P8106 Final Project
Predictive Modeling for COVID-19 Recovery Time

Tianshu Liu, Jiong Ma, Lincole Jiang

May 2023

Contents

1 Introduction

COVID-19 (Coronavirus Disease 2019) is an infectious disease caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). For this study, we analyze data from three existing cohort studies that collected recovery information through questionnaires and medical records and existing data leveraged on personal characteristics prior to the pandemic. The primary analysis considers regression models that treat recovery time as a continuous response; whereas the secondary analysis trains classification models that treat recovery time as a binary outcome (30 days as benchmark). The ultimate goal is to develop models for recovery time prediction purpose and identify important risk factors for long recovery time. The optimal regression and classification models are selected based on the lowest root mean square error (RMSE) and highest accuracy in resampling, respectively.

2 Exploratory Analysis and Data Visualization

The dataset for our analysis was selected randomly from the original dataset with 10000 observations. Using the `sample` function from base `R`, two random samples of size 2000 are drawn using random seed `3196` and `2575`, keeping 3623 unique observations. Before model training, the dataset is partitioned using the `createDataPartition()` in `caret` package into training data (80%, n=2900) and testing data (20%, n=723) with random seed `2023`. Here exploratory analysis and data visualization are conducted on the training data only.

Our dataset contains 14 predictors, 6 of which are numeric and 8 of which are categorical. `recovery_time` is directly used as a continuous response in primary analysis and is converted to a binary factor outcome ($>$ 30 days as `gt30` vs. $\leq$ 30 days as `lt30`) in secondary analysis. The summary of our dataset is shown in Table 1 generated by `skim()` in `skimr` package.

A more in-depth exploratory analysis and visualization is detailed in Figure 1: for primary analysis, Figure 1(a) shows the scatter plots of the continuous outcome w.r.t each continuous variable with smoothing lines, Figure 1(b) displays grouped boxplots of the continuous response w.r.t categorical variables. Similarly, for secondary analysis, Figure 1(c) shows boxplots of each continuous variable w.r.t. the two response levels and Figure 1(d) shows barplots of each categorical variable w.r.t the response levels. In particular, we note a roughly U-shaped trend between bmi and the continuous recovery time and somewhat plausible associations between gender, smoking status, vaccination status, and severity of infection with recovery time. To visualize the distribution of the response variable, a histogram in Figure 1(e) is plotted for the continuous outcome, which has a slightly right-skewed bell-shape distribution; and a barplot for the classified binary outcome shown in Figure 1(f), indicating imbalanced response: the number of observations with recovery time greater than 30 is roughly 2.5 times more than the the number of observations with recovery time less than or equal to 30. Finally, to examine multicollinearity of explanatory variables, the `corrplot()` function from the `corrplot` package is used to sketch a correlation plot of continuous explanatory variables in Figure 1(g), which shows substantial correlation between SBP levels and age as well as bmi and height.

3 Model Training

In this section, we detail the model tuning process for both the regression models in the primary analysis as well as the classification models for secondary analysis. All the categorical variable are transformed to dummy variables prior to model fitting, so there are 18 predictors in total. The process of model training is conducted with `caret` package using the following steps for each model:

1. Define the training control object using `trainControl()`. In this project, 10-fold cross validation (CV) is conducted in each model training process.

2. Set random seed of `2023`, train the model using `train()`, passing formula, training data, and the training control object. RMSE and Rsquared are used as the summary metric in primary analysis, Accuracy and Kappa are used in secondary analysis.

3. Visualize the CV performance and select the best tuning parameters with minimum average CV error or maximum average CV accuracy.

3.1 Primary Analysis

3.1.1 Model Tuning

1. **Linear model (LM)** assumes linearity, homoscedasticity, independent observations, normality of residuals or error terms and no multicolinearity. A linear model without any tuning parameters can be fitted by using `method = "lm"`.

2. **Lasso** model is a regularized linear model by plugging in a lasso penalty term. Besides the assumptions in linear model, it also assumes all the variables are transformed to be on the same scale and the true model is sparse. When training a lasso model using `method = "glmnet"`, the tuning parameters contains `alpha = 1` and `lambda` using a sequence of 100 values from e^{-7} to e^{0}. Figure 2(a) shows CV result for lasso model. The best tuning parameter is `lambda = 0.001495865`.

3. **Ridge** model is a regularized linear model using a ridge penalty. It also assumes that the independent variables are scaled. When setting `method = "glmnet"`, the tuning parameters are `alpha = 0` and `lambda` ranging from e^{-5} to e^{1}. Figure 2(b) shows CV result for ridge model. The best tuning parameter for `lambda = 0.8594049`.

4. **Elastic Net** is another type of regularized linear regression model combining both lasso and ridge penalties whose assumptions are similar to the lasso and ridge model. When using `method = "glmnet"`, the grid of tuning parameters contains 2 parameters: `alpha` using a sequence of 11 values from `0` to `1` and `lambda` using a sequence of 50 values from e^{-8} to e^{0}. Figure 2(c) shows CV result for elastic net model. The best tuning parameter are `alpha = 0.9` and `lambda = 0.001051915`.

5. **Principal Component Regression (PCR)** is a technique to do dimension reduction and regression using principal components which assumes that the variables are centered and scaled and the response does not supervise the identification of the principal components. A PCR model can be trained by specifying `method = "pcr"`. The tuning range of `ncomp` is from 1 to 18. The CV result is shown in Figure 2(d). The best tuning parameter is `ncomp = 18` indicating no dimension reduction is conducted.

6. **Partial Least Square (PLS)** reduces the data dimensions by creating latent variables as predictors in a regression model, which shares almost similar assumptions with PCR model but identifies new features in a supervised way. A PLS model can be fitted by using `method = "pls"`. The tuning range of `ncomp` is from 1 to 18. Figure 2(e) shows the CV result. The best tuning parameter is `ncomp = 13`.

7. **Generalized Additive Model (GAM)** is a regression model which allows for flexible nonlinearities in several variables, but retains the additive structure of linear models, as well as for interactions between independent variables. Additive smooth functions are used in GAM model which are assumed to be differentiable at all points and do not interact with each other. When specifying `method = "gam"`, the grid of tuning parameters contains `method = "GCV.Cp"` and `select` to be `TRUE` or `FALSE`. Figure 2(f) shows CV result for GAM model. The best tuning parameter is `select = TRUE`.

8. **Multivariate Adaptive Regression Splines (MARS)** is a regression technique that creates piece wise linear models using hinge function to handle nonlinear relationships. Thus, MARS assumes that the predictors have a linear relationship with the response within each segment separated by cut points. When training a MARS model using `method = "earth"`, the grid of tuning parameters contains 2 parameters: `degree` ranging from 1 to 5 and `nprune` ranging from 2 to 14. Figure 2(g) shows CV result for MARS model. The best tuning parameters are `degree = 4` and `nprune = 6`.

9. **K-nearest Neighbour (KNN)** is a non-parametric algorithm based on the distance between the predictors. KNN assumes the predictor variables are measured on the same scale, no missing values, and similarity in response between observations close to each other in the predictor space. When training a KNN model using `method = "knn"`, there is only one tuning parameter k ranging from 1 to 20. Figure 2(h) shows CV result for KNN model. The best tuning parameter is `k = 6`.

10. **Bagging** is an ensemble technique that averages the predictions of multiple individual base models to reduce the variance. It assumes diverse base models are trained independently on each bootstrapped training set. It can be implemented by using `method = "ranger"`. The tuning grid contains 3 parameters: `mtry = 18`, `splitrule = "variance"` and `min.node.size` ranging from 1 to 20. The CV result is shown in Figure 2(i). The best tuning parameter is `min.node.size = 16`.

11. **Random Forest (RF)** provides an improvement of bagging by introducing randomness to decorrelate the trees. A number of decision trees with randomly chosen predictors are independently built on bootstrapped training samples. When training RF using `method = "ranger"`, the tuning parameters are `mtry` ranging from 1 to 18 which is the number of randomly selected predictors, `splitrule = "variance"` and `min.node.size` ranging from 12 to 18. The CV result is shown in Figure 2(j). The best tuning parameters are `mtry = 14` and `min.node.size = 16`.

12. **Boosting** is an ensemble model that combines weak learners to create a strong predictions. Unlike bagging and RF, trees in boosting are grown sequentially. A boosting model can be trained by specifying `method = "gbm"`. The tuning parameters are `n.trees` ranging from 2000 to 4000, `interaction.depth` ranging from 1 to 4, `shrinkage` ranging from 0.001 to 0.005, and `n.minobsinnode` to be 1 or 10. The CV result is shown in Figure 2(k). The best tuning parameters are `n.trees = 3000`, `interaction.depth = 3`, `shrinkage = 0.0025` and `n.minobsinnode = 1`.

13. **Regression tree** is a supervised algorithm that segments the predictor space into a number of simple regions and fits a simple model in each region. It is a piecewise constant model that assumes a hierarchical structure in the data, where the response can be divided into segments based on the values of the input features. It can be implemented by specifying `method = "rpart"` with `cp` as the tuning parameter ranging from e^{-6} to e^{-3}. The CV result is shown in Figure 2(l). The best tuning parameter is `cp = 0.04251515`.

3.1.2 Model Selection

`resamples()` is used to compare models based on CV results. Figure 4(a) shows the resampled result of all regression models based on CV RMSE and Rsquared. Boosting is selected as the final regression model with the lowest median CV RMSE (22.55 in CV RMSE, 0.4937 in CV Rsquared).

3.2 Secondary Analysis

3.2.1 Model Tuning

1. **Logistic Regression** assumes linearity between the log odd of the binary outcome and the independent predictors. A logistic model without any tuning parameters is fitted by using `method = "glm"`.

2. **Penalized Logistic Regression** is a variation of logistic regression that incorporates penalty terms. Thus, it has the same assumptions as logistic regression but strongly recommends to scale the predictors. It can be fitted by using `method = "glmnet"`. There are 2 tuning parameters including `alpha` ranging from 0 to 1 and `lambda` ranging from the e^{-10} to e^{-5}. Figure 3(a) shows the CV result. The best tuning parameters are `alpha = 0.95` and `lambda = 0.00387`.

3. **GAM** for binary response has the same assumptions as continuous response. A GAM model with `select` as tuning parameter can be fitted by specifying `method = "gam"`. The CV result is shown in 3(b). The best tuning parameter is selected as `select = TRUE`.

4. **MARS** for binary response has the same assumptions as continuous response. It can be fitted by using `method = "earth"`. The tuning parameters are `degree`, ranging from 1 to 3, and the `nprune`, ranging from 2 to the total number of the predictors. The CV result is shown in 3(c). The best tuning parameters are `nprune = 11` and `degree = 1`.

5. **Linear Discriminant Analysis(LDA)** is a dimensionality reduction and classification technique used to find linear combinations of features that best separate multiple classes or groups in a dataset. The key assumptions include normality, homoscedasticity, linearity, independence, large sample size and balanced classes. A LDA model without any tuning parameters can be fitted by using `method = "lda"`.

6. **Quadratic Discriminant Analysis(QDA)** is a classification algorithm similar to LDA but allows for more flexible decision boundaries. The assumptions of QDA are similar to LDA, but the linearity assumption is relaxed. A QDA model without any tuning parameters can be fitted by using the `method = "qda"`.

7. **Naive Bayes(NB)** is a classification algorithm based on Bayes' theorem which assumes that features are independent of each other, given the class variable. It can be fitted by specifying `method = "nb"`. The 3 parameters in the tuning grid are `usekernel` set to be `FALSE` or `TRUE`, `adjust` ranging from 0.1 to 1, and `fL = 1`. The CV result is shown in 3(d). The best tuning parameters are `usekernel = TRUE` and `adjust = 0.2`.

8. **Bagging** for binary response has the same assumptions as continuous response. It is implemented by using the `method = "ranger"`. The tuning parameters include `mtry = 18`, `splitrule = "gini"`, and `min.node.size` ranging from 1 to 19. The CV result is shown in 3(e). The best tuning parameter of `min.node.size` is `15`.

9. **Random Forest (RF)** for binary response has the same assumptions as continuous response. It is implemented by using the `method = "ranger"`. The tuning parameters are `mtry` ranging from 1 to 18, `splitrule = "gini"`, and `min.node.size` ranging from 10 to 18. The CV result is shown in 3(f). The best tuning parameters are `min.node.size = 16`, and `mtry = 4`.

10. **Boosting** for binary response has the same assumptions as continuous response. It is fitted by using `method = "gbm"`. The tuning parameters include `n.trees` ranging from 6000 to 8000, `interaction.depth` from 1 to 3, `shrinkage` from 0.0025 to 0.01, and `n.minobsinnode` to be 1 or 10 . The CV result is shown in 3(g). The best tuning parameters are: `n.trees = 7000`, `interaction.depth = 2`, `shrinkage = 0.005`, and `n.minobsinnode = 1`.

11. **Classification Tree** assumes binary splits, recursive partitioning, independence, relevant predictors, and balanced classes. It is fitted by using the `method = "rpart"`. The tuning parameter is `cp` ranging from e^{-6} to e^{-4}. The CV result is shown in 3(h). The best tuning parameter is `cp = 0.004961126`.

12. **Support Vector Machine(SVM)** aims to find an optimal hyperplane that maximally separates the data points of different classes. The linear SVM assumes linear separability, feature independence, feature scaling, large margin, and the noise-free data. It is fitted by using `method = "svmLinear"`. The tuning parameter `C` is set to range from e^{-6} to e^{3} but the best tuning parameter tends to approach 0. SVM with radial kernel (SVMR) is effective when the data is not linearly separable. It is fitted by specifying `method = "svmRadialSigma"`. The 2 tuning parameters are `C` ranging from e^{-2} to e^{5} and `sigma` ranging from e^{-4} to e^{1}. The CV result is shown in 3(i). The best tuning parameters are `sigma = 0.1353353` and `C = 0.8539397`.

3.2.2 Model Selection

Accuracy and Kappa are used as metrics to select classification models. The resampled results of models are shown in Figure 4(b). SVMR is selected as the final model with highest median CV accuracy (0.7190 in accuracy, 0.2018 in Kappa statistic).

4 Results

4.1 Interpretation

Since boosting, the optimal regression model, is often considered as a black-box model, a variable importance plot (VIP) shown in Figure 5(a) can help to identify the most significant contributors, including `BMI`, `studyB`, and `LDL`. For further investigation, partial dependence plots (PDPs) in Figure 5(b) indicates how `bmi` and `studyB` contribute to the prediction. Since `bmi` shows a roughly U-shaped curve, the same as expected from EDA, a patient with a too low or too high value of bmi tends to have longer recovery time from COVID-19. For those from study B, a high bmi value would lead to a large amount of recovery time

For SVMR is selected as the optimal classification model which does not provide prediction probabilities, we derive ad-hoc prediction probabilities and variable importance scores only for better interpretation. As shown in Figure 6(a), the most important variables in SVMR are `studyB`, `vaccine1`, and `studyC`, followed by `bmi`, `gender1`, and `height`. Figure 6(b) shows PDPs of these variables, indicating a patient from study C without vaccination is more likely to have recovery time greater than 30 days. Similar to the boosting model in primary analysis, PDP of `bmi` also shows a U-shaped curve, indicating that extreme bmi values may lead to longer recovery time which is consistent with the boosting model selected in primary analysis.

4.2 Training/Test Performance

The RMSE for the boosting model in training data and test data are 19.1267 and 22.3039, respectively. For the SVMR model, prediction accuracy is 0.7917 in training data and 0.7344 in test data.

5 Conclusions

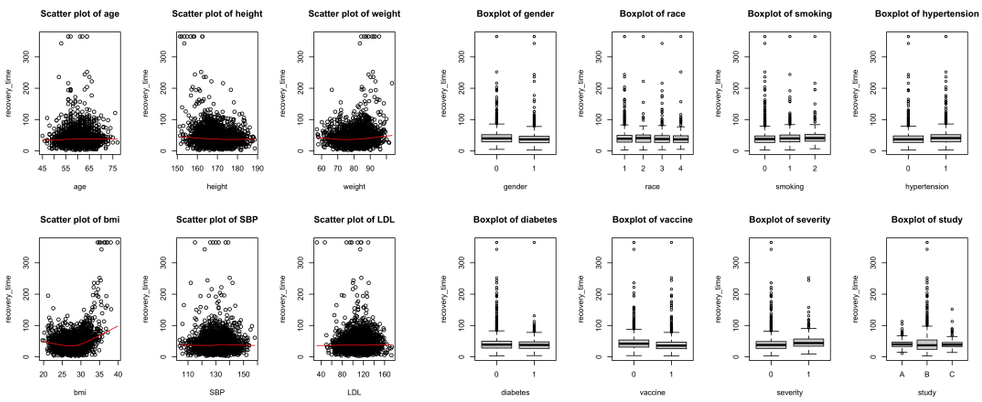
In this project, regression and classification models are fitted to predict recovery time from COVID-19. Boosting is finally selected as the optimal regression model based on 10-fold CV RMSE, and SVMR is selected as the optimal classification model based on 10-fold CV accuracy. Since both final models are non-linear mode and most non-linear models perform better than linear models especially in primary analysis, it may suggest the non-linear relationship between the response and the predictors of interest.

Both of the models emphasize `bmi` as a significant risk factor which has a U-shaped contribution to recovery time. Since `bmi` is an important measure of body fat based on height and weight. Both extremely high or low bmi values can prolong recovery time especially for those patients from study B. To recover quickly from COVID-19, it's important to maintain a healthy weight by adopting a sustainable lifestyle.

Table 1: Training data summary table

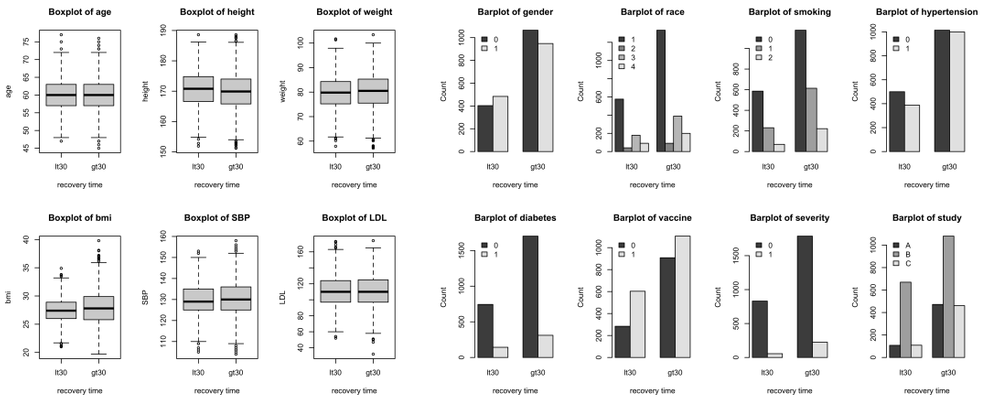
	type	name	n_missing	factor_levels	factor_counts
1	factor	gender	0	2	0: 1468, 1: 1432
2	factor	race	0	4	1: 1909, 3: 568, 4: 291, 2: 132
3	factor	smoking	0	3	0: 1763, 1: 845, 2: 292
4	factor	hypertension	0	2	0: 1514, 1: 1386
5	factor	diabetes	0	2	0: 2446, 1: 454
6	factor	vaccine	0	2	1: 1708, 0: 1192
7	factor	severity	0	2	0: 2619, 1: 281
8	factor	study	0	3	B: 1750, A: 580, C: 570

	type	name	n_missing	mean	sd	p0	p25	p50	p75	p100
9	numeric	age	0	60.0672414	4.51267381	45	57	60	63	77
10	numeric	height	0	170.167207	6.03841364	151.2	166.1	170.15	174.1	188.6
11	numeric	weight	0	80.1959655	6.9998749	57.1	75.4	80.3	84.9	103.4
12	numeric	bmi	0	27.7552414	2.72664281	19.7	25.9	27.7	29.5	39.8
13	numeric	SBP	0	130.192414	8.08421168	104	125	130	136	158
14	numeric	LDL	0	110.26931	19.8722782	32	97	110	124	174
15	numeric	recovery_time	0	43.0248276	30.5053912	3	28	38	49	365



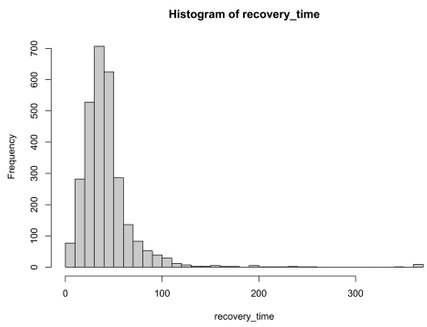
(a) Scatter plot of continuous variables

(b) Boxplots of categorical variables

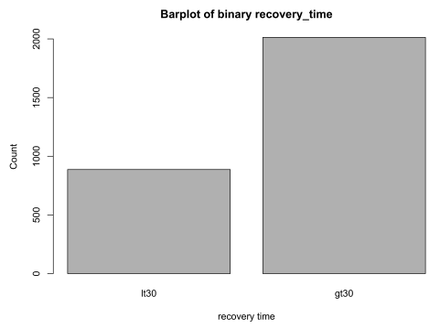


(c) Boxplots of continuous variables

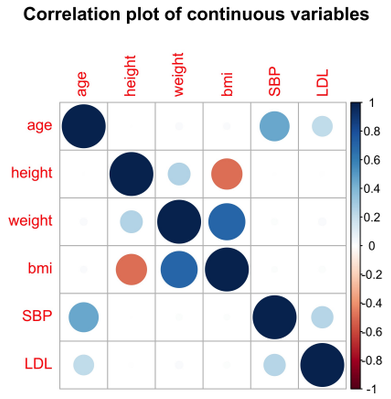
(d) Barplots of categorical variables



(e) Histogram of continuous response

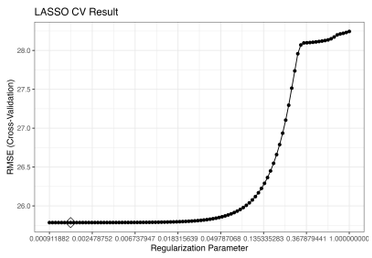


(f) Barplot of binary response

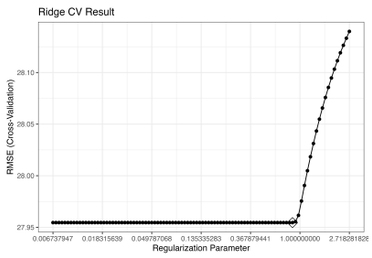


(g) Correlation plot of continuous variables

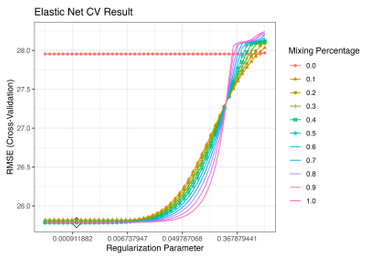
Figure 1: EDA plots



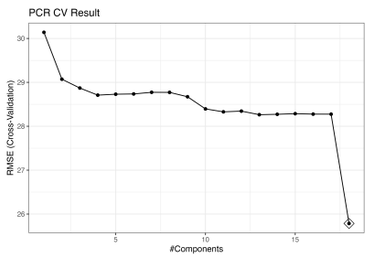
(a) LASSO



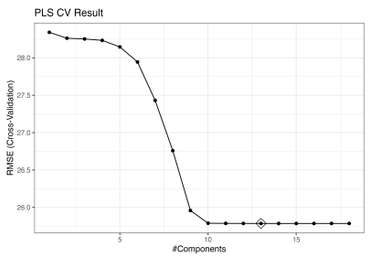
(b) Ridge



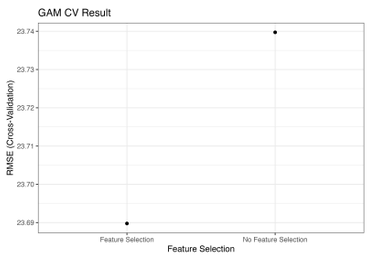
(c) Elastic net



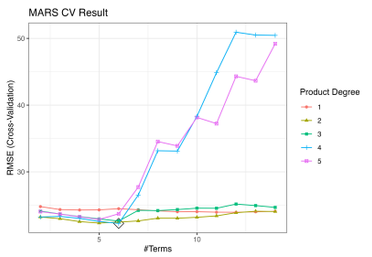
(d) PCR



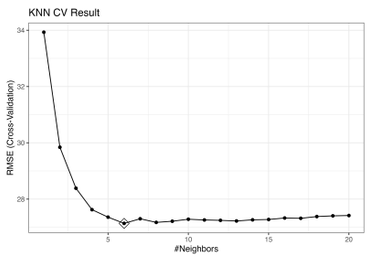
(e) PLS



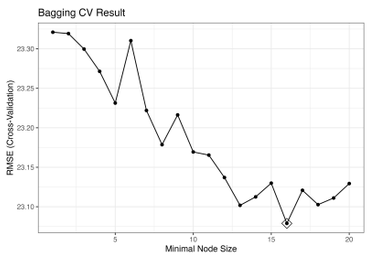
(f) GAM



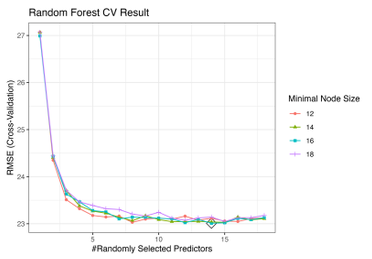
(g) MARS



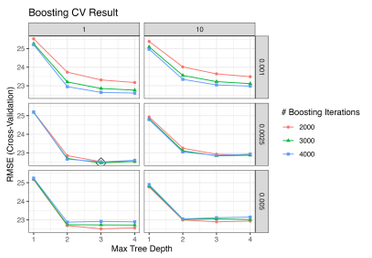
(h) KNN



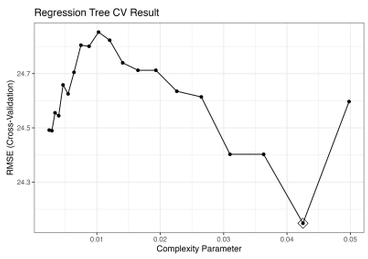
(i) Bagging



(j) Random Forest

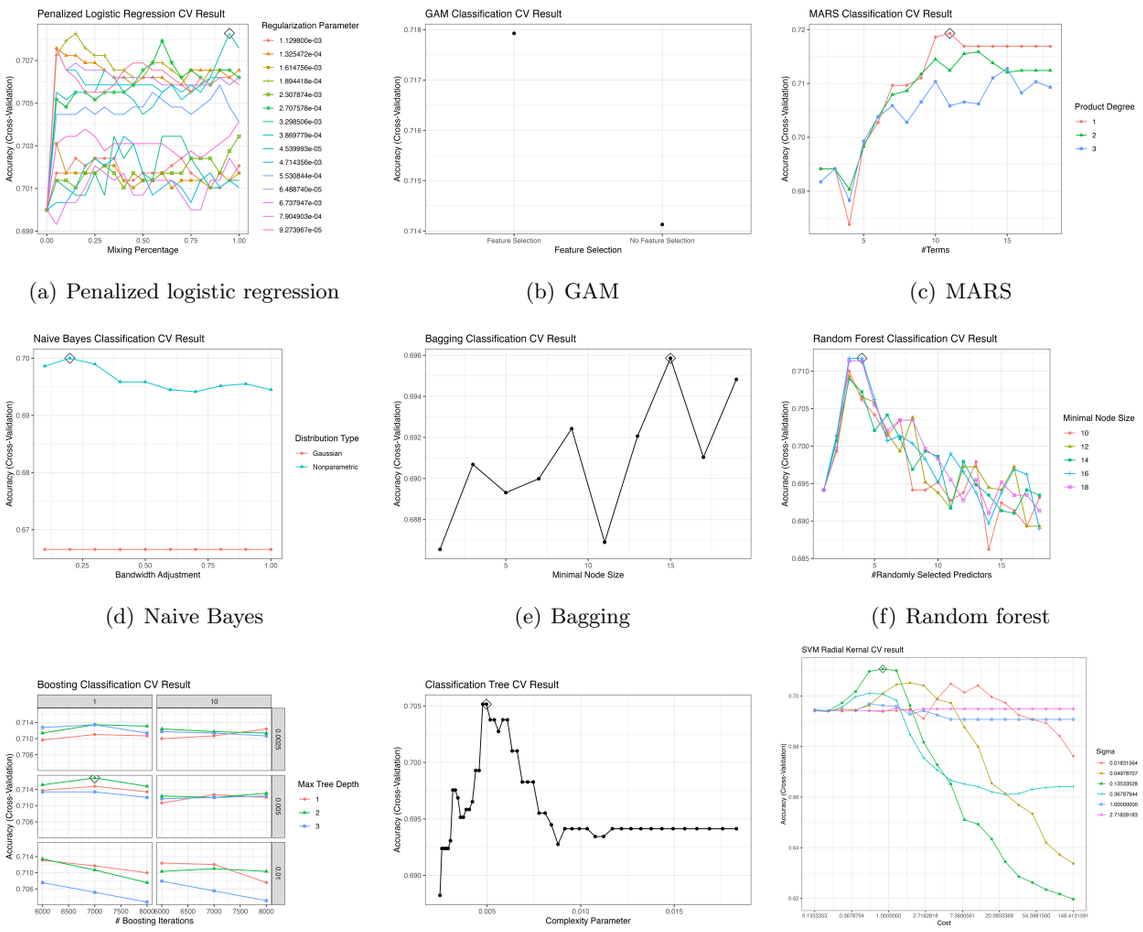


(k) Boosting



(l) Regression tree

Figure 2: Primary analysis CV results

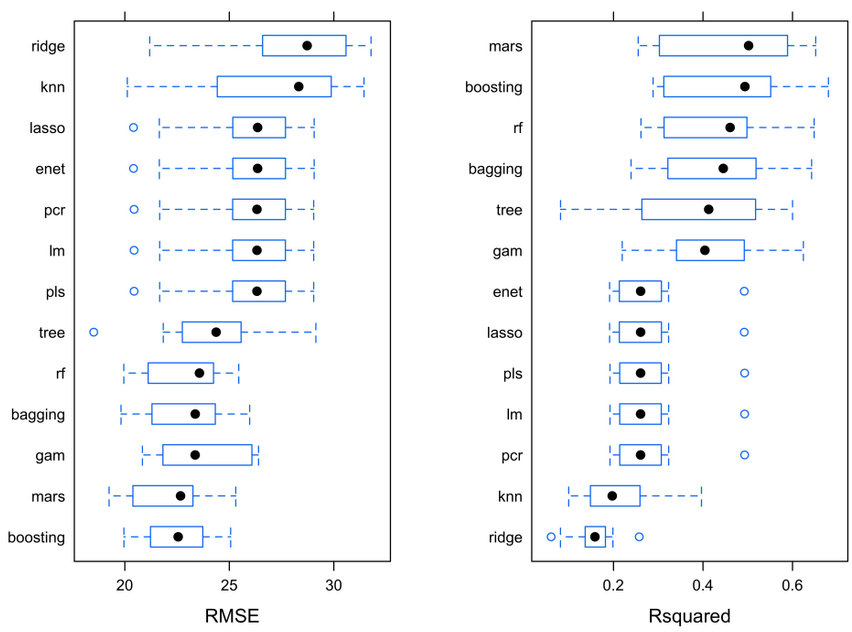


(a) Penalized logistic regression (b) GAM (c) MARS

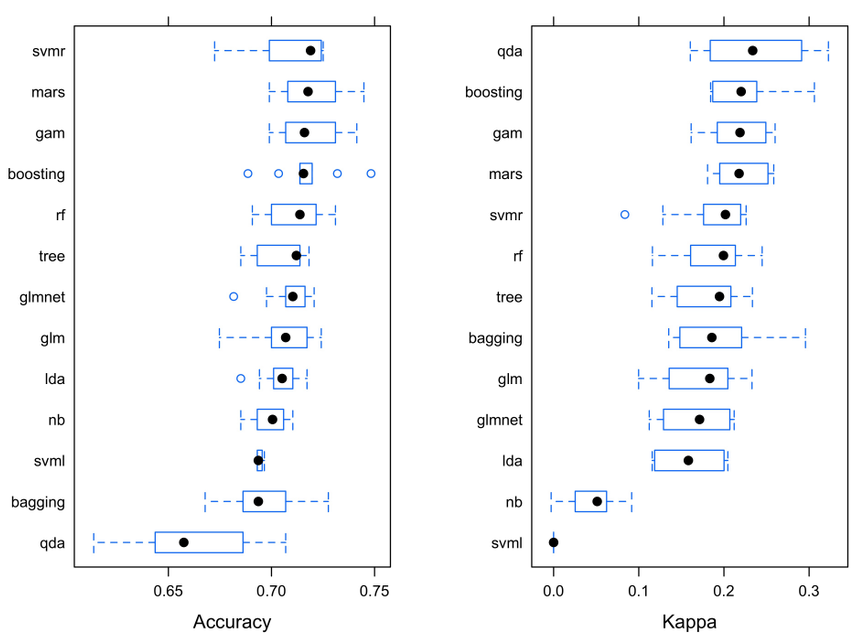
(d) Naive Bayes (e) Bagging (f) Random forest

(g) Boosting (h) Classification tree (i) SVMR

Figure 3: Secondary analysis CV results

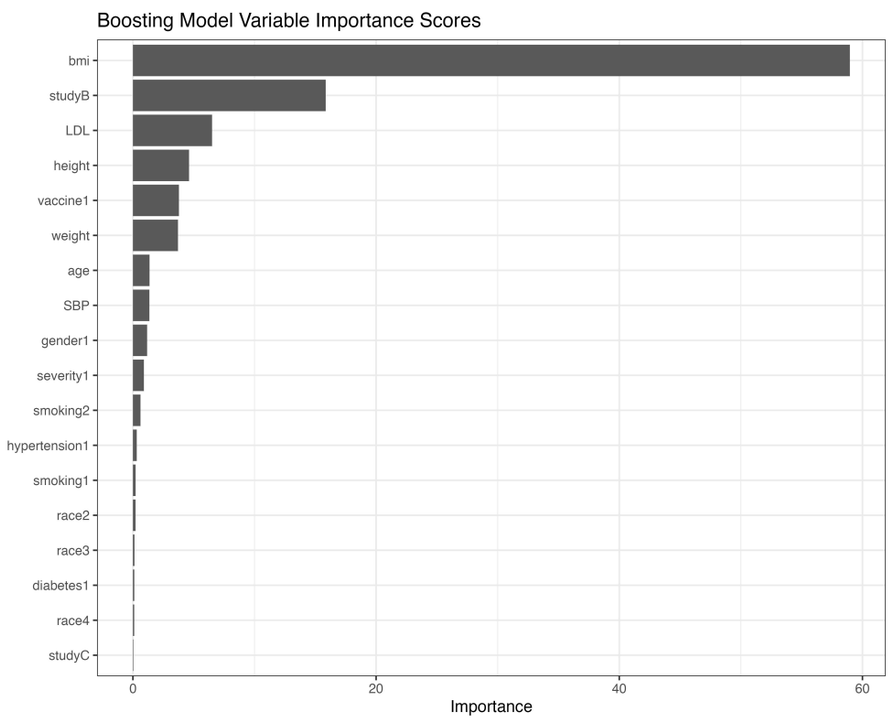


(a) Primary analysis

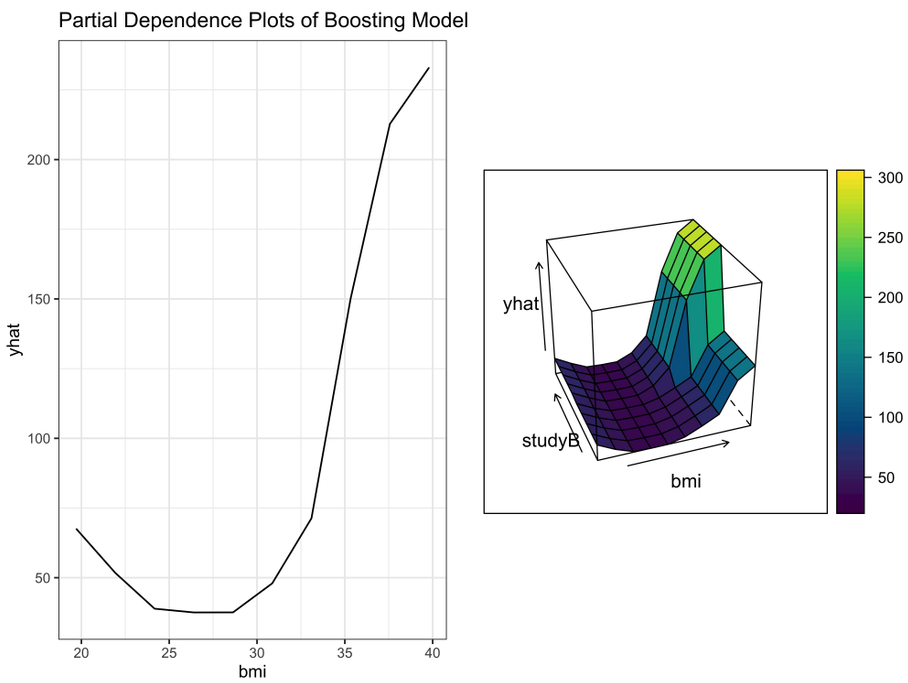


(b) Secondary analysis

Figure 4: Resample results

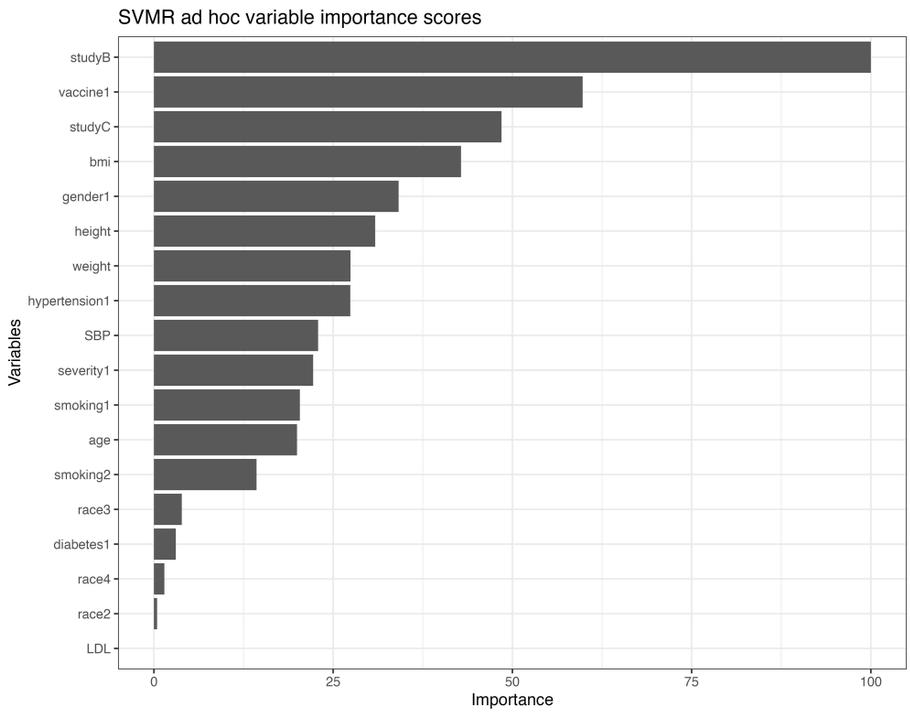


(a) Variable importance scores

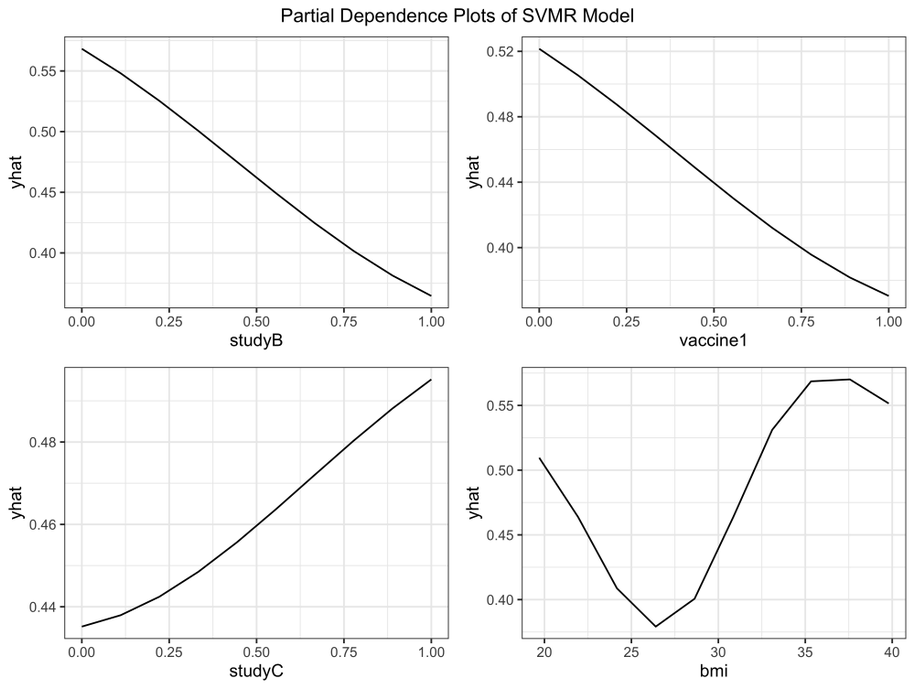


(b) Partial dependence plots

Figure 5: Boosting model interpretation



(a) Variable importance scores



(b) Partial dependence plots

Figure 6: SVMR model interpretation