P8106 Final Project

Wenjia Zhu, Ruihan Zhang, Jasmine Niu

May 10, 2023

Contents

Background

To better understand the factors that predict recovery time from COVID-19 illness, a study was designed to combine three existing cohort studies that have been tracking participants for several years. The study collects recovery information through questionnaires and medical records and leverages existing data on personal characteristics before the pandemic.

Introduction

The aim of this research is to develop a prediction model for recovery time from COVID-19 by merging three cohort studies that have been following participants for several years. Recovery information will be gathered through questionnaires and medical records, and personal characteristics data from before the pandemic will be utilized. The ultimate goal is to develop a prediction model for recovery time and identify important risk factors for a long recovery time.

Data and Exploratory Analysis

This study employs the recovery.RData file, which comprises a dataset of 10,000 participants. The dataset contains a variable for recovery time from COVID-19 (in days) along with 14 predictor variables, including demographic features, personal characteristics, vital measurements, and disease status. The predictors consist of both continuous and categorical variables. The study used two merged random samples of 2000 participants, which were obtained from the seed set to 2631 and 2855 respectively, to create the final dataset. The training dataset contains 80% of the sample and, the test dataset contains the remaining 20%.

Table 1. Description of variables		
1	ID (id)	Participant ID
2	Age (age)	Participant age
3	Gender (gender)	1 = Male, 0 = Female
4	Race/ethnicity (race)	1 = White, 2 = Asian, 3 = Black, 4 = Hispanic
5	Smoking (smoking)	Smoking status; 0 = Never smoked, 1 = Former smoker, 2 = Current smoker
6	Height (height)	Height (in centimeters)
7	Weight (weight)	Weight (in kilograms)
8	BMI (bmi)	Body Mass Index; BMI = weight (in kilograms) / height (in meters) squared
9	Hypertension (hypertension)	0 = No, 1 = Yes
10	Diabetes (diabetes)	0 = No, 1 = Yes
11	Systolic blood pressure (SBP)	Systolic blood pressure (in mm/Hg)
12	LDL cholesterol (LDL)	LDL (low-density lipoprotein) cholesterol (in mg/dL)
13	Vaccination status at the time of infection (vaccine)	0 = Not vaccinated, 1 = Vaccinated
14	Severity of COVID-19 infection (severity)	0 = Not severe, 1= Severe
15	Study (study)	The study (A/B/C) that the participant belongs to
16	Time to recovery (tt_recovery_time)	Time from COVID-19 infection to recovery in days

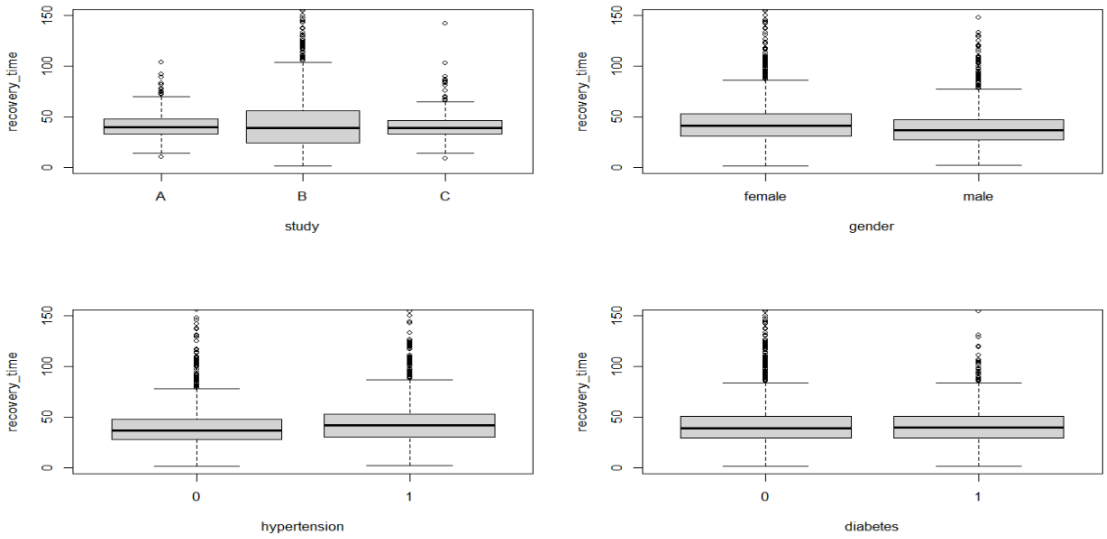


Figure 1. the relationship between continuous predictors(sbp, ldl, age, weight, height and bmi) and recovery time

Five lattice plots have been used to visualize the relationship between continuous predictors(sbp, ldl, age, weight, height and bmi) and recovery time. There is almost no relationship between sbp and recovery time, between age and recovery time, and between ldl and recovery time. There is a positive relationship between height and recovery time, and between weight and recovery time. There is a negative relationship between bmi and recovery time.

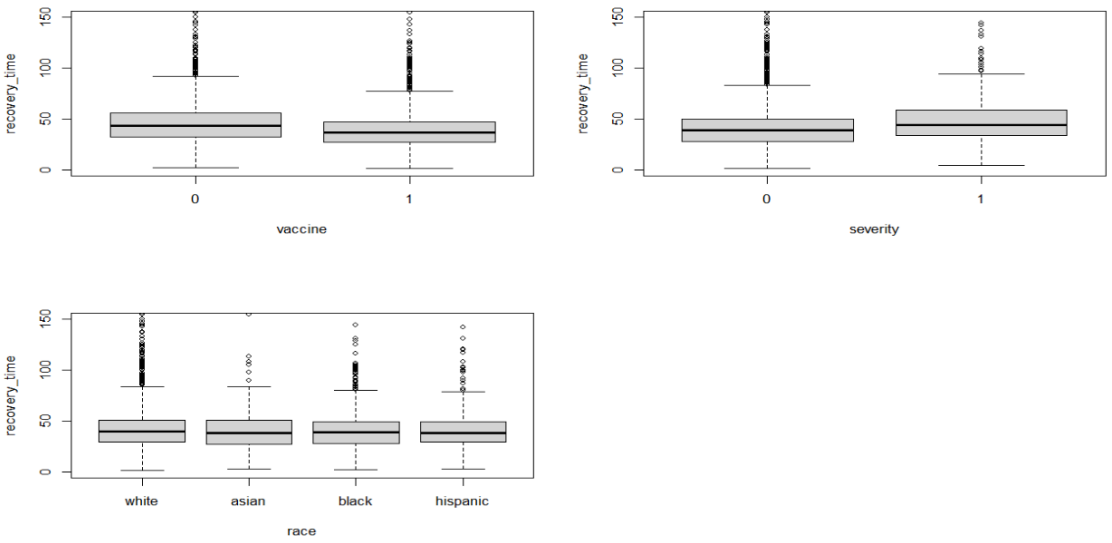


Figure 1. the relationship between predictors and recovery time

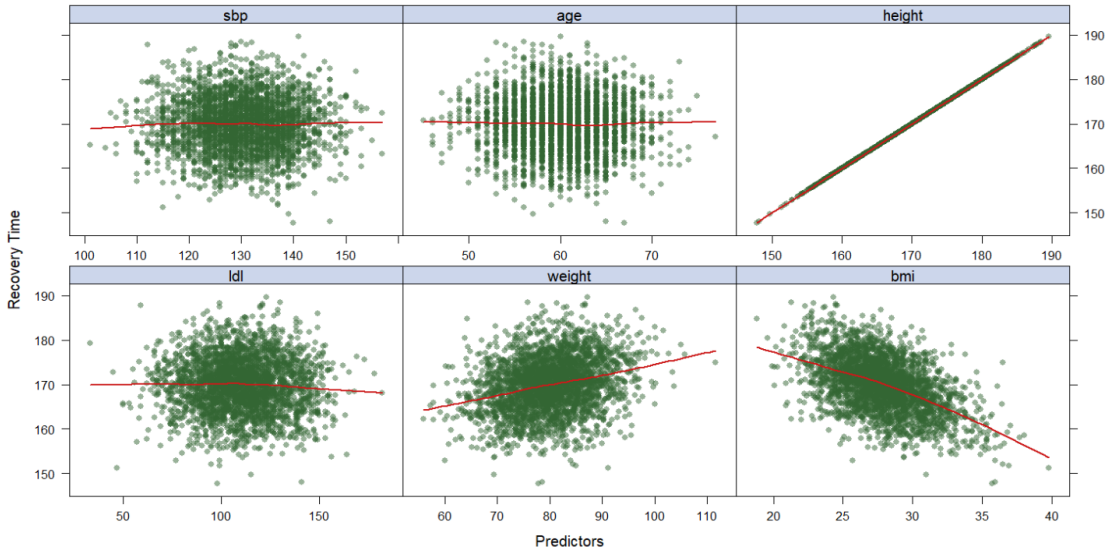


Figure 2. the relationship between categorical predictors(study, hypertension, gender, diabetes, vaccine, race, and severity) and recovery time

Seven boxplots have been used to visualize the relationship between continuous predictors(study, hypertension, gender, diabetes, vaccine, race, and severity) and recovery time. There is no large difference in recovery time among patients in study A, study B and study C. Patients with hypertension have a slightly longer recovery time than patients without hypertension. Female patients have a slightly longer recovery time than male patients. There is almost no difference in recovery time between patients with diabetes and patients

without diabetes. Vaccinated patients have a slightly shorter recovery time than unvaccinated patients. There is almost no difference in recovery time among white patients, Asian patients, black patients and Hispanic patients.

Model Training

Eleven models were used in this project as follows:

A linear model assumes $Y = \beta_0 + \beta_1 X + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$. The train() function with 10-fold cross-validation was used to fit this linear model 5 times, as specified by the trainControl() function, using linear regression. Statistical information about the model was obtained by summarizing it.

The ridge regression assumes that the dependence of outcome on predictors is linear and that the errors have constant variance and normal distribution. It allows the coefficients $\hat{\beta}_\lambda^R$ to minimize $RSS + \lambda \sum_i^p \beta_i^2$ towards zero to prevent overfitting. Ridge regression also assumes that all predictor variables have the same scale, as it involves the penalty term that adds the square of the coefficients to the objective function, and without the same scale, variables with larger values can dominate the penalty term. This model was created using the train function with 10-fold cross-validation repeated 5 times, as specified by the trainControl() function. The method used was glmnet within the train function. The tuning parameters were set using the tuneGrid argument with a range from $\exp\{-1\}$ to $\exp\{6\}$ and a length of 100. The model was specified by setting the alpha parameter to 0.

The Lasso model assumes that the dependence of the outcome on predictors is linear and that the errors have a constant variance and normal distribution. It also assumes that the predictors are not highly correlated with each other. One key aspect of the Lasso model is its use of the L1 penalty, which shrinks some of the coefficient estimates $\hat{\beta}_\lambda^R$ to minimize $RSS + \lambda \sum_i^p |\beta_i|$ towards zero and can effectively perform variable selection by setting some coefficients exactly to zero. To fit this Lasso model, the train() function was used with 10 folds cross validation, repeated 5 times. The method used was glmnet within the train function. The tuning parameters were set using the tuneGrid argument with a range from $\exp\{-1\}$ to $\exp\{5\}$ and a length of 100. The Lasso model was specified by setting the alpha parameter to 1. To obtain all possible combinations of lambda and alpha, the expand.grid() function was utilized.

Elastic Net is a linear regression model that combines the penalties of Lasso and Ridge regression to overcome some of their limitations. It minimizes $\sum_i^n (y_i - \beta_0 - \sum_i^n \beta_j x_{ij})^2 + \lambda_1 \sum_i^p \beta_j^2 + \lambda_2 |\beta_j|$. It assumes that the dependence of the outcome on predictors is linear and that the errors have a constant variance and normal distribution. Elastic Net also assumes that there is no multicollinearity among predictors. Additionally, it assumes that the data is standardized before fitting the model, so that all predictors have the same scale. This model was trained using the train function with 10-fold cross-validation for 5 iterations, utilizing the trainControl() function. The method employed was glmnet within the train function. The tuning parameters for this model included 21 values for the alpha parameter ranging from 0 to 1, and 50 values for the lambda parameter ranging from $\exp\{-10\}$ to $\exp\{10\}$. The expand.grid() function was utilized to generate all possible combinations of lambda and alpha for tuning the model.

PCR (Principal Component Regression) assumes that the predictors are linearly related to the outcome variable, and that there is no multicollinearity among the predictors. PCR also assumes that the first few principal components capture most of the variability in the predictors and that the remaining principal components do not contain any significant information. The pcr method was used within the train() function. The tuneGrid argument used a data frame with one column, ncomp, which ranged from 1 to 19. The training data were centered and scaled using the preProcess argument with "center" and "scale" options, respectively.

MARS (Multivariate Adaptive Regression Splines) is a non-parametric regression method that can capture non-linearities and interactions between variables. It assumes that the relationship between the predictors and the outcome is additive, i.e., the effect of each predictor on the outcome is independent of the values of other predictors. MARS also assumes that the relationship is continuous and that there are no significant outliers or influential observations. This model was created using the train() function with 10-fold cross-validation repeated 5 times, as specified by the trainControl() function. The method used was

earth within the train function. To create a grid of tuning parameters, the expand.grid function was used with the degree ranging from 1 to 3 and the nprune ranging from 2 to 18.

GAM (Generalized Additive Model) assumes that the relationship between the response variable and predictors is non-linear but retains the additive structure of linear models. It can be represented as a sum of smooth functions of the predictors. The model assumes that the errors have a constant variance and are independent and normally distributed. Additionally, GAM assumes that the effects of each predictor are additive and can be represented by smooth functions. The model is also assumed to have no multicollinearity among the predictor variables. This model was trained using the train() function with 10-fold cross-validation repeated 5 times, using the trainControl() function. The method used was gam, as specified in the train() function.

Regression trees assume that the relationship between the dependent variable and the independent variables is non-linear and can be represented as a tree-like structure. The model can capture complex interactions between the predictors and the outcome variable, and it can handle non-normal distributions and non-constant variance of errors. The method was rpart, which stands for Recursive Partitioning and Regression Trees. The tuneGrid argument was used to specify a grid of 50 values for cp is created using the exp function to generate values between $\exp\{-6\}$ and $\exp\{-2\}$. This model was trained using the train() function with 10-fold cross-validation repeated 5 times, using the trainControl() function.

Bagging (Bootstrap Aggregation) combines multiple models to improve the overall performance of the prediction. It does not make any assumptions about the relationship between predictors and outcome, as it can work with any base model. The idea is to generate multiple training sets by sampling the original data with replacement, and then train a base model on each of these training sets. The final prediction is obtained by averaging the predictions of all the base models. The assumption in bagging is that the errors of the base models are uncorrelated, so that the averaging procedure can effectively reduce the variance of the prediction. The random forest model used the train() function. The model was built using the ranger method, and the tuneGrid argument was set to a data frame with combinations of tuning parameters: mtry ranged from 1 to 16, splitrule was set to "variance", and min.node.size ranged from 1 to 6. This model was trained using the train() function with 10-fold cross-validation repeated 5 times, using the trainControl() function.

Boosting is a general technique for improving the accuracy of any given learning algorithm. The main assumption is that the individual base learning algorithms combined to create a boosted ensemble are weak learners. This means that each individual algorithm performs only slightly better than random guessing, but when combined in a certain way, they can form a strong ensemble that significantly improves predictive accuracy. Boosting also assumes that there is no overfitting of the training data by the individual base learners. Bagging, random forests and boosting are powerful methods for improving the prediction accuracy of trees. The method used for Boosting was gbm. There was a grid of tuning parameters using the expand.grid() function, with the number of trees equal to 1000, 2000, 3000, 4000 and 5000, interaction depth from 1 to 5, shrinkage with 0.0001, 0.0003 and 0.0005, and minimum number of observations 1 and 10 in a node (n.minobsinnode). This model was trained using the train() function with 10-fold cross-validation repeated 5 times, using the trainControl() function.

The SVM (Support Vector Machine) assumes that the data is linearly separable, meaning that there exists a hyperplane that can perfectly separate the two classes in the data. SVM also assumes that the data is normalized, and that the margin around the separating hyperplane is maximized. Additionally, SVM assumes that the selected kernel function is appropriate for the data and the problem at hand. Finally, SVM assumes that there are no outliers in the data that can greatly affect the separation of the classes.In this model, the code performed a grid search to tune hyperparameters for a support vector machine regression model (svmRadialSigma). Expand.grid() function creates a data frame with all possible combinations of the C and sigma values specified in the seq() functions.A range of values ranged between $\exp\{-2\}$ and $\exp\{2\}$ for cost, and ranged between $\exp\{-1\}$ and $\exp\{2\}$ for sigma. This model was trained using the train() function with 10-fold cross-validation repeated 5 times, using the trainControl() function.

Additionally, the RMSE of each model was computed by comparing predicted and actual recovery time values.

Results

For primary analysis (regression), we aimed to develop a prediction model for time to recovery as a continuous outcome variable. On the other hand, for secondary analysis (classification), we considered time to recovery as a binary outcome (>30 days vs. <= 30 days) and aimed to develop a prediction model for this binary outcome. The 11 models were compared both for regression and classification.

According to Figure 3, the boosted tree model has the smallest mean and median RMSE or prediction error rate. Thus, it was the final model selected with the best performance in the primary analysis. In addition, we compared the predicted values and the true values from the test dataset to evaluate the performance of the final model. Shown in Table 1, the RMSE of the boosted tree model for prediction is 18.99, which indicates that the predicted time to recovery will differ the true value by 18.99 days on average. Furthermore, we identified the variables with the largest overall impact in the final model using the Variable Importance Plot (VIP, shown in Figure 4). BMI is the most important predictor, followed by study.

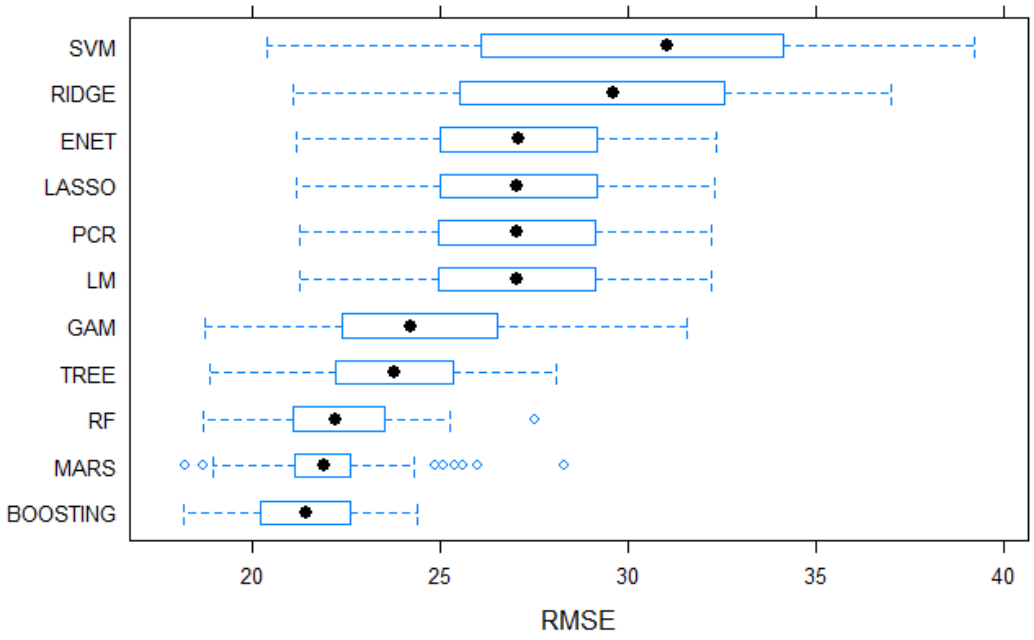


Figure 3. the Distribution of RMSE of the eleven models (regression)

From Figure 3, the boosted tree model have the smallest RMSE.

Description: df [11 × 2]

model <chr>	test_rmse <chr>
LM	23.7462437715614
RIDGE	24.5850633133414
LASSO	31.8643478753079
ENET	23.6962272128894
PCR	19.7536245232316
GAM	20.5804263050801
MARS	20.4688652609227
TREE	19.6710310045745
RF	19.6208480973905
BOOSTING	18.9875413794448
SVM	24.6463267912706

11 rows

Table 1. RMSE of the eleven models for prediction (regression)

From Table 1, the boosted tree model had the smallest RMSE which is about 18.988.

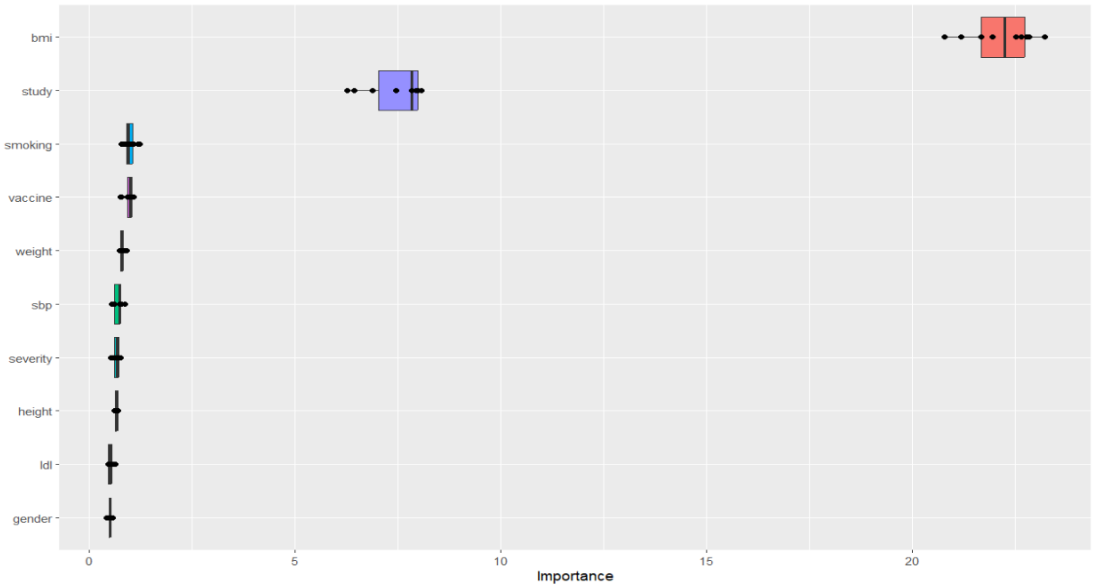


Figure 4. the Variable Importance Plot (regression)

As for classification, shown in Figure 5, the boosted tree model has the smallest mean and median RMSE, so we selected it as the best model, same as regression, to predict the binary value of time to recovery from COVID-19. We also estimated the performance of the final model with the test data. Shown in Figure 6, the R-Squared of the boosted tree model is 70.81%, revealing that around 70% of the variability observed in the outcome is explained by this model. And the most important variable in the final model for classification is bmi.

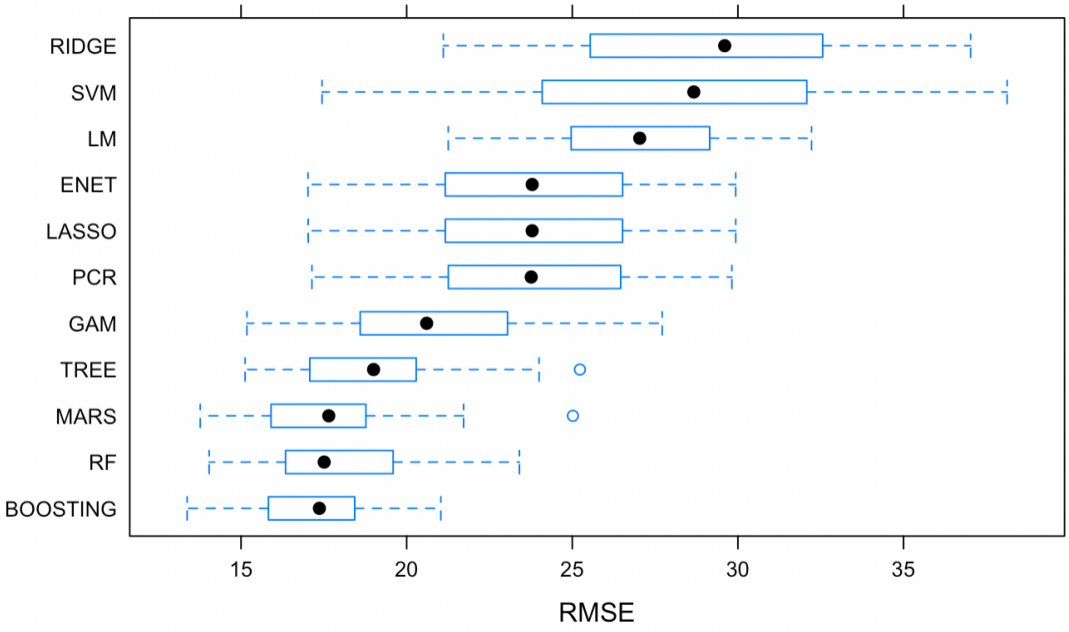


Figure 5. Distribution of RMSE of the eleven models (classification)

From Figure 5, the boosted tree model had the smallest RMSE, so it should be selected as the final model with the best performance.

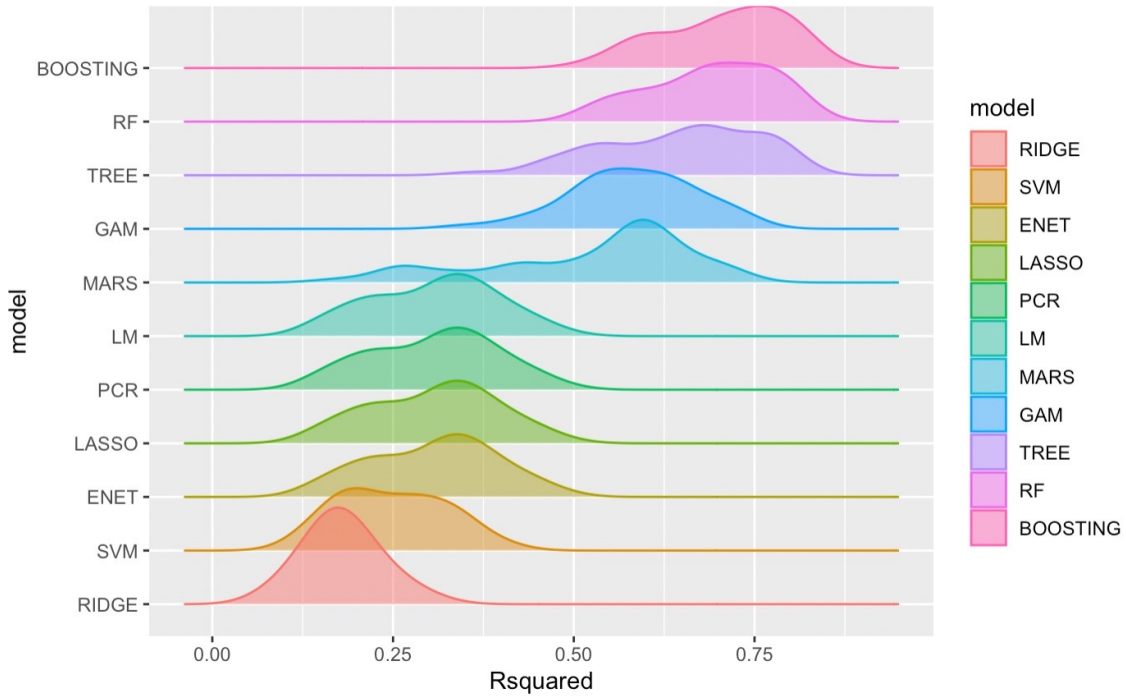


Figure 6. Prediction accuracy of the eleven models (classification)

Theoretically, the regression model can still fit binary data, but it will produce wrong fitting results. To illustrate this point, in Figure.6 we compare the accuracy of the classification model with the regression models used in the primary analysis. The results show that Boosting, RF, TREE, GAM and MARS have higher accuracy, indicating a better fit. Besides, according to Figure.6, the boosted tree model have the largest R squared value, so it should be chosen as the final model with the best performance.

Discussion

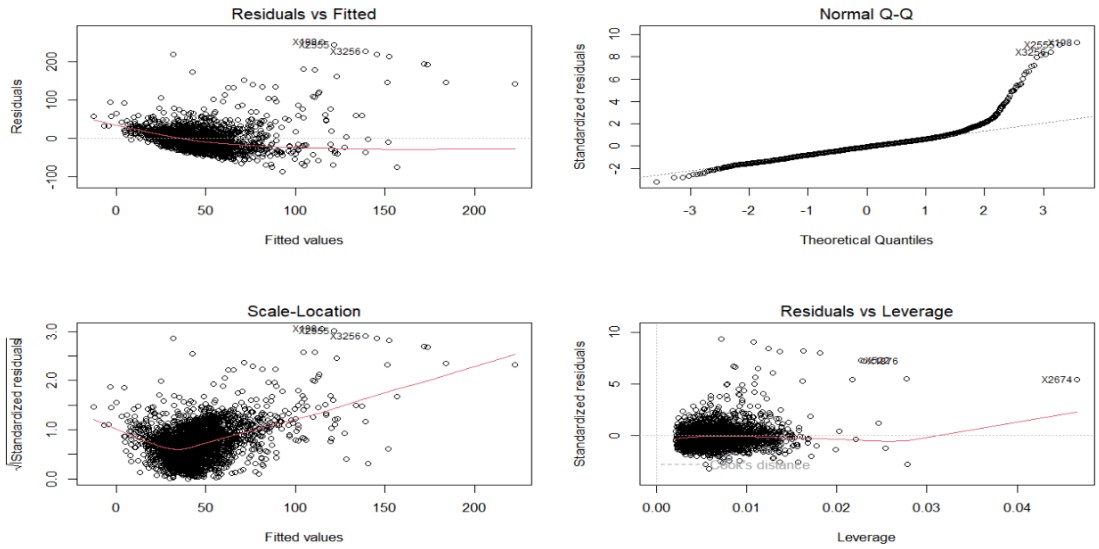
From the EDA above, we observed high correlation between bmi and recovery time, which supports our finding in primary analysis that bmi is a significant variable to predict recovery time in the final model. However, study type, as the second most important predictor of the final model, has no significant correlation with recovery time in EDA. We presumed that there might be interaction among study and some other variables, which requires further research and analysis in the future.

Conclusion

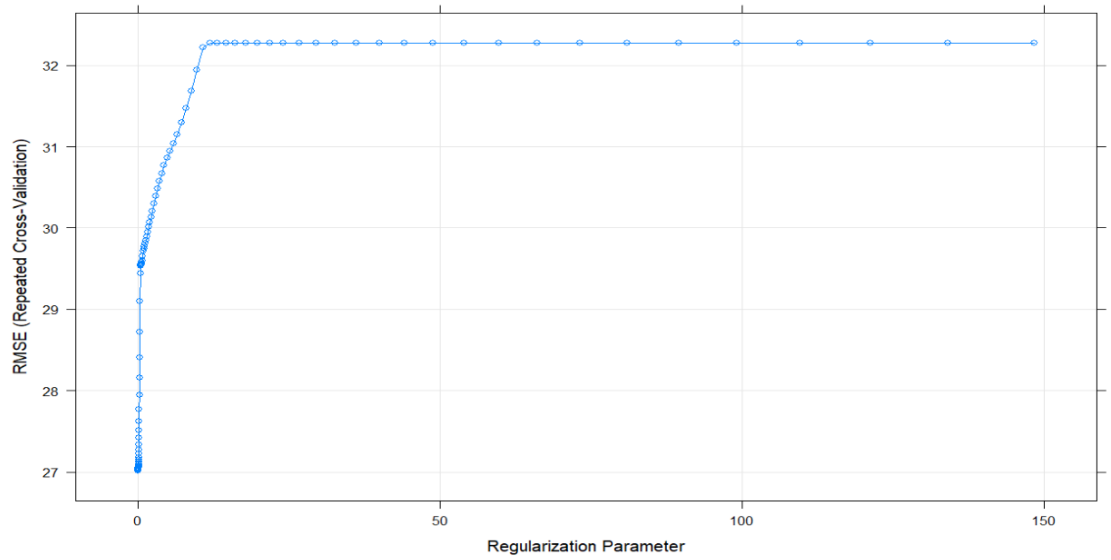
To identify important risk factors for longer recovery time and gain a better understanding of the predictors of recovery time from COVID-19, we compared eleven models both for regression and classification respectively with a dataset of 3595 observations and 14 predictors. Among them, the boosted tree models show the greatest performance on predicting time to recovery from COVID-19, with RMSE 18.99 for regression and R-Squared 70.81% for classification.

Appendix

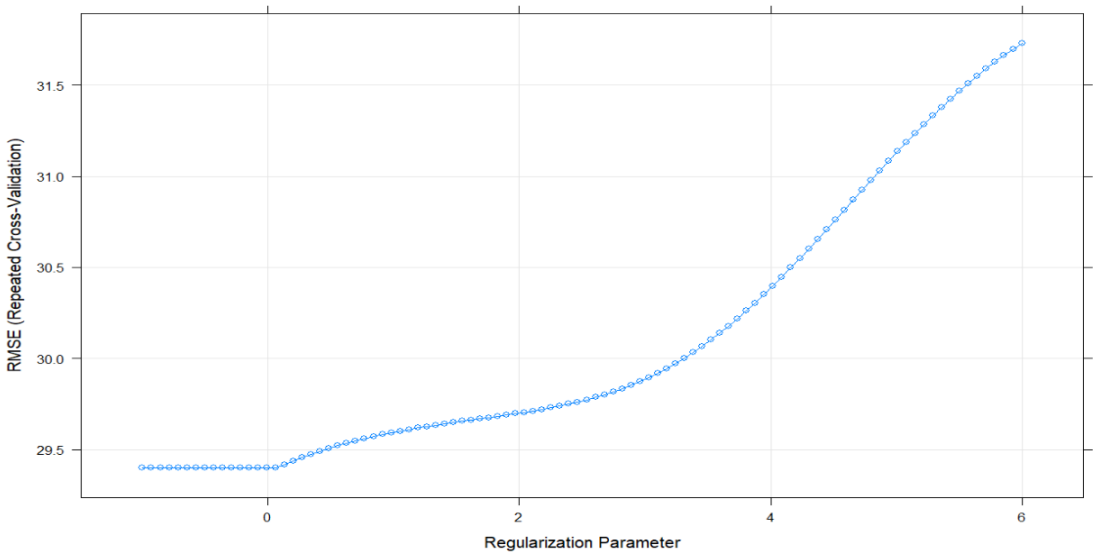
Tables and figures for regression



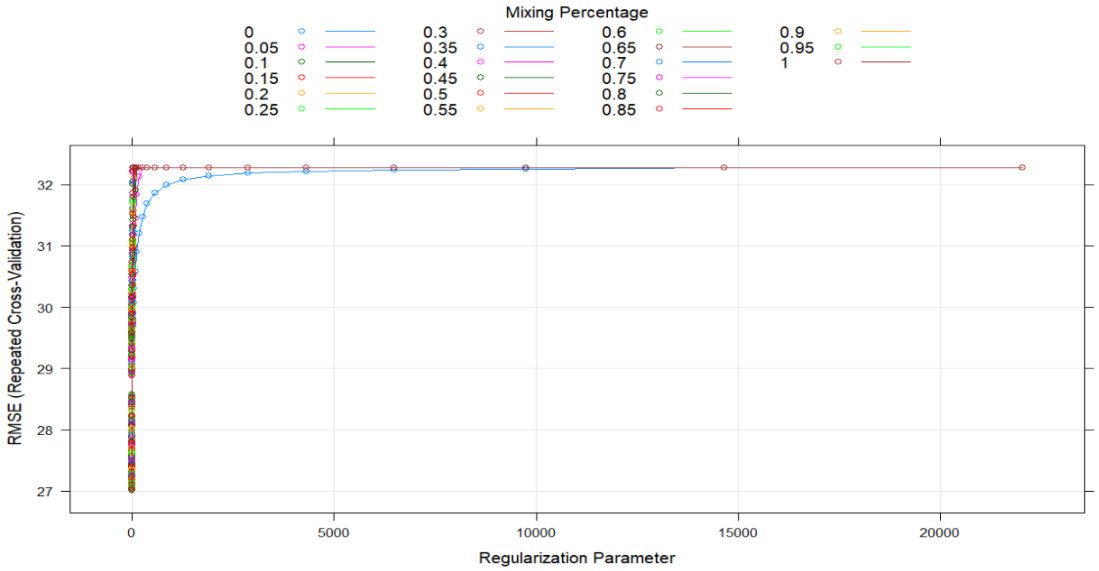
Appendix I. Linear model



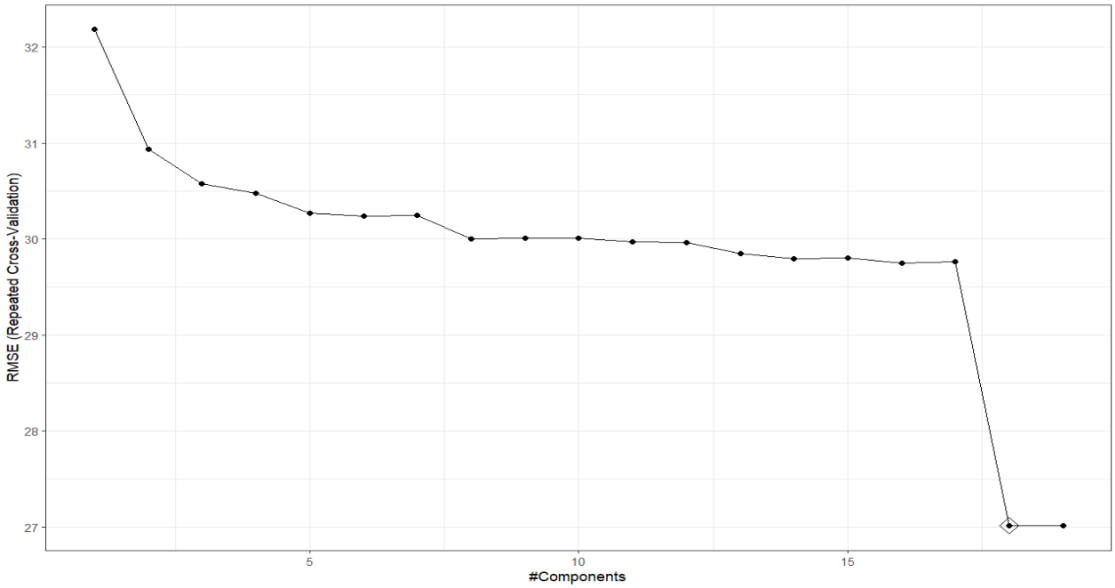
Appendix II. Lasso



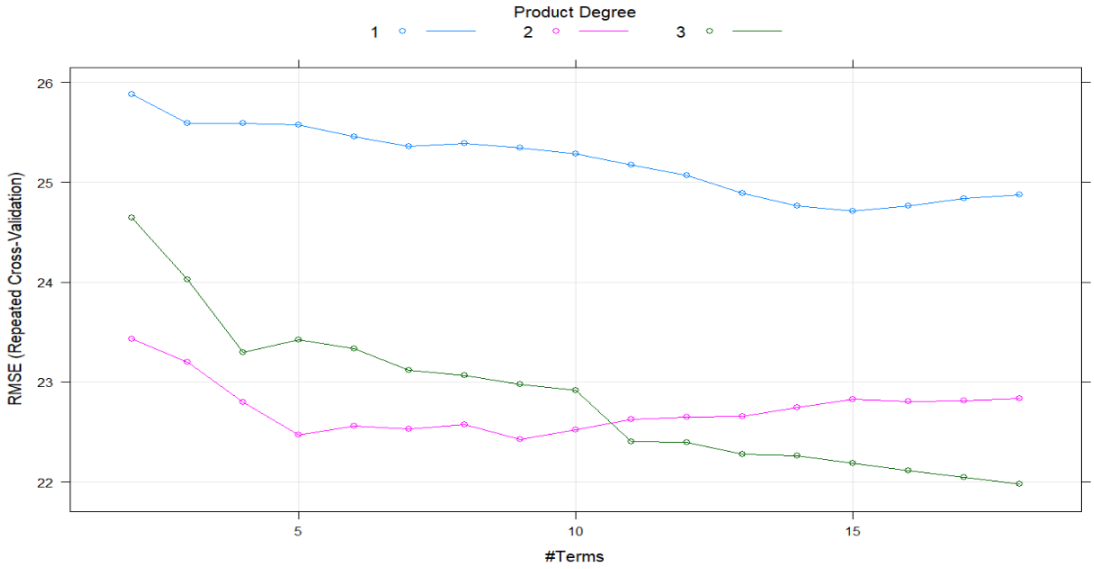
Appendix III. Ridge



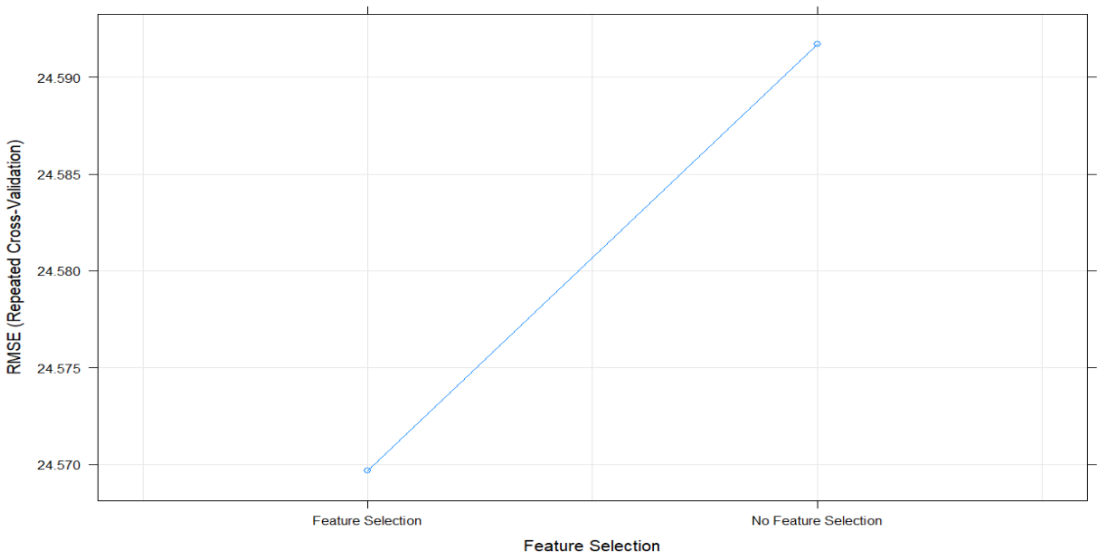
Appendix IV. Elastic net



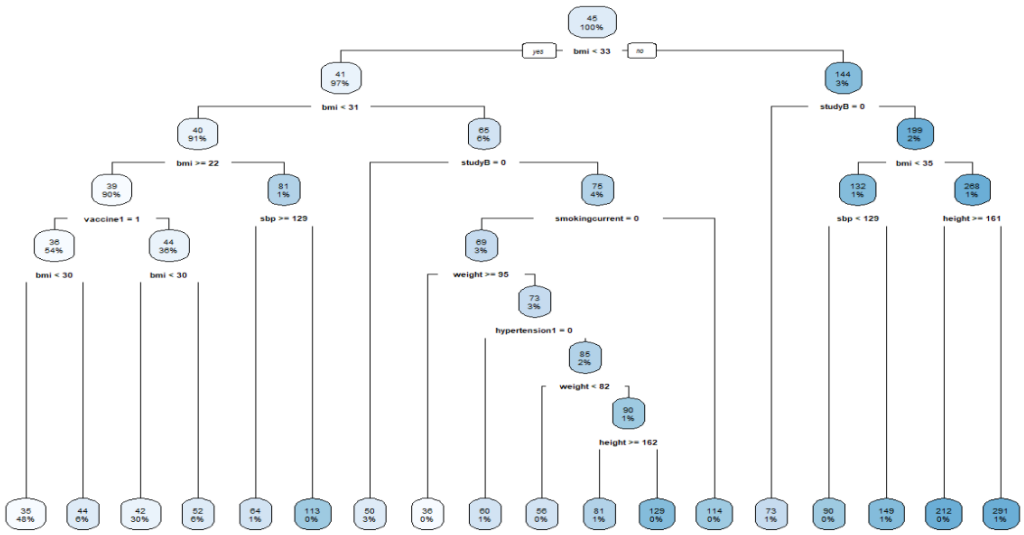
Appendix V. PCR



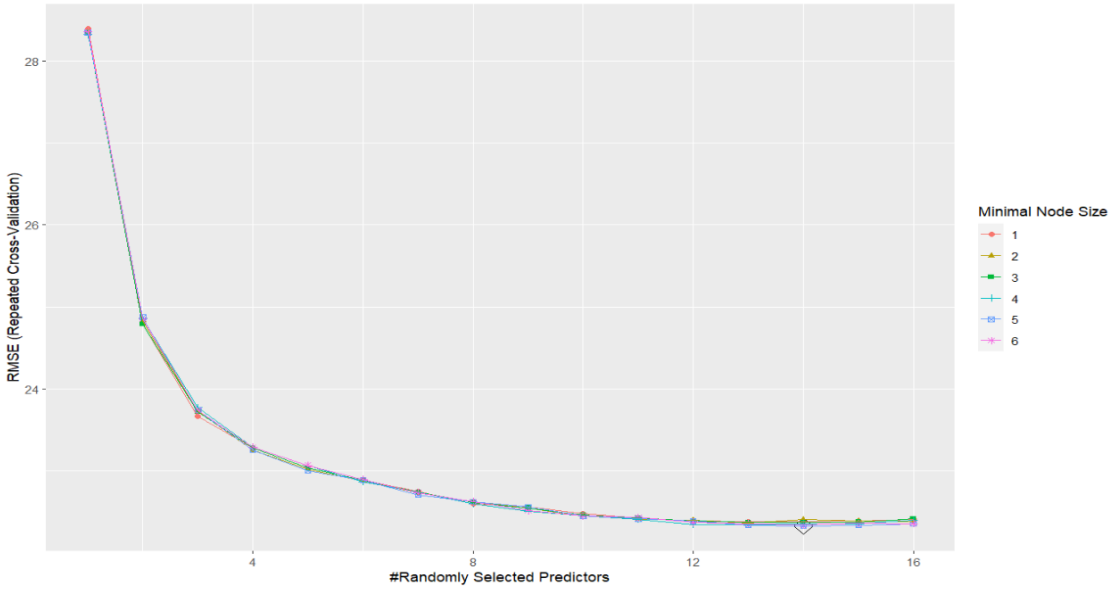
Appendix VI. MARS



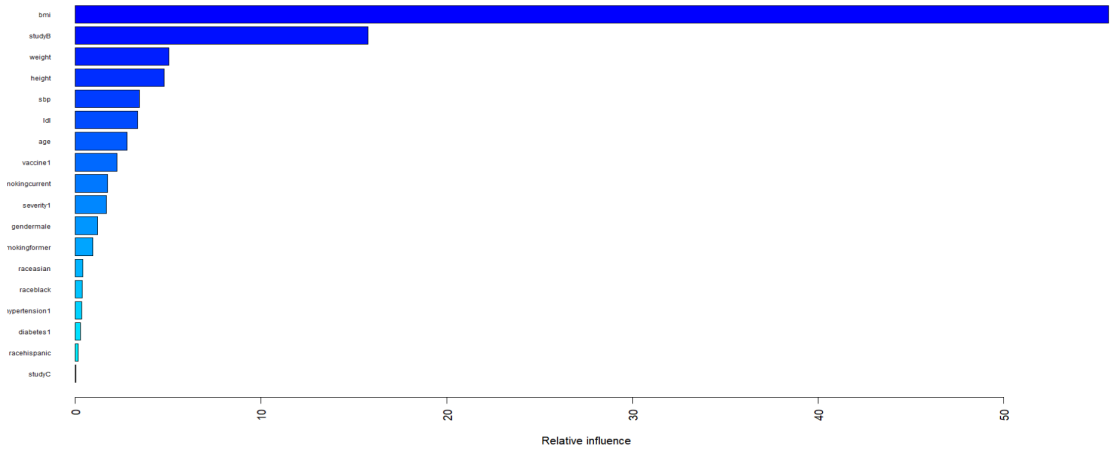
Appendix VII. GAM



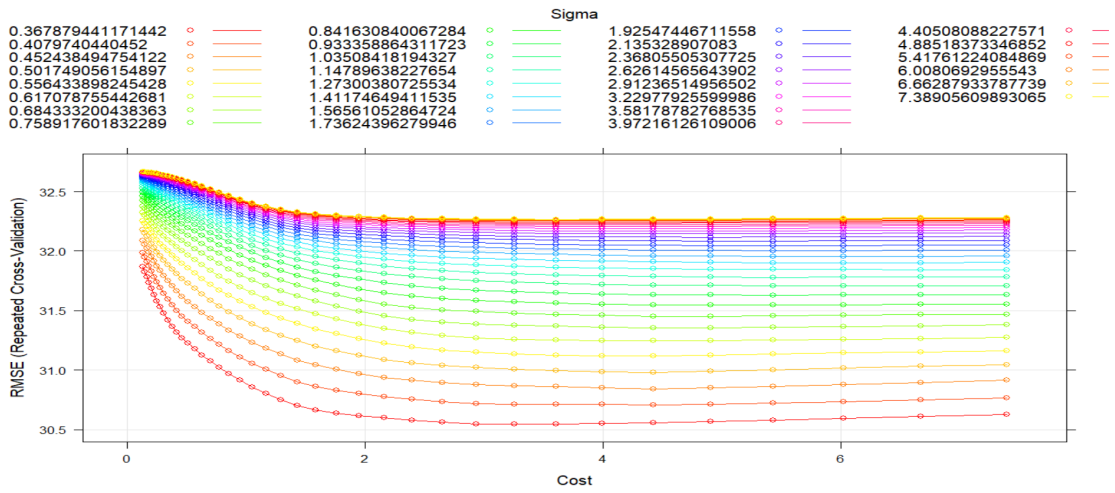
Appendix VIII. Regression tree



Appendix VIIII. Bagging

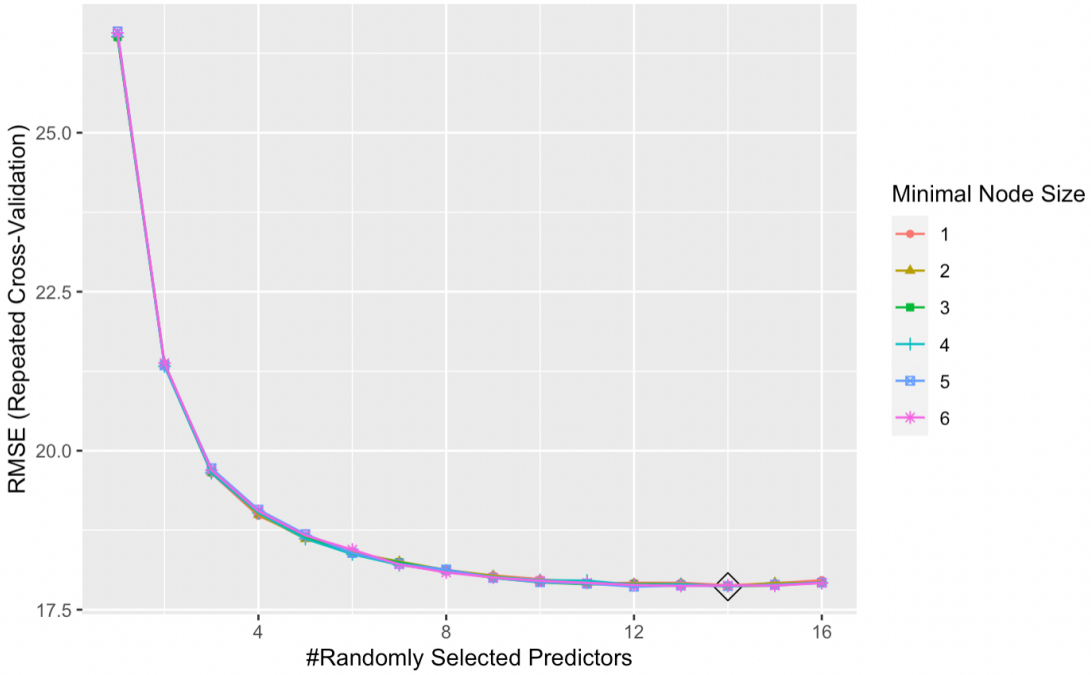


Appendix X. Boosted tree model

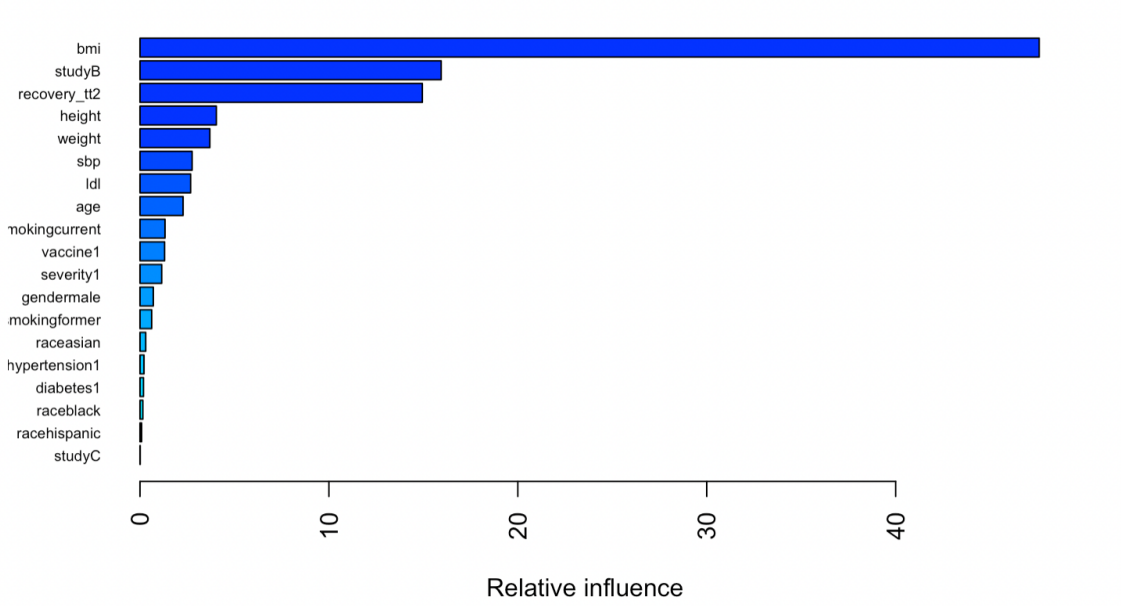


Appendix XI. SVM

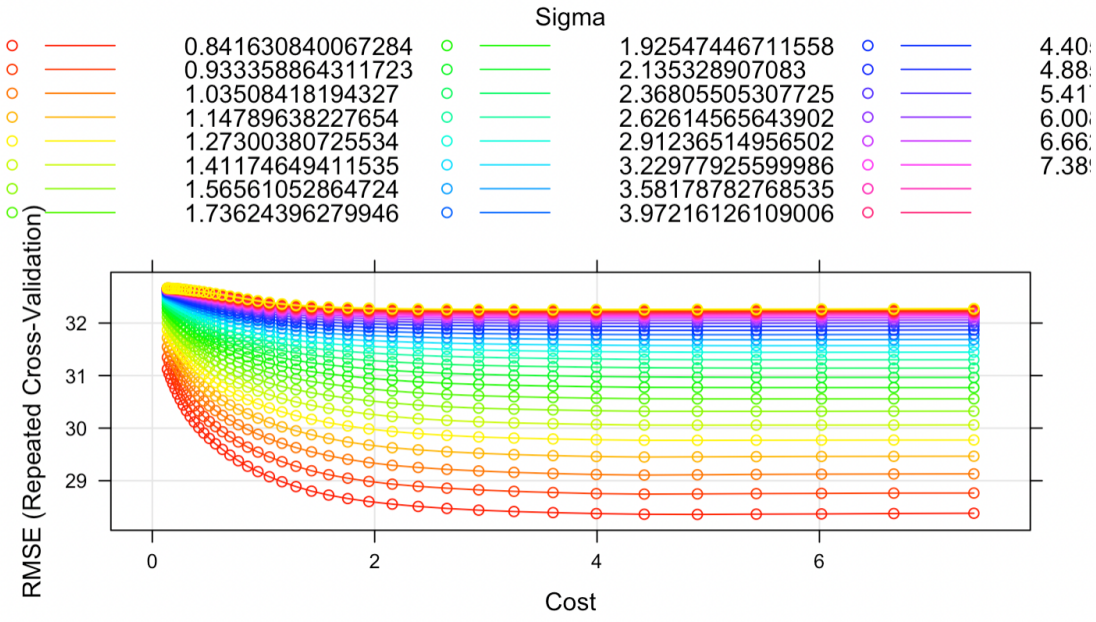
Tables and figures for classification



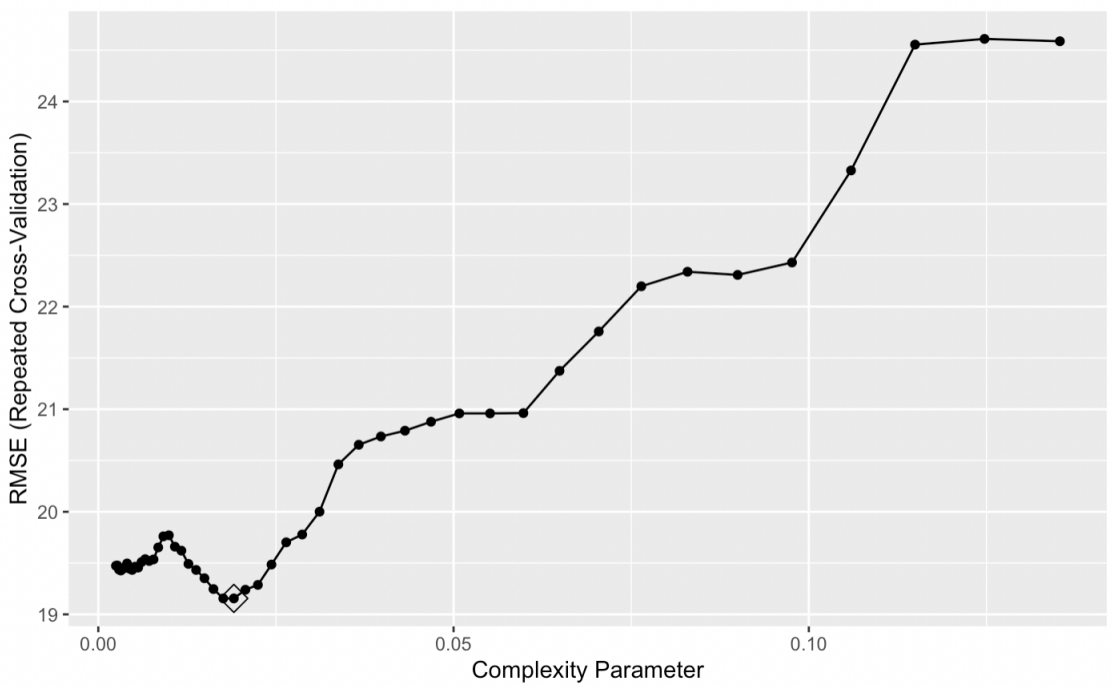
Appendix XII. Bagging



Appendix XIII. Boosting



Appendix XIV. SVM



Appendix XV. Classification Tree