: https://link.springer.com/article/10.1007/s11135-006-9018-6#Bib1.

[16] Natasha Sharma. *Understanding and Applying F1 Score: AI Evaluation Essentials with Hands-On Coding Example*. arize. June 5, 2023. URL: https://arize.com/blog-course/f1-score/.

[17] Julia Silge. *Use racing methods to tune xgboost models and predict home runs*. Posit. URL: https://juliasilge.com/blog/baseball-racing/.

[18] Eduardo Simões. *Brazil hospitals pushed to limit as COVID-19 death toll soars*. Reuters. Dec. 3, 2021. URL: https://www.reuters.com/world/americas/brazil-hospitals-pushed-limit-covid-19-death-toll-soars-2021-03-11/.

[19] da Silva Baum K. Sott MK Bender MS. *Covid-19 Outbreak in Brazil: Health, Social, Political, and Economic Implications*. Int J Health Serv. Apr. 10, 2022. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9445630/.

[20] Stephani Sutherland. *Long COVID Now Looks like a Neurological Disease, Helping Doctors to Focus Treatments*. Scientific American. Jan. 3, 2023. URL: https://www.scientificamerican.com/article/long-covid-now-looks-like-a-neurological-disease-helping-doctors-to-focus-treatments1/.

[21] Soumya Swaminathan. *Coronavirus disease (COVID-19): Herd immunity, lockdowns and COVID-19*. World Health Organization. URL: https://www.who.int/news-room/questions-and-answers/item/herd-immunity-lockdowns-and-covid-19.